Contributions à la cryptographie post-quantique

Jean-Christophe Deneuville

Thèse de Doctorat de l'Université de Limoges

Spécialité

Informatique et Applications

École Doctorale 521 : Sciences et Ingénierie pour l'Information, Mathématiques

Année : 2016

Présentée et soutenue publiquement par

Jean-Christophe Deneuville

le 1er Décembre 2016, pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Informatique

Contributions à la Cryptographie Post-Quantique

JURY :

Philippe Gaborit	Professeur, Université de Limoges	Directeur
Carlos Aguilar Melchor	Maître de Conférence HDR, Université de Toulouse	Co-directeur
Jean-Pierre Tillich	Directeur de Recherche, INRIA Rocquencourt	Rapporteur
Caroline Fontaine	Chargée de Recherche HDR, CNRS, Lab-STICC Télécom Bretagne	Rapporteuse
Ayoub Otmani	Professeur, Université de Rouen	Examinateur
Olivier Blazy	Maître de Conférence, Université de Limoges	Examinateur

"If we knew what it was we were doing, it would not be called research, would it ?"
"Si nous savions ce que nous faisons, ce ne serait plus de la recherche, n'est-ce pas ?"

Albert Einstein
1879 – 1955

À mes parents,
À mon épouse,

Remerciements

J'aimerais ouvrir ce manuscrit en exprimant ma reconnaissance et ma gratitude aux quelques personnes qui ont rendu possible ce travail. Philippe et Carlos, je suis de ceux qui pensent que de bons rapports entre une personne et ses supérieurs hiérarchiques stimulent à la fois la capacité et la qualité du travail. Au cours des quatre années qui viennent de s'écouler, vous m'avez soutenu sans relâche, encouragé (parfois botté les fesses, au sens figuré du terme, bien que l'envie de passer au sens propre eut été forte), supporté mes questions souvent évidentes, et parfois tourné au ridicule lorsque j'avais du mal à comprendre des notions pourtant élémentaires, mais toujours avec bienveillance et pour cela vous avez ma reconnaissance inconditionnelle. J'ai sincèrement apprécié travailler à vos côtés avec l'autonomie que vous m'avez accordée, et j'ai pu trouvé le degré d'explications que je cherchais tantôt chez toi Philippe, qui a cette incroyable capacité à prendre du recul sur le monde qui t'entoure, ou chez toi Carlos, technicien hors-pair, j'espère un jour parvenir à ton niveau de compréhension des détails. À vous deux, merci de m'avoir accordé votre confiance en me proposant cette thèse.

La fastidieuse relecture à titre gracieux d'un manuscrit de cent cinquante pages rédigées tantôt en français tantôt en anglais est une expérience ingrate, bien que partie entière de l'expérience d'un chercheur. Caroline, Jean-Pierre, je ne saurais que trop vous remercier d'avoir accepté cette lourde responsabilité. Je suis certain que votre expertise scientifique aura contribué à rendre ce manuscrit épuré d'erreurs. Ayoub, Olivier, merci également à vous deux d'avoir accepté d'examiner ce travail.

J'adresse également mes remerciements à mes co-auteurs (par ordre alphabétique) Carlos Aguilar Melchor, Olivier Blazy, Xavier Boyen, Philippe Gaborit, Tancrède Lepoint, Thomas Ricosset et Gilles Zémor. Travailler avec vous a été un réel plaisir, et je vous remercie – outre pour vos contributions scientifiques – de m'avoir souvent accompagné jusqu'au bout de la nuit dans ce sport doctoral dont j'ai fait ma spécialité et que j'appelle "*le sprint à la deadline*".

Je souhaite également remercier nos secrétaires qui par leur efficacité allègent notre charge de travail considérablement : Yolande Vieceli, Odile Duval, Julie Urroz, Annie Nicolas, et Débora Thomas au Département Maths-Info (DMI), et Sylvie Armengaud-Meche (*aka* SAM), Sylvie Eichen et Audrey Cathala à l'IRIT de Toulouse. Je remercie également l'équipe pédagogique du DMI et tout particulièrement, Damien Sauveron, Pierre-François Bonnefoi, Karim Tamine, Abdelkader Necer et Frédéric Mora. Enseigner à vos côtés

fut une expérience incroyablement formatrice, j'espère conserver cette soif de diffuser les connaissances et cette envie de se renouveler de longues années. Je tiens également à remercier chaleureusement Réda Guédira d'avoir accepté de me confier des enseignements à l'Institut Universitaire de Technologie.

Au cours de cette thèse, j'ai eu l'opportunité aux travers des différentes journées Codage et Cryptographie, de summer/winter school ou de conférences de rencontrer des personnes très intéressantes. J'ai notamment apprécié les discutions enrichissantes avec Léo Ducas, Morgan Barbier, Alain Couvreur, Guillaume Quintin, Julien Devigne, Vincent Grosso, Adeline Langlois, Martin Albrecht, Andreas Hulsing, Thomas Pöpplemann. Thomas Prest le plaisir de discuter autour de plus d'un verre était partagé. Je souhaite également remercier quelques personnes d'Xlim, à commencer par Patricia et Sandrine, pour leur bonne humeur matinale systématique et leur sympathie à mon égard, Yoann Weber pour la patience dont il a souvent fait preuve en acceptant de partager le même bureau que moi, ainsi que les thésards actuels du DMI ou non : Johann Sence, Pierre Leclerc, Cécile Chassagne, Quentin Alamélou, Paul Germouty, Mathieu Jossent, Julien Gerhards, je vous souhaite des thèses prolifiques. Je remercie vivement Stéphane Tarrade, Nicolas Pavie, Richard Bézin, Evans Bohl, Julien Schrek, Slim Bettaieb, Maryna Kudinova et Romain Dauliat pour toutes ces pauses cafés / discussions / soirées pizzas+HIMYM, ainsi que pour leur amitié. En cette fin de parcours d'étudiant, j'ai également une pensée amicale et nostalgique envers Nadia Mourier, Nicolas Legaud, ainsi qu'Arnaud Dambra.

J'adresse mes remerciements à Frédéric Desforges, Émilie David et Rémi Joffre pour leur amitié de longue date, et d'être toujours à mes côtés malgré certaines périodes de silence radio ou aux abonnés absents. Frédo merci pour ces soirées piconades lors de mes rares venues à C-land, ta culture youtubesque n'a d'égale que Chuck Norris, définitivement. Je remercie également les LaLuLiLeLo's de m'avoir approuvé. Je tiens également à souligner la patience et le soutien de ma famille (Deneuville, Demery, Sevin, et Donzé) et ma belle-famille (Michèle Theillaud, Marie-Louise, Jean-Pierre et Francine Garabeuf), et à les en remercier.

Enfin, j'aimerais remercier du plus profond de mon cœur mes parents, qui m'ont toujours poussé à me dépasser, à donner le meilleur de moi-même. Bien que vous n'ayez pas toujours approuvé mes choix, vous m'avez toujours soutenu, et c'est grâce à vous que j'en suis là, Merci ! Je termine par toi Élodie, ma tendre épouse. Tu m'apportes du bonheur quotidiennement, tu me soutiens inconditionnellement, et tu arrives à m'apaiser quand je suis stressé, Merci pour tout ! Je vous dédie cette thèse à tous les trois.

Tables des Matières

Liste des Figures

Liste des Tables

Liste des Algorithmes

Première partie

Introduction et Contexte

Cette première partie a pour objectif de fournir au lecteur peu familier aux notions de sécurité informatique et plus largement de cryptologie les notions basiques pour la compréhension de ce mémoire. En aucun cas cette introduction ne saurait être exhaustive, et nous conseillons au lecteur en quête d'approfondissements les lectures suivantes :

— [Sin99] : Ouvrage de vulgarisation scientifique à propos de la science des secrets, avec une approche historique réaliste et détaillée,

— [MVOV96] : Recueil des techniques de sécurisation de l'information, relativement détaillées,

— [Knu98] : Œuvre sur les méthodes de programmation semi-numérique, utiles à l'implémentation des méthodes cryptographiques présentes dans ce mémoire,

— [TK03] : Synthèse des méthodes de sécurité, ou [Mus09] pour un recueil d'attaques pratiques sur les mesures de sécurité actuelles, hors du cadre de cette thèse.

Dans un premier chapitre, nous présentons un bref historique des idées et techniques ayant progressivement dessiné le paysage cryptographique. Ce chapitre introduit également les principaux problèmes sur lesquels repose la sécurité des cryptosystèmes asymétriques actuels, ainsi que les deux modèles de calcul connus : classique et quantique.

Une fois les principales notions abordées dans cette thèse introduites, nous présentons quelques schémas de chiffrement historiques, et expliquons en quoi l'existence d'un modèle de calcul alternatif remet en question la sécurité de ces schémas.

Les enjeux de proposer de nouveaux schémas (supposés) résistants à ce modèle de calcul mis en avant, nous présentons le contexte de cette thèse ainsi que les contributions apportées par ce présent travail.

Le second chapitre de cette partie – beaucoup plus technique – a pour but de fournir au lecteur le bagage mathématique nécessaire à la bonne compréhension de ce mémoire. Tout au long de ce manuscrit, nous supposons que les bases de l'algèbre linéaire, des statistiques et probabilités sont familières au lecteur, et recommandons l'ouvrage pédagogique [YW09] le cas échéant. Ce chapitre aborde donc notamment des notions essentielles à la cryptographie à base de réseaux euclidiens et de codes correcteurs d'erreurs, deux des outils mathématiques candidats à la cryptographie dite "post-quantique" les plus prometteurs.

Chapitre 1

Introduction

Sommaire

La *cryptologie*, du grec $\kappa\rho\upsilon\pi\tau o\zeta$: ce qui est caché ou couvert, et $\lambda o\gamma\iota\alpha$: la science, désigne donc la science des secrets. Cette science, indissociable des moyens de communication, a donc pour objectif principal la transmission d'informations (potentiellement sensibles) entre deux interlocuteurs (traditionnellement Alice et Bob [1]), de manière sûre (voir plus bas). On distingue principalement deux disciplines "duales" au sein de la cryptologie : la *cryptographie*, qui s'intéresse à la sécurisation de l'information, et la *cryptanalyse*, qui cherche à l'attaquer. Bien que ce mémoire traite principalement de cryptographie, il est indispensable de connaître certains aspects de la cryptanalyse afin de s'assurer de la robustesse des systèmes proposés.

Historiquement, le sens commun de la cryptographie s'est légèrement écarté de son sens étymologique. Le fait de dissimuler une information au sein d'un document est apparenté à de la *stéganographie*, alors que la cryptographie a pour but de rendre l'information incompréhensible à tous sauf au destinataire, via des techniques de chiffrement que nous détaillerons plus tard. Un des exemples de stéganographie les plus anciens remonte à l'Antiquité : afin de communiquer en toute sécurité, les rois et les généraux militaires utilisaient pour communiquer des esclaves dont ils rasaient la tête, y tatouaient leur message, avant de les envoyer au destinataire lorsque leurs cheveux avaient repoussé [Sin99]. Un exemple plus contemporain, et probablement plus connu, peut être trouvé dans les correspondances entre George Sand et Alfred de Musset [SdM04]. Par ailleurs la cryptographie continue d'évoluer, et bien qu'elle fut longtemps considérée d'un point de vue légal comme une arme de guerre, sa réglementation s'est adoucie à la fin du XX$^{\text{ème}}$ siècle, et la cryptographie a de nos jours plus vocation à apporter de la confiance dans les transactions numériques (voir la loi LCEN de 2005).

La cryptographie a donc pour but de sécuriser la transmission entre l'expéditeur et le destinataire. Sécuriser l'information s'entend à plusieurs niveaux :

— Confidentialité : Seul le destinataire Bob (et éventuellement l'expéditeur Alice) peut comprendre l'information transmise,

— Intégrité : L'information transmise n'a pas été altérée durant son trajet,

— Authentification : Chaque interlocuteur est bien celui qu'il prétend être,

— Non-Répudiation : L'expéditeur ne peut pas nier avoir envoyé l'information,

— Contrôle d'Accès : Seuls la/les personne(s) autorisée(s) peu(ven)t accéder à l'information.

La confidentialité a longtemps été obtenue grâce à la connaissance d'un secret commun aux interlocuteurs (cryptographie symétrique ou à clé secrète). L'inconvénient majeur de ce type de cryptographie réside en la nécessité d'une rencontre physique préalable entre les interlocuteurs afin que ceux-ci conviennent d'un secret qui leur permettra de communiquer de manière sûre. Les autres aspects de la sécurité ne sont apparus que bien plus tard avec l'avènement de la cryptographie à clé publique, qui permet de s'affranchir de la précédente contrainte. Ces deux types de cryptographie sont abordés plus en détail dans la suite de ce chapitre.

Enfin, avant de présenter un historique des cryptosystèmes, nous tenions à souligner que cryptographie et stéganographie peuvent être associées pour compliquer la tâche des

1. Bien qu'il existe des propositions plus exotiques [Par13].

cryptanalystes. Le meilleur exemple à notre connaissance est le logo de l'Agence Nationale de la Sécurité des Systèmes d'Information, publié en février 2012 [2].

1.1 Cryptographie Ancestrale

Dans cette section, nous présentons les principaux systèmes de chiffrement qui ont marqué l'histoire mais qui, de par leur simplicité relative ou du fait qu'ils aient été cryptanalysés, ne sont plus utilisés aujourd'hui. Cette présentation ne se veut pas exhaustive mais plutôt pédagogique car elle permet d'introduire de manière simple des notions qui ne le sont pas nécessairement. Afin de mieux comprendre en quoi un cryptosystème est robuste, il est nécessaire de comprendre les différents types d'attaques pouvant être menées, et quel(le) facteur/quantité de travail cela demanderait à un ou plusieurs ordinateurs récents.

Différents types d'attaques. En fonction des connaissances d'un adversaire et de si ce dernier a accès ou non à l'algorithme de chiffrement, on distingue différentes catégories d'attaques :

— Chiffré seul : l'attaquant est en possession d'un (ou plus) message chiffré, mais ignore le message clair lui correspondant,

— Clair connu : l'attaquant connaît un (ou plus) couple message chiffré, message clair correspondant,

— Clair choisi : l'attaquant a le droit de choisir des messages clairs et d'en obtenir les chiffrés correspondant.

Le dernier type d'attaque correspond à l'adversaire le plus puissant que l'on rencontre en cryptographie. On peut imaginer que cette personne a un accès physique au mécanisme de chiffrement, et son but est d'extraire la clé/le secret utilisé(e) pour passer d'un chiffré à un clair. C'est généralement par rapport à ce type d'adversaire que les paramètres des cryptosystèmes sont établis.

Facteur de travail. Cette notion est apparue avec l'avènement des ordinateurs personnels, et désigne la quantité de travail nécessaire à un ordinateur pour cryptanalyser un schéma cryptographique. Cette quantité de travail s'exprime généralement en opérations élémentaires – telles que des additions de bits – chaque opération élémentaire prenant un cycle d'horloge. Par exemple, 100000 ordinateurs à architecture 64 bits fonctionnant à 4 GHz pendant un an peuvent atteindre un facteur de travail de

$$100000 \times 64 \times 4 \cdot 10^9 \times 365 \times 24 \times 60 \times 60 \approx 2^{80} \tag{1.1}$$

opérations élémentaires. À l'heure actuelle (septembre 2016), la communauté cryptographique estime assez unanimement qu'un facteur de travail de 2^{100} (on dit aussi un niveau de sécurité de 100 bits) est hors de portée. C'est pourquoi, lorsque nous paramétrerons nos cryptosystèmes, nous veillerons à ce que la meilleure attaque connue requière au minimum un facteur de travail supérieur à 2^{100} opérations élémentaires.

2. Une version de ce logo est disponible ici : http://www.unilim.fr/pages_perso/deneuville/img/anssi.png. La solution a été publiée en janvier 2014, et est disponible ici : http://blog.bienaime.info/2015/01/le-challenge-du-logo-anssi.html.

Ces deux notions abordées, nous allons maintenant décrire les schémas de chiffrement “antiques” et dégager les idées importantes qui contribuent encore à l'élaboration de cryptosystèmes actuels. La plupart des informations historiques et images proviennent de [Sin99].

1.1.1 Scytale

C'est actuellement le plus ancien système de chiffrement connu. La scytale est un bâton de bois autour duquel on enroule une lanière de cuir, avant d'y écrire le message perpendiculairement à la façon dont on l'a enroulée. Une fois le message écrit, la lanière de cuir était déroulée et généralement portée en guise de ceinture par son coursier afin de dissimuler le message. Un schéma de scytale est présenté Fig. 1.1, on peut y lire “`SEND MORE TROOPS TO SOUTHERN FLANK AND`”.

Afin de déchiffrer correctement le message, le destinataire devait posséder une scytale de même diamètre que celle ayant servie pour le chiffrement du message, faute de quoi les lettres ne s'alignaient pas correctement et le message restait incompréhensible.

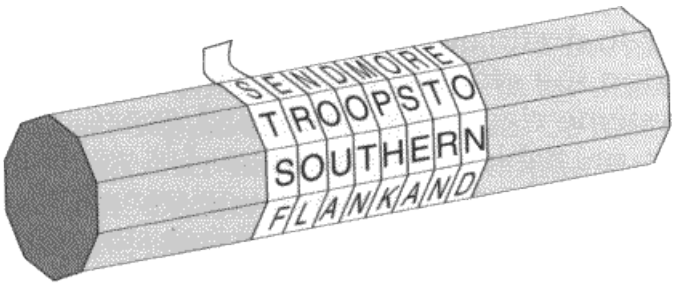


FIGURE 1.1 – Illustration d'une scytale ayant le diamètre adéquat pour déchiffrer correctement le message [Sin99].

On peut remarquer que dans les messages chiffré (la lanière de cuir déroulée) et clair partagent les mêmes lettres, seul leur ordre se trouve modifié. C'est ce qu'on appelle une transposition. Le nombre de transpositions possible croît très vite en fonction du nombre de lettres à permuter (plus qu'exponentiellement). Cependant, toutes les permutations ne sont pas possibles, il faut que le destinataire puisse reconstruire efficacement le message. Ainsi la transposition est un bon outil cryptographique, mais nécessite d'être employée conjointement à d'autres mécanismes cryptographiques pour rendre un schéma robuste, comme nous le verrons plus tard dans ce chapitre.

Le fait que seulement certaines transpositions soient possibles rend la cryptanalyse de ce schéma assez aisée, même sans ordinateur. Il “suffit” de retrouver le bon arrangement des lettres. Admettons que par quelque moyen, nous ayons intercepté une lanière de cuir ayant l'aspect suivant, mais que son transporteur refuse de nous dévoiler le diamètre de la scytale utilisée pour chiffrer ce message :

`N A S A O Q D M U U E I S E M N A R A U T O I I T N N T`

Il suffit de compter le nombre de lettres (ici 28), d'énumérer les diviseurs de ce nombre (2×14, 4×7, 7×4 et 14×2), d'écrire à la verticale dans une grille de ces dimensions, et de voir si le message lu horizontalement dans cette même grille a du sens ou non.

Sens d'écriture ↓ / Sens de lecture →

N	S	O	D	U	E	S	M	A	A	T	I	T	N
A	A	Q	M	U	I	E	N	R	U	O	I	N	T

Ici, le message obtenu est "NSODUESMAATITNAAQMUIENRUOINT", qui n'a aucun sens. On passe aux dimensions suivantes : 4×7.

Sens d'écriture ↓ / Sens de lecture →

N	O	U	S	A	T	T
A	Q	U	E	R	O	N
S	D	E	M	A	I	N
A	M	I	N	U	I	T

Le message obtenu est "NOUS ATTAQUERONS DEMAIN À MINUIT", la cryptanalyse est réussie.

Comme nous venons de le voir, l'inconvénient majeur de ce schéma est que chaque lettre du texte clair se retrouve dans le texte chiffré. L'idée de mélanger les lettres est quant à elle plutôt bonne et sera reprise bien plus tard dans d'autres schémas comme Enigma (voir Sec. 1.1.4), DES ou AES (voir Sec. 1.2.1). Cette constatation établie, nous allons maintenant décrire le chiffre de Cæsar, premier schéma à corriger ce problème.

1.1.2 Chiffre de Cæsar

Bien que l'origine de ce schéma de chiffrement ne soit pas connue de manière précise, il porte de nom de Jules Cæsar car ce dernier l'utilisait pour certaines de ses communications, à but militaire notamment. Il consiste dans sa version de base à remplacer chaque lettre de l'alphabet par celle qui se trouve trois positions plus loin, X, Y, et Z étant remplacées respectivement par A, B, et C.

Texte clair	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Texte chiffré	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

TABLE 1.1 – Correspondance entre les lettres du texte clair et celles du texte chiffré.

Ainsi, le message "Veni, vidi, vici" serait chiffré en :

Texte clair VENI VIDI VICI
Texte chiffré YHQL YLFL YLFL

Comme nous l'avons mentionné plus haut, le principal avantage de ce schéma est que les messages chiffrés et clairs ne comportent pas les mêmes lettres. Cependant, chaque lettre est systématiquement remplacée par la même lettre. Et c'est exactement ce principe qui sera utilisé pour la cryptanalyse de ce schéma.

D'autres versions de ce schéma d'apparence plus complexes et nommées plus généralement chiffrement par substitution (au lieu de décalage) furent développées. Elles consistent à appliquer une permutation plus complexe qu'un simple décalage. Cependant, l'effet sur le texte chiffré reste le même : une même lettre dans le texte clair sera chiffrée par une même lettre dans le texte chiffré.

Au IX$^{\text{ème}}$ siècle, Al-Kindi écrit le premier manuscrit traitant de cryptanalyse (retrouvé il y a seulement 30 ans à Istanbul). Il y montre notamment qu'étant donné une langue, la fréquence d'apparition de chaque lettre n'est pas la même. Ainsi dans la langue de Molière, le 'e' apparaît bien plus souvent que le 'w'. En se basant sur la loi des grands nombres et en analysant la fréquence d'apparition des lettres dans la Bible, écrite en français, on obtient le tableau de fréquence d'apparition suivant :

Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
Fréquence	9,42	1,02	2,64	3,39	15,87	0,95	1,04	0,77	8,41	0,89	0,00	5,34	3,24
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Fréquence	7,15	5,14	2,86	1,06	6,46	7,90	7,26	6,24	2,15	0,00	0,30	0,24	0,32

TABLE 1.2 – Table des fréquences d'apparition des lettres dans la Bible en langue française.

À partir de cette table, le raisonnement est le suivant. Nous connaissons la fréquence d'apparition des lettres dans le texte chiffré. Si nous faisons l'hypothèse que telle ou telle langue a été utilisée pour le texte clair, et sous réserve que celui-ci soit suffisamment représentatif de la langue supposée[3], alors comme chaque lettre est toujours chiffrée par la même lettre, la lettre apparaissant le plus dans le texte chiffré sera avec une bonne probabilité celle qui a la plus grande fréquence d'apparition dans la langue utilisée pour la rédaction du texte clair. Par essais successifs au maximum de vraisemblance, on parvient assez rapidement à retrouver la clé secrète, et donc à déchiffrer.

1.1.3 Chiffre de Vigenère

Remontant au XVI$^{\text{ème}}$ siècle, le chiffre de Vigenère est un schéma de chiffrement par substitution polyalphabétique, c'est-à-dire combinant plusieurs chiffrements par décalage (26 dans sa version originale). Il a donc pour particularité qu'une même lettre du texte clair peut-être chiffrée différemment, et ce en fonction d'une clé.

La clé est généralement un mot-clé, beaucoup plus court que le texte à chiffrer (et c'est ce qui fera par la suite sa faiblesse). Ce mot-clé est répété autant de fois que nécessaire pour atteindre la longueur du message à chiffrer. Dans sa version originale, chaque lettre est chiffrée en utilisant la Table 1.3. Ainsi le texte suivant chiffré avec la clé `SECRET` devient :

clair		UN TEXTE CHIFFRE AVEC VIGENERE
clé		SE CRETS ECRETSE CRET SECRETSE
chiffré	→	MR VVBMW GJZJYJI CMIV NMIVRXJI

Dans le chiffré, on voit qu'il y a des successions de lettres qui se répètent. Celles-ci peuvent correspondre à un enchaînement de lettres du texte clair, chiffrées avec le même enchaînement de lettres de la clé, ou pas (faux-positifs), c'est ce qu'on appelle le test de Kasiski. Dans le premier cas, on en déduit que la clé a été répétée une fois ou plus entre ces deux enchaînements, et il suffit alors de compter la longueur du décalage (ici, 12 entre les deux JI). La clé a donc une longueur divisant ce nombre (ici, 1, 2, 3, 4, 6, ou 12). Il suffit alors de faire une hypothèse sur la longueur de la clé, et de mener une analyse fréquentielle sur les sous-textes impliqués pour retrouver la clé et donc le message. À noter que plus le

3. L'écrivain Georges PEREC a réussi l'exploit d'écrire un livre (lipogramme) d'environ 300 pages en français sans utiliser la lettre 'e' ailleurs que sur la page de garde pour signer son œuvre [Per69].

Lettre de la clé \ Lettre du message clair	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

TABLE 1.3 – Correspondance entre les lettres du texte clair et celles du texte chiffré pour le chiffre de Vigenère. Ici, la lettre J chiffrée par la lettre C donne la lettre L. Pour déchiffrer, on commence par regarder la lettre de la clé utilisée, on se déplace jusqu'à la lettre chiffrée dans la même ligne, puis on obtient la lettre déchiffrée en remontant en haut de la colonne.

nombre de lettres contenues dans les enchaînements se répétant est élevé, moindre est le risque de faux positifs. Nous ne rentrerons pas plus dans les détails mais le calcul de l'indice de coïncidence permet de déterminer si un texte a été chiffré en utilisant un chiffrement mono- ou poly-alphabétique, ainsi que la langue utilisée le cas échéant.

Enfin, il est important de remarquer que cette cryptanalyse fonctionne lorsque le mot-clé a une longueur bien inférieure à celle du texte qui est chiffré. Dans le cas (peu pratique) où la clé est de même longueur que le texte clair, ce chiffrement est assimilé à un chiffrement de Vernam. Ce dernier est le seul schéma de chiffrement dit à "secret parfait" connu, c'est-à-dire qu'il n'est théoriquement pas cassable dès lors que la clé n'est utilisée qu'une seule fois, ce qui vaut également à ce schéma le nom de masque jetable ou one-time pad.

1.1.4 Enigma

Cette machine (Fig. 1.2) utilisée par les Allemands au cours de la Seconde Guerre Mondiale marque probablement le plus grand tournant de l'histoire en terme de mécanisation pour le

traitement de l'information. Les efforts considérables qui ont été faits tant pour son élaboration que sa cryptanalyse ont bouleversé le domaine de l'informatique et de la cryptologie. D'une conception très complexe, elle allie les différentes techniques vues précédemment, substitution et permutation notamment, de telle sorte qu'une même lettre peut être chiffrée en n'importe quelle autre en fonction de la configuration actuelle de la machine.

Cette machine est composée de plusieurs rotors (jusqu'à 5 dans ses versions les plus élaborées), qui fonctionnent comme les aiguilles d'une montre : lorsque le premier rotor a fait une révolution complète, il entraîne une rotation d'un cran du rotor suivant, et ainsi de suite. À ces rotors s'ajoute un tableau de permutation des lettres sur la devanture, ainsi qu'un miroir à l'arrière afin de renvoyer le signal à travers les rotors. Le tout crée un enchevêtrement de canaux au sein desquels l'information circule le temps d'une frappe de caractère, avant que la rotation d'un rotor ne vienne bouleverser cet arrangement. Enigma ne possède donc pas de clé au sens cryptographique du terme, il "suffit" de retrouver la configuration initiale de la machine pour déchiffrer un message.

FIGURE 1.2 – Une machine Enigma prête à l'emploi : rotors installés à leur position initiale, tableau câblé. L'opérateur tape au clavier le texte à chiffrer, le texte chiffré s'allume sur la réplique du clavier au dessus, le premier rotor tournant à chaque frappe, en entraînant éventuellement avec lui d'autres rotors.

Lorsque les sous-marins allemands partaient en mission pour plusieurs semaines, ils emportaient de conséquents livres dans lesquels figuraient les configurations initiales à adopter pour chiffrer leur message. La procédure était très rigoureuse, et tous les messages comportaient les mêmes entêtes : les initiales de la délégation, l'heure, la longueur du message, et bien d'autres... Le message commençait notamment par une répétition de la clé chiffrée. Cette clé chiffrée était constituée des 3 lettres correspondant aux positions respectives des rotors,

répétée 1 fois, et était chiffrée avec cette même configuration initiale. Ce furent ces 6 lettres présentes dans chaque message qui conduisirent à la cryptanalyse d'Enigma.

La Seconde Guerre Mondiale a probablement été remportée par les Alliés grâce à leurs prouesses cryptographiques. Entre Bletchley Park et différents services de renseignements, un jeune esprit du nom d'Alan Turing va révolutionner la science. Ce dernier parvient à créer ce qu'il appelle à l'époque une "bombe" logique (notamment due au brouhaha qu'elle provoque une fois mise en marche) capable d'analyser, de manière mécanique et beaucoup plus rapidement qu'à la main, différentes configurations possibles de la machine allemande [4]. Cette capacité à tester rapidement énormément de possibilités a permis de déchiffrer certains messages cruciaux par force brute (en retrouvant la clé correspondant à celle chiffrée), et de prendre l'avantage par effet de surprise. On estime que les prouesses cryptographiques réalisées à Bletchley Park ont permis de raccourcir la guerre de 2 ans. La création d'Alan Turing lui vaudra la paternité de l'informatique telle qu'on la connaît aujourd'hui.

À travers les siècles, les forces et les faiblesses de chaque cryptosystème ont marqué les esprits, car de nombreuses vies en ont généralement dépendu. Avec la rupture technologique Enigma se dessine une nouvelle ère, celle de l'avènement de l'ordinateur. Au lendemain de la victoire des Alliés, la course entre cryptographes et cryptanalystes est relancée. Cependant, la mécanisation du traitement de l'information ouvre la voie à des techniques cryptographiques encore plus évoluées.

1.2 Cryptographie Moderne

Si la conception ainsi que les attaques ou tentatives d'attaques contre les cryptosystèmes vus jusqu'à présent ont eu un impact positif sur notre compréhension des bonnes pratiques et stratégies pour protéger l'information, avec l'avènement de l'ordinateur, il n'est plus possible d'espérer sécuriser ses communications à l'aide de tels cryptosystèmes.

En effet, le plus simple des chiffrements par substitution mono-alphabétique serait rapidement cassé par une analyse de fréquence, et même le plus complexe d'entre eux, Enigma, ne résisterait pas à un ordinateur. Afin de cryptanalyser Enigma, il suffit de retrouver la position initiale. Il y a 3 rotors à choisir parmi 5, leur ordre a une importance, chaque rotor peut être dans 26 positions différentes, et 20 lettres sont permutées sur le tableau devant (6 restent inchangées). Au total, il y a donc

$$(5 \times 4 \times 3) \cdot \left(26^3\right) \cdot \left(\frac{26!}{6! \times 10! \times 2^{10}}\right) \approx 2^{67} \tag{1.2}$$

combinaisons possibles. Bien qu'un ordinateur récent puisse essayer par force brute toutes ces combinaisons en quelques semaines, les travaux supplémentaires d'Alan Turing ainsi que de ses collaborateurs ont permis de descendre le nombre de combinaisons à seulement $24^3 \times 60 \approx 2^{20}$, ce qu'un ordinateur récent pourrait faire en quelques secondes...

Ainsi, des cryptosystèmes plus complexes et robustes ont été mis au point après la guerre, permettant d'augmenter le facteur de travail nécessaire à un ordinateur pour la cryptanalyse. Cette section donne un aperçu historique des cryptosystèmes dits modernes.

4. L'histoire de la cryptanalyse d'Enigma prendrait un ouvrage à elle seule. Outre [Sin99], nous conseillons au lecteur le film "Imitation Game" de Morten Tyldum relatant la vie d'Alan Turing.

1.2.1 Cryptosystèmes Symétriques

Les cryptosystèmes abordés en Section 1.1 ont comme point commun que l'expéditeur et le destinataire doivent partager un même secret, connaître un algorithme de déchiffrement particulier ou la configuration initiale de la machine ayant servie à chiffrer un message. L'ensemble de ces cryptosystèmes est regroupé dans la famille de la cryptographie symétrique. Bien que ce mémoire aborde peu les aspects du chiffrement symétrique, nous en présentons les grandes lignes dans cette section.

Au sein des cryptosystèmes symétriques, on distingue principalement deux catégories :

— les chiffrements par blocs : chaque message à chiffrer est décomposé en blocs de taille fixe (quitte à compléter les blancs), et le chiffrement agit sur l'ensemble du bloc à l'aide de permutations (transpositions) et de substitutions.

— les chiffrements à flots de données : ces chiffrements sont généralement les plus rapides, et appropriés lorsque le débit de caractères à chiffrer n'est pas constant. Ces chiffrement agissent généralement bit par bit ou octet par octet.

Les chiffrements par blocs fonctionnent essentiellement de la même façon : le message est encodé en binaire, puis découpé en blocs de taille fixe, éventuellement "paddé" avec des 0. Chaque bloc est transformé par Xor [5] avec tout ou partie de la clé, transformée ou non, avant que ces données ne soient permutées. En termes de théorie de l'information au sens de Shannon, le fait d'ajouter la clé aux données permet d'assurer de la confusion (une lettre est transformée en une autre), et les permutations assurent la diffusion des modifications (les lettres se retrouvent mélangées). Ces permutations et substitutions sont répétées sur plusieurs tours de sorte à créer un effet d'avalanche : la moindre modification sur les données en entrée entraîne une sortie bien différente.

Parmi les chiffrement par blocs, les schémas de Feistel sont probablement les plus connus. Proposés en 1973, ils ont comme propriété singulière qu'ils sont inversibles même si la fonction qu'ils utilisent ne l'est pas. Pour cette raison ils furent largement acceptés par la communauté et furent même standardisés en 1977 comme standard de chiffrement symétrique dans l'algorithme Data Encryption Standard (DES).

Bien qu'aucune réelle faille n'ait été trouvée sur le DES, celui-ci utilise des clés de 64 bits, dont seulement 56 apportent de la sécurité (il y a un bit de contrôle par octet), ce qui ne résisterait pas plus de quelques heures à une machine récente. Afin d'en relever simplement sa sécurité, le Triple-DES fut adopté en 1998.

Rapidement après cette standardisation, le National Institute of Standards and Technology (NIST) lance le concours AES (pour Advanced Encryption Standard), un appel à la communauté cryptographique afin de concevoir le prochain standard de chiffrement par blocs. Les vainqueurs de ce concours seront Joan Daemen et Vincent Rijmen, et leur proposition Rijndael sera rebaptisée AES lors de sa standardisation en 2001. Contrairement à son prédécesseur, Rijndael n'utilise pas de réseau de Feistel, mais un état interne dans lequel est stocké l'information à chiffrer. Le principe de réseau de substitutions-permutations sur plusieurs tours reste intègre. L'AES reste le standard aujourd'hui, intégré dans les processeurs, les

5. Un Xor est un OU logique exclusif, c'est-à-dire qu'il vaut un si et seulement si un seul des deux bits est à un.

cartes bancaires, et même certains téléphones. Avec des clés de 128 à 256 bits, cet algorithme pourrait résister encore bien des années, à moins qu'une faille de conception ne soit découverte, ou que les prouesses technologiques ne compromettent sa sécurité. Nous aborderons ce second aspect en Sec. 1.3.4.

1.2.2 Un Nouveau Paradigme

En 1976, Whitfield Diffie et Martin Hellman suggèrent une nouvelle approche en cryptographie : l'utilisation de deux clés distinctes, une pour chiffrer qui serait publique, l'autre pour déchiffrer qui resterait privée [DH76]. Ils proposent également dans cet article une manière sûre de convenir d'un secret en utilisant un canal de communication qui ne l'est pas forcément. Ce schéma d'échange de clé sera décrit en Sec. 1.3.2 lorsque nous aurons introduit le problème du logarithme discret.

Cependant, le problème de trouver un schéma de chiffrement asymétrique reste entier : le secret échangé par leur méthode ne permet que de chiffrer des données en symétrique, et le problème d'une clé par destinataire demeure.

1.2.3 Cryptosystèmes Asymétriques

En 1977, Ronald Rivest, Adi Shamir, et Léonard Adleman proposent le premier schéma de chiffrement asymétrique [RSA78], nommé par leurs initiales RSA. Celui-ci est basé sur la théorie des nombres et utilise notamment le théorème de Fermat afin de disposer de deux clés distinctes inverses l'une de l'autre. Nous présentons ici sa version initiale par simplicité.

Alice qui souhaite permettre à ses correspondants de lui écrire de manière sûre, choisit deux grands nombres premiers p et q, et calcule $N = pq$. Elle choisit un entier e premier avec $\phi(N) = (p-1)(q-1)$ et détermine (par l'algorithme d'Euclide étendu) l'entier d tel que $ed = 1 \mod \phi(N)$. La clé publique d'Alice est le couple (N, e), tandis que sa clé privée est d.

Bob, qui souhaite envoyer un message à Alice sous la forme d'un entier $m < N$ calcule $c = m^e \mod N$ et l'envoie à Alice. Afin de déchiffrer le message, Alice utilise sa clé privée et calcule :

$$c^d = (m^e)^d = m^{ed} = m^1 = m \mod N, \tag{1.3}$$

l'avant dernière égalité résultant du théorème d'Euler (qui généralise le petit théorème de Fermat).

Ainsi, n'importe qui ayant connaissance de la clé publique d'Alice peut lui envoyer des messages chiffrés que, a priori, elle seule pourra déchiffrer. À ce jour, [DH76] et [RSA78] sont unanimement reconnus comme des contributions majeures en cryptographie, et comptent parmi les papiers les plus cités de ce domaine (environ 15000 et 17000 respectivement).

Par ailleurs l'usage de deux clés, une privée et une publique, a également ouvert la voie aux signatures électronique (entre autre). Cette fois-ci Alice peut utiliser sa clé privée pour signer un document numérique, de sorte que quiconque en possession de la clé publique pourra vérifier que le document provient bien de cette dernière. RSA permet en outre de signer un message, en inversant le rôle des deux clés. Par la suite, les schémas de signature que nous verrons ne permettent pas nécessairement de chiffrer des données c'est pourquoi, même si

c'est le cas pour RSA, il serait abusif de dire que la signature correspond à un chiffrement utilisant la clé secrète.

1.2.4 Cryptographie Hybride

Les principaux avantages des schémas symétriques sont leur efficacité, leur rapidité, ainsi que leur petite taille de clé. Cependant, ils requièrent tous la connaissance préalable d'un secret commun entre l'expéditeur et le destinataire, qui impliquerait l'obligation que les deux protagonistes se rencontrent physiquement. De plus, il est nécessaire de disposer d'une clé secrète pour chaque personne avec qui on souhaite communiquer.

Bien que la cryptographie à clé publique permette de se substituer à cette rencontre physique et aux multiples clés, ses performances à la fois en termes de clés et de coût des opérations ne la rendent pas compétitive. Il n'est pas envisageable de devoir attendre cinq minutes à la caisse d'un supermarché afin que la transaction bancaire se termine.

Afin de bénéficier des avantages de chaque type de cryptographie en en limitant les inconvénients, il est fréquent de recourir à la cryptographie asymétrique afin de convenir d'un secret avec son interlocuteur, puis une fois celui-ci fixé, d'échanger des messages chiffrés à l'aide de schémas symétriques utilisant ce secret.

1.3 Problèmes et Modèle de Calcul

Dans cette section, nous abordons les différentes notions de complexité nécessaires pour juger de la robustesse d'un cryptosystème, ainsi que le principaux problèmes sous-jacents à la cryptographie asymétrique moderne. Nous abordons également l'architecture de calcul quantique, et discutons des implications sur cette même cryptographie.

1.3.1 Notions de Complexité

Traditionnellement, les algorithmes sont conçus pour effectuer une tâche bien précise, résoudre un problème de manière automatique. Les notions élémentaires de l'algorithmique ont été introduites et formalisées par Turing, dans le modèle de la machine qui porte son nom. Une machine de Turing est un automate constitué d'un ruban infini faisant office de mémoire, d'une tête de lecture/écriture se déplaçant sur ce ruban, ainsi qu'une table de transition d'état régissant les opérations à effectuer en fonction de ce qui est lu ou écrit sur le ruban. Les ordinateurs actuels, bien que substantiellement plus complexes, appartiennent à cet architecture de calcul. Tout l'enjeu de la théorie de la complexité consiste à évaluer le nombre d'opérations nécessaires à une telle machine afin de calculer une fonction, ou de résoudre un problème.

En fonction de la mémoire requise, du temps requis, du déterminisme ou non du problème et d'encore bien d'autres critères, il existe toute une hiérarchie de problèmes (voir Fig. 1.3). Parmi ces classes, deux d'entre elles sont d'un intérêt particulier pour la cryptographie. La première est la classe P, des problèmes décisionnels pour lesquels il existe une solution déterministe nécessitant un temps polynomial en la taille des entrées. Cette classe contient entre autres les problèmes d'optimisation linéaire, de connexité de graphes, et d'autres moins triviaux comme la primalité d'un nombre [AKS04]. La seconde est la classe NP (pour non-déterministe polynomial), composée des problèmes décidables par une machine de Turing non-déterministe en temps polynomial en la taille des entrées. Cette classe contient donc des

problèmes pour lesquels nous n'avons pas d'algorithme efficace les résolvant, mais dont toute solution à ce problème est vérifiable en temps polynomial. Les problèmes décidables en temps polynomial étant donc vérifiables en temps polynomial, nous avons trivialement $\mathsf{P} \subseteq \mathsf{NP}$.

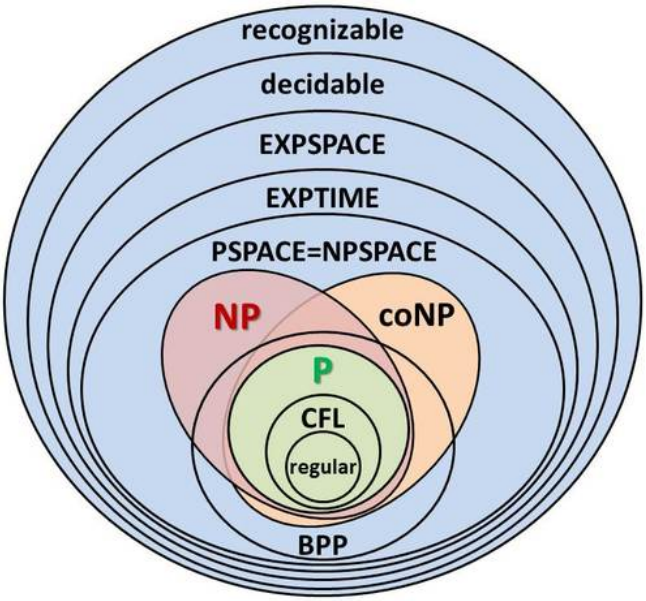


FIGURE 1.3 – Hiérarchie des différentes classes de complexité.

La plupart des cryptosystèmes à clé publique sont basés sur des problèmes à trappes, c'est-à-dire que les problèmes sur lesquels ils reposent sont a priori difficiles, mais la connaissance d'une information supplémentaire les rend plus aisés pour la personne en possession de cette information. En termes de complexité, cela revient à résoudre un problème a priori appartenant à NP, mais dont la connaissance d'une trappe permet de résoudre ce problème en temps polynomial.

La cryptographie nécessiterait idéalement des réductions à des problèmes de la classe NP, mais le fait de ne pas pouvoir résoudre ces problèmes en temps polynomial en fait également un obstacle, ne serait-ce que pour la génération de clés.

Ainsi, il est fréquent d'utiliser en cryptographie des problèmes dont certaines instances sont connues pour leur appartenance à NP, alors que d'autres appartiennent à P.

Un des plus grands problèmes ouverts en théorie de la complexité est de savoir si $\mathsf{P} = \mathsf{NP}$. Ce problème figure par ailleurs sur la liste des 7 problèmes du millénaire de l'institut Clay, et aurait de lourdes implications en cryptographie dans le cas où cette égalité tiendrait. En l'état actuel des connaissances, la communauté semble s'orienter vers une inclusion stricte.

1.3.2 Factorisation et Problème RSA

Trouver un algorithme permettant de factoriser n'importe quel nombre en temps polynomial en la taille de ce nombre est un problème de longue haleine. Même le problème de trouver un algorithme polynomial permettant de déterminer si un nombre est premier ou non a résisté longtemps à la communauté [AKS04].

Le problème RSA, qui consiste à extraire une racine $e^{\text{ème}}$ modulo un nombre composé, correspond à inverser la fonction de chiffrement sans la trappe. Nous savons que ce problème

se réduit à celui de la factorisation, dans le sens où un algorithme résolvant la factorisation peut être simplement dérivé en un algorithme résolvant le problème RSA. En effet, il suffit de factoriser $N = p \cdot q$, de calculer $\phi(N) = (p-1)(q-1)$ et d'inverser $e \mod \phi(N)$ pour retrouver la clé secrète d. Cependant, on ne sait pas si la factorisation peut se réduire au problème RSA, auquel cas nous aurions une équivalence entre ces deux problèmes, ou non. Toutefois, la factorisation est à l'heure actuelle la meilleure façon (en termes de complexité) de s'attaquer à RSA.

À l'heure actuelle, l'algorithme le plus efficace pour factoriser un grand nombre (de $\lfloor \log_2 n \rfloor + 1$ bits) est le crible algébrique [Pol93]. Il a une complexité en $L_n\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$, où la fonction de complexité L est définie comme suit :

$$L_n\left[\alpha, c\right] = \exp^{(c+o(1))(\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}}, \text{ pour } 0 < \alpha < 1. \tag{1.4}$$

Le coefficient α détermine à quel point la fonction L est exponentielle : pour $\alpha = 0$ nous avons $L_n[0, c] = (\ln n)^{c+o(1)}$, qui est une fonction polynomiale, et pour $\alpha = 1$, $L_n[1, c] = (n)^{c+o(1)}$, qui est une fonction exponentielle. Plus α est proche de 0, moindre est la complexité.

Le terme en $o(1)$ étant difficile à approximer, il est généralement nécessaire de le déterminer expérimentalement. Cette fonction appliquée à des modules de 768 bits nous donne $L \approx 2^{76.5}$, soit environ 76.5 bits de sécurité. Or le record de factorisation sur ces mêmes modules a nécessité environ 2000 ans de temps CPU tournant à 3GHz, ce qui donne une complexité de $\log_2(2000 \times 365.25 \times 24 \times 60 \times 60 \times 3 \cdot 10^9) \approx 67.4$ bits. On en déduit que le facteur $o(1)$ introduit une diminution de la fonction L, d'un facteur $k = \frac{67.4}{76.5} \approx 0.88$. Ainsi, les clés de 1024 bits que l'on voit régulièrement circuler dans les certificats des sites réputés de confiance n'offrent qu'une sécurité de $k \cdot L_{2^{1024}}\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right] \approx 76.4$ bits. L'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) a récemment recommandé de n'utiliser que des clés de 2048 bits ($\approx$ 103 bits de sécurité) pour des utilisations jusqu'en 2030, ou de 3072 bits ($\approx$ 122 bits de sécurité) au delà[6]. Les administrations françaises sont tenues d'appliquer ces recommandations depuis le 1er juillet 2016.

1.3.3 Logarithme Discret et Problème de Diffie-Hellman

Le logarithme discret consiste à inverser la fonction d'exponentiation dans un groupe. Ce groupe est historiquement $(\mathbb{F}_p^*, \times)$: le groupe multiplicatif des éléments inversibles dans le corps fini à p éléments, avec p premier, mais ce problème possède d'autres versions sur des groupes plus exotiques tels que les points d'une courbe elliptique.

Étant donné p premier et $g \in \mathbb{F}_p^*$ un générateur de $\mathbb{F}_p^*$, le protocole d'échange de clés de Diffie-Hellman permet à Alice et Bob de convenir d'un secret commun de la sorte : Alice choisit $a < p-1$ aléatoirement et envoie $g^a \mod p$ à Bob. De même, Bob génère $b < p-1$ aléatoirement et envoie $g^b \mod p$ à Alice. Alice et Bob peuvent tous les deux calculer simplement $(g^a)^b = \left(g^b\right)^a = g^{ab}$, alors qu'Eve qui écoute leur communication ne connaît que g^a et $g^b \mod p$.

Le problème du logarithme discret demande, à partir de g, p, et $g^x \mod p$ de retrouver x. Le problème de Diffie-Hellman demande quant à lui de retrouver $g^a b \mod p$ à partir de p, g, g^a et $g^b \mod p$. Si ce dernier se réduit trivialement au premier, il n'existe aujourd'hui aucune réduction du logarithme discret au problème de Diffie-Hellman.

6. L'annexe B1 du Référentiel Général de Sécurité V2.0 est disponible ici.

Cependant, la meilleure approche pour résoudre le problème Diffie-Hellman est en l'état des connaissances de résoudre le logarithme discret. Il existe beaucoup d'algorithmes permettant d'effectuer cette tâche, et il n'y a pas de meilleur algorithme de manière générique. Chaque algorithme possède une complexité à la fois en temps et en mémoire qui dépend fortement du groupe dans lequel le logarithme discret doit être résolu. Bien que le crible algébrique puisse également être utilisé pour résoudre le logarithme discret, d'autres algorithmes s'avèrent plus efficaces en pratique. Cependant ces algorithmes ont une complexité en $L_n\left[\frac{1}{2}, c\right]$ plutôt que $\frac{1}{3}$, ce qui les rend légèrement plus exponentiels que le crible algébrique. Cela a pour conséquence directe que lorsque les besoins en sécurité augmentent, les tailles de clés pour la cryptographie basée sur le logarithme discret augmentent moins vite que pour RSA.

Néanmoins, l'ANSSI recommande également l'usage de nombres premiers de taille 2048 bits pour un usage jusqu'en 2030, puis 3072 bits après , ou respectivement 200 bits et 256 bits dans le cas où le cryptosystème est instancié sur courbe elliptique.

Factorisation et logarithme discret partagent des points communs : les progrès réalisés pour résoudre un de ces problèmes ne tardent généralement pas à être dérivés pour résoudre l'autre problème. Outre le fait que ces deux problèmes soient relativement récurent dans la cryptographie moderne, ils partagent un point commun beaucoup moins désirable que nous détaillerons en Sec. 1.3.4, il existe des algorithmes polynomiaux permettant de résoudre ces problèmes dans le modèle de calcul quantique.

1.3.4 L'Ordinateur Quantique et ses Algorithmes

Cette section n'a pas pour but de se perdre dans des considérations de mécanique quantique avancée, mais plutôt de présenter comment deux de ces phénomènes physiques que nous vulgariserons permettent à un calculateur exploitant cette architecture de réaliser des exploits comparativement au modèle de Turing.

Pour appréhender le premier de ces phénomènes, il faut comprendre que l'information manipulée n'est pas la même. L'informatique classique utilise des bits d'information, stockés dans des transistors. Ces bits peuvent être dans un état soit 0, soit 1, mais pas les deux en même temps, tandis que l'informatique quantique manipule des qubits, pouvant être à 0, 1, ou n'importe quelle superposition de ces deux états (selon certaines probabilités). Ce principe s'appelle la *superposition* des états, et est grandement utilisé dans les algorithmes de Shor [Sho97] et Grover [Gro96].

Le second de ces principes est l'intrication quantique. Il consiste à appréhender un système dans sa globalité et non dans la singularité des éléments qui le composent. C'est-à-dire que les perturbations subies par un élément peuvent avoir des répercussions immédiates sur un autre élément même distant.

Conjointement, la superposition des états et l'intrication quantique permettent à un vecteur de qubits intriqués de prendre simultanément tous les états qu'un vecteur de bits classique pourrait prendre. Ainsi, un ordinateur quantique disposant de n qubits manipulera un vecteur de n qubits intriqués et se situant dans une superposition des 2^n états classiques possibles. À la fin d'un algorithme quantique (une succession de portes logiques dites "quantiques", adaptées aux qubits), le vecteur de qubits est mesuré, le figeant ainsi dans un et un seul des 2^n états possibles. Cette dernière étape représente à l'heure actuel un goulot d'étranglement

pour l'avènement du calcul quantique : chacun des états possibles dépend d'une probabilité, il est nécessaire d'augmenter de manière exponentielle la probabilité de l'état solution *avant* d'effectuer la mesure du vecteur de qubits, procédure encore mal maîtrisée aujourd'hui.

L'algorithme de Shor [Sho97] permet de résoudre la factorisation en temps $\mathcal{O}\left((\log n)^2 (\log\log n) (\log\log\log n)\right)$ plus un temps polynomial en la taille des entrées $(\log n)$ de post-traitement pour retrouver les facteurs de N à partir de la sortie de l'ordinateur quantique, au lieu de $L_n\left[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right]$ en classique. L'algorithme de Grover [Gro96] permet de trouver un élément au sein d'une base de n données en temps $\mathcal{O}(\sqrt{n})$ (et en espace $\mathcal{O}(\log n)$) à l'aide d'un ordinateur quantique, contre $\mathcal{O}(n)$ en classique. Ces algorithmes ne rentrant pas directement dans le cadre de cette thèse, nous laissons le lecteur curieux le soin de consulter les détails dans les articles originaux.

L'état actuel des connaissances ne permet pas de construire des ordinateurs quantiques capables de résoudre la factorisation ni le logarithme discret pour les paramètres recommandés par l'ANSSI. Les deux principaux obstacles dont il faudra s'affranchir pour ce faire sont le nombre de qubits d'un ordinateur quantique d'une part, ainsi que le problème d'évaluation de la solution (les qubits ne peuvent être lus aussi simplement que des bits) d'autre part. Cependant, le fait que des algorithmes quantiques polynomiaux existent pour les cryptosystèmes utilisés aujourd'hui amplifie l'urgence de concevoir et éprouver de nouveaux cryptosystèmes résistants à ce type de calculateur.

1.4 Vers l'Ère du Post-Quantique

Il existe aujourd'hui plusieurs problèmes, reposant sur différents outils mathématiques, pour lesquels aucun algorithme efficace n'est connu, ni classique, ni quantique. Pour cette raison, ces outils mathématiques sont de plus en plus étudiés et les cryptosystèmes basés sur les problèmes qui en découlent fleurissent. Cette section a pour objectif de présenter deux des outils mathématiques candidats à la cryptographie post-quantique, sur lesquels nous nous sommes focalisés au cours de cette thèse : les réseaux euclidiens et les codes correcteurs d'erreurs. Nous présenterons également les autres candidats et introduirons les travaux qui font l'objet de cette thèse.

1.4.1 Réseaux Euclidiens

Les réseaux euclidiens sont des ensembles périodiques de points de l'espace. Ces structures permettent entre autre de concevoir des problèmes d'algèbre linéaire pour lesquels, à notre connaissance, aucun algorithme quantique ne permet d'améliorer les algorithmes classiques existants. Les détails relatifs aux réseaux euclidiens seront abordés en section 2.2, et présents dans toute cette thèse.

Bien qu'étudiés depuis le XVIII$^{\text{ème}}$ siècle, les réseaux ont reçu beaucoup d'attention du point de vue cryptographique dû aux travaux d'Ajtaï [Ajt96], montrant que certaines instances (cas moyen) de problèmes peu connus pouvaient se réduire aux instances les plus difficiles à résoudre (pire cas) de problèmes bien connus. S'en sont suivies de nombreuses recherches sur les réductions pire-cas cas-moyen en réseaux et dans d'autres domaines cryptographiques.

Les réseaux ont également suscité beaucoup d'intérêt pour leur côté novateur. En effet, la conjecture [RAD78] concernant le chiffrement complètement homomorphe fut résolue en 2009 par la thèse de Gentry présentant le premier schéma de chiffrement capable d'évaluer un

nombre arbitraire d'opérations sur des données chiffrées. Nous avons proposé une application pratique du chiffrement complètement homomorphe, celle-ci sera détaillée au chapitre 5.

Sur un aspect plus pragmatique, la cryptographie à base de réseaux euclidiens ne nécessitant que des opérations sur des vecteurs modulo un petit entier (par rapport à RSA notamment), leur implémentation semble en outre plus aisée et même plus efficace à sécurité constante. Cela en fait un premier candidat post-quantique idéal.

1.4.2 Codes Correcteurs d'Erreurs

Précurseur de la cryptographie à base de réseaux, la théorie des codes correcteurs d'erreurs se nourrit des mêmes problèmes, avec une métrique différente : celle de Hamming la plupart du temps, ainsi que la métrique Rang, encore peu connue mais déjà prometteuse.

En effet, comme nous le verrons en section 2.3, la théorie des codes offre des problèmes NP-difficiles, et les principaux problèmes sous-jacents aux cryptosystèmes actuels sont bien étudiés et compris.

Dans ce cadre, nous avons proposé un cryptosystème ne nécessitant plus de masquage statistique de la structure employée, permettant ainsi de se ramener à un problème générique basé sur cette même théorie.

1.4.3 Autres Candidats

Bien que moins populaires et/ou moins étudiées, il existe d'autres alternatives à la cryptographie post-quantique. Nous resterons bref sur ces alternatives, en dehors du cadre de cette thèse.

Cryptographie basée sur les fonctions de hachage

La principale propriété d'une fonction de hachage est d'être à sens unique : c'est-à-dire qu'il est simple de trouver l'image de n'importe quel élément, mais calculatoirement infaisable de trouver un antécédent par cette même fonction étant donné une image [NY89].

Les fonctions de hachage sont utilisées quotidiennement, notamment à des fins d'authentification, mais leur inconvénient majeur est dû au modèle de sécurité sous-jacent, à savoir celui de l'oracle aléatoire (Random Oracle Model (ROM)). Dans ce modèle, les fonctions à sens unique sont vues comme des fonctions complètement aléatoires or, il est impossible de simuler un phénomène purement aléatoire. Le ROM est de plus en plus critiqué et la majorité des primitives proposées tendent à se passer de ce modèle.

Cependant, les fonctions de hachage permettent de construire des primitives cryptographiques pour lesquelles aucun algorithme de cryptanalyse (classique ou quantique) efficace n'existe, et ces fonctions ainsi que leurs propriétés sont étudiées depuis maintenant des décennies, ce qui en fait un candidat pour la cryptographie post-quantique crédible.

Cryptographie basée sur les équations multivariées

Trouver une même racine commune à un ensemble de polynômes non-linéaires en plusieurs variables est un problème prouvé NP-complet (c'est-à-dire au moins aussi dur que tous les autres problèmes de la classe NP) [Pat96]. Basées sur ce problème, plusieurs primitives cryptographiques ont été proposées.

Bien qu'il existe des schémas de signatures efficaces basés sur le multivarié, ceux-ci souffrent de grosses tailles de clés, et les outils de cryptanalyse (utilisant des bases de Gröbner) ont

généralement une complexité difficile à évaluer de manière rigoureuse (rendant complexe la détermination de paramètres pratiques).

1.5 Contexte de la Thèse

Dans un besoin imminent de s'intéresser à des primitives a priori résistantes à un potentiel ordinateur quantique, nous avons travaillé à l'élaboration de différents schémas cryptographiques, faisant l'objet des prochains chapitres.

Réparation de NTRUSign

Le premier de ces chapitres concerne un schéma de signature basé sur les réseaux, paru en 2001 [HPS01]. Le schéma initial proposé par les auteurs exploite la connaissance d'une base de petite norme du réseau comme trappe afin de retrouver aisément des antécédents. Bien que particulièrement efficace notamment grâce à la structure du réseau employé, ce schéma de signature laisse fuir de l'information qui permet, après quelques centaines de signatures, de retrouver la base secrète ayant été utilisée. Des contre-mesures heuristiques ont été proposées [HHGP+03, HWH08], mais rapidement cassées [DN12]. Récemment, une réparation plus théorique que pratique a été proposée, mais avec inconvénient majeur de rendre inutilisable le schéma initial. Nous proposons dans le chapitre 2 une réparation pratique de NTRUSign. Cette réparation utilise des techniques issues d'un des schémas de signature basée sur les réseaux les plus efficaces [Lyu12, DDLL13]. Nous proposons de plus des paramètres pour l'implémentation de notre schéma.

Signature de Groupe sur les Réseaux

Une des primitives cryptographiques indispensable à la protection de la vie privée est la signature de groupe. Une telle primitive permet à un utilisateur membre d'une communauté de signer au nom de celle-ci sans pour autant divulguer sa propre identité. Ce type de primitive a notamment été développé en utilisant de la cryptographie basée sur les courbes elliptiques (et leur opération de couplement sous-jacent), mais peu de schémas existent à base de réseaux. Les principaux schémas ont eu pour objectif de réduire la taille des signatures, celle des clés, puis d'apporter de nouvelles fonctionnalités [LLLS13, LLNW14, LNW15, NZZ15]. Si le côté dynamique de la signature de groupe (le fait que la communauté puisse évoluer) est relativement récent [LLM+16], une des propriétés essentielles à la protection de la vie privée manque encore : la *non-frameability*. Cette propriété assure que le responsable de la communauté ne peut, même s'il devient malicieux, abuser les membres de cette communauté. Nous avons proposé un schéma de signature traçable (extension d'une signature de groupe) basé sur le schéma de signature du Chapitre 2, permettant à la fois une gestion dynamique de la communauté et une garantie que le responsable de la communauté ne pourra abuser de ses membres.

Délégation Sécurisée de Signatures ECDSA

Comme mentionné dans la Sec. 1.4.1, le chiffrement homomorphe est un des aspects qui font de la cryptographie basée sur les réseaux un thème de recherche à la fois attractif et dynamique. Cependant, malgré les récentes avancées sur ce domaine, les cas d'applications pratiques du chiffrement complètement homomorphe sont plutôt restreints. À travers des

techniques de délégation de calculs et de calculs multi-parties, nous proposons un cas d'usage pratique : celui de la délégation sécurisée de signatures ECDSA. Les calculs usuellement coûteux sont déportés dans le cloud de manière sécurisée de sorte à pouvoir, dans le cas d'un usage par des Hardware Security Module (HSM), apporter de l'efficacité, un aspect de réponse à la demande, ainsi qu'une capacité économique de mise à l'échelle, le tout nourri de tests de performances concrets.

Chiffrement efficace basé sur les codes, sans masquage de structure

Tous les schémas de chiffrement basés sur les codes reposent intrinsèquement sur la qualité du masquage de la structure du code correcteur employé. Si ce paradigme semble tenir pour certains cryptosystèmes (McEliece-Goppa, MDPC), il a longtemps et souvent été source de critiques. Par ailleurs, des attaques exploitant les erreurs de déchiffrement ont récemment fait leur apparition [GJS16]. Le schéma de chiffrement que nous proposons dans le chapitre 6 a pour avantages majeurs de ne pas reposer sur le masquage de la structure du code employé, ainsi que de proposer une analyse de l'erreur de déchiffrement de sorte à contrevenir aux attaques proposées précédemment. De plus, cette approche est générique et peut être instanciée dans différentes métriques. Nous donnons des paramètres à la fois pour la métrique de Hamming et la métrique Rang, de plus ces paramètres sont compétitifs avec les meilleurs actuels (MDPC) et même meilleurs asymptotiquement.

1.6 Organisation du manuscrit

Après ce chapitre d'introduction, nous introduisons dans le chapitre 2 les principales notations, nous définissons les primitives cryptographiques formellement, et nous présentons deux des outils utiles à la cryptographie post-quantique à savoir les réseaux euclidiens et les codes correcteurs d'erreurs. Les chapitres 3, 4 et 5 concernent des schémas basés sur les réseaux que nous avons proposés : le chapitre 3 est une réparation d'un des schémas de signature les plus efficaces sur les réseaux, le chapitre 4 présente une signature traçable basée sur la réparation que nous viendrons de présenter, et le chapitre 5 expose un protocole de délégation de signature utilisant du chiffrement complètement homomorphe. Enfin, le chapitre 6 décrit un nouveau paradigme pour le chiffrement basé sur les codes, instancié dans deux métriques. Nous conclurons ce manuscrit en chapitre 7 en présentant également quelques perspectives de recherche.

Chapitre 2

Préliminaires

Sommaire

Dans ce chapitre, nous fournissons la plupart des prérequis nécessaires à la compréhension du reste de ce mémoire. Comme mentionné dans l'introduction de cette partie, nous supposons toutefois que le lecteur est familier avec les notions élémentaires d'algèbre linéaire, de probabilités, et de statistiques, et référons ce dernier vers [YW09] le cas échéant.

2.1 Contexte Mathématique et Cryptographique

2.1.1 Notations

Ensembles. Dans le reste de ce mémoire, nous désignons par $\mathbb{N}$ l'ensemble des entiers naturels, $\mathbb{Z}$ l'ensemble des entiers relatifs, et pour $q \in \mathbb{N}$, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ désigne l'anneau quotient des entiers réduits modulo q. Nous désignerons par $\mathbb{K}$ un corps et par $\mathbb{F}_q$ le corps fini à q éléments, pour q une puissance d'un nombre premier. L'anneau des polynômes (en l'indéterminée X) à coefficients dans le corps $\mathbb{K}$ sera dénoté $\mathbb{K}[X]$. Pour tout réel x, ses arrondis à partie entière inférieure, supérieure, et à l'entier le plus proche sont respectivement notés $\lfloor x \rfloor$, $\lceil x \rceil$, et $\lfloor x \rceil$ respectivement (avec pour convention que $\lfloor 0.5 \rceil = 1$). Cette opération s'étend naturellement aux vecteurs et matrices coordonnée par coordonnée.

Vecteurs, matrices, et normes. Les vecteurs seront notés en lettres minuscules grasses et les matrices en lettres majuscules grasses. L'opérateur $^\top$ qui s'applique indifféremment aux vecteurs et aux matrices dénotera la transposition. Pour n'importe quel vecteur $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{Z}^n$ et pour $p \in \mathbb{N}^*$, la norme p de $\mathbf{x}$ est définie par :

$$\|\mathbf{x}\|_p = \ell_p(\mathbf{x}) = \sqrt[p]{\sum_{i=0}^{n-1} |x_i|^p}. \tag{2.1}$$

Par extension, $\ell_\infty(\mathbf{x}) = \max_i |x_i|$. La norme la plus couramment utilisée sera la norme 2, et nous omettrons parfois par soucis de clarté l'indice $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$.

Pour le chapitre 3, deux autres normes sont utilisées :

— la norme centrée : $\|\mathbf{s}\|_c^2 = \sum_{i=0}^{N-1} s_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} s_i\right)^2 = N \cdot \mathrm{Var}(s_i)$

— la norme balancée[1] : $\|(\mathbf{s}, \mathbf{t})\|_\nu^2 = \|\mathbf{s}\|_c^2 + \nu^2 \cdot \|\mathbf{t}\|_c^2$.

Ces deux normes sont typiques à NTRU et ne donnent pas beaucoup d'intuition sur la longueur réelle d'un vecteur au sens commun (euclidien) du terme. C'est pourquoi nous avons proposé une méthode *expérimentale* pour transformer cette norme balancée en norme euclidienne avec une bonne probabilité. Étant donné que $\|\mathbf{s}\|_c = \sigma_\mathbf{s}\sqrt{N}$, avec $\sigma_\mathbf{s}$ l'écart-type des s_is, chaque s_i vaut approximativement $\sigma_\mathbf{s}$, et par le lemme 2.2, on a :

$$\|\mathbf{s}\|_2 \leq \alpha \cdot \sigma_\mathbf{s}\sqrt{N} \text{ avec probabilité } 1 - 2^{-k}$$

où k est le paramètre de sécurité et le α correspondant peut être lu dans la table 3.1.

Complexités asymptotiques. Lorsque nous étudierons des quantités de manière asymptotique, nous utiliserons les notations de Landau classiques $o(\cdot)$, $\mathcal{O}(\cdot)$, $\omega(\cdot)$, $\Omega(\cdot)$, et $\Theta(\cdot)$,

1. "Balanced normed" en anglais, mais nous avons préféré cette traduction plutôt que norme "équilibrée" étant donné que cette norme introduit précisément un biais.

définies comme suit :

$$
\begin{aligned}
f(n) \in o(g(n)) &\Leftrightarrow \forall \epsilon > 0, \exists n_{0,\epsilon} \mid \forall n > n_{0,\epsilon}, |f(n)| \leq \epsilon \cdot |g(n)| \\
f(n) \in \mathcal{O}(g(n)) &\Leftrightarrow \exists k > 0, \exists n_{0,k} \mid \forall n > n_{0,k}, |f(n)| \leq k \cdot |g(n)| \\
f(n) \in \omega(g(n)) &\Leftrightarrow \forall k > 0, \exists n_{0,k} \mid \forall n > n_{0,k}, |f(n)| \geq k \cdot |g(n)| \\
f(n) \in \Omega(g(n)) &\Leftrightarrow \exists k > 0, \exists n_{0,k} \mid \forall n > n_{0,k}, |f(n)| \geq k \cdot |g(n)| \\
f(n) \in \Theta(g(n)) &\Leftrightarrow \exists k_1, k_2 > 0, \exists n_{0,k_1,k_2} \mid \forall n > n_{0,k_1,k_2}, k_1 \cdot |g(n)| \leq |f(n)| \leq k_2 \cdot |g(n)|
\end{aligned}
$$

Nous utiliserons également les notations "soft" $\tilde{\mathcal{O}}$, $\tilde{\Omega}$, et $\tilde{\Theta}$ définies de manière analogue, à un facteur poly-logarithmique près. De plus, pour un paramètre de sécurité λ fixé, nous dirons qu'une fonction f est négligeable si cette fonction vérifie $f(n) \in \lambda^{-\omega(1)}$.

Distributions. Pour un ensemble fini $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ signifie que x est échantillonné uniformément au hasard parmi les éléments de $\mathcal{S}$. Pour une variable aléatoire X, on note respectivement $\mathbb{E}[X]$ et $\mathrm{Var}(X)$ son espérance et sa variance. Pour toute distribution $\mathcal{D}$ sur $\mathbb{R}^n$, sa covariance est la matrice $n \times n$ symétrique semi-définie positive $\mathrm{Cov}(\mathcal{D}) = \mathbb{E}_{x \leftarrow \mathcal{D}}[\mathbf{x}^\top \mathbf{x}]$. La distribution normale (ou Gaussienne) sur $\mathbb{R}$, centrée en c et d'écart-type σ est notée $\mathcal{D}_{\mathbb{R},\sigma,c}$. Par extension, $\mathcal{D}_{\mathbb{R}^n,\sqrt{\mathbf{\Sigma}},\mathbf{c}}$ désigne la distribution normale sur $\mathbb{R}^n$, centrée en $\mathbf{c}$ et de matrice de covariance $\mathbf{\Sigma}$. Enfin, la loi de Bernouilli de paramètre $p \in [0,1]$ qui vaut 1 avec probabilité p et 0 avec probabilité $1-p$ est notée $\mathcal{B}_p$.

2.1.2 Statistiques et Probabilités

La plupart des notions et propriétés présentées dans cette section peuvent être retrouvées de manière plus détaillée dans [Fel08]. Nous utiliserons principalement au cours de ce mémoire la distance statistique et la divergence de Rényi. Il existe toutefois d'autres mesures employées pour les preuves de sécurité, telles que la divergence de Kullback-Leibler [DLP14].

Distance statistique. Il s'agit d'une mesure omniprésente en cryptographie qui permet de mesurer à quel point deux distributions sont proches l'une de l'autre.

Définition 1.1 (Distance statistique)**.** *Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux distributions sur un même ensemble dénombrable $\mathcal{S}$, leur distance statistique est définie par*

$$\Delta(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2} \sum_{x \in \mathcal{S}} |\mathcal{D}_1(x) - \mathcal{D}_2(x)|. \tag{2.2}$$

On montre assez simplement que cette mesure satisfait les trois propriétés propres aux distances. La distance statistique est notamment utile pour réduire la sécurité d'un cryptosystème ou d'un schéma de signature à un autre problème, en modifiant légèrement les distributions en jeu (schéma simulé).

Divergence de Rényi. Cette mesure partage certaines propriétés avec la distance statistique mais d'un point de vue multiplicatif. Elle s'est avérée particulièrement adaptée pour prouver la sécurité de primitives basées sur les réseaux dans le cas où un attaquant doit résoudre un problème de recherche [BLL+15].

Définition 1.2 (Divergence Rényi)**.** *Soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux distributions discrètes telles que $Supp(\mathcal{D}_1) \subseteq Supp(\mathcal{D}_2)$. Soit $p > 1$ un entier, la divergence Rényi d'ordre p de $\mathcal{D}_1$ et $\mathcal{D}_2$ est*

définie par

$$R_p(\mathcal{D}_1, \mathcal{D}_2) = \left(\sum_{x \in \mathrm{Supp}(\mathcal{D}_1)} \frac{\mathcal{D}_1(x)^p}{\mathcal{D}_2(x)^{p-1}} \right)^{\frac{1}{p-1}}. \tag{2.3}$$

Entropie. Mesure introduite par Claude SHANNON en 1948, l'entropie quantifie l'information présente dans un signal. Cette notion fut adaptée pour capturer le côté imprévisible d'une variable aléatoire.

Définition 1.3 (Entropie de Shannon)**.** *Soit $\mathcal{D}$ une distribution sur un ensemble dénombrable $\mathcal{S}$. L'entropie de Shannon associée à $\mathcal{D}$ est définie par :*

$$H(\mathcal{D}) = -\sum_{x \in \mathcal{S}} \mathcal{D}(x) \cdot \log_2 \mathcal{D}(x). \tag{2.4}$$

Remarque 1. *La notion d'entropie a été étendue en famille par Rényi. Pour tout $p \in \mathbb{N} \setminus \{1\}$, l'entropie de Rényi d'ordre p est définie par :*

$$H_p(\mathcal{D}) = \frac{1}{1-a} \log_2 \left(\sum_{x \in \mathcal{S}} \mathcal{D}(x)^p \right). \tag{2.5}$$

Remarque 2. *Conformément à la définition des fonctions d'entropie de Rényi, l'entropie de Shannon correspond à $\lim_{p \to 1} H_p$. De plus, H_∞ et H_0 sont respectivement appelées max- et min-entropies.*

Le lemme suivant est un outil relativement puissant en cryptographie, et permet d'affirmer que les fonctions de hachage sont de bons extracteurs de source d'aléa. Sa formulation nécessitant beaucoup d'autres notions dont nous n'aurons pas besoin dans la suite de cet ouvrage, nous redirigeons le lecteur vers [HILL99] pour une définition formelle.

Rejection sampling. Le rejet d'échantillons permet de s'assurer qu'une variable échantillonnée suivant une loi donnée suit en pratique cette loi. Le lemme suivant sera crucial pour annihiler la fuite d'information de NTRUSign dans le chapitre 3.

Lemme 1.1 (Rejection Sampling [Lyu12])**.** *Pour tout ensemble V, pour toute distribution de probabilité $h : V \to \mathbb{R}$ et $f : \mathbb{Z}^{2N} \to \mathbb{R}$, si $g_{\mathbf{v}} : \mathbb{Z}^{2N} \to \mathbb{R}$ est une famille de distributions de probabilité indexée par $v \in V$ tel que $\exists M \in \mathbb{R} \ / \ \forall v \in V, Pr[Mg_v(\mathbf{z}) \geq f(\mathbf{z}); \mathbf{z} \xleftarrow{\$} f] \geq 1 - \epsilon$, alors les sorties des algorithmes $\mathcal{A}$ et $\mathcal{F}$ suivants ont une distance statistique inférieure à ϵ/M.*

Algorithme $\mathcal{A}$
1: $v \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} g_v$
3: *output* $(\mathbf{z}, v)$ *avec probabilité* $\min\left(\frac{f(\mathbf{z})}{Mg_v(\mathbf{z})}, 1\right)$

Algorithme $\mathcal{F}$
1: $v \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} f$
3: *output* $(\mathbf{z}, v)$ *avec probabilité* $1/M$

2.1.3 Sécurité Prouvée

Comme précisé plus haut dans la Sec. 1.3.1, la cryptographie a besoin de problèmes difficiles pour que la sécurité puisse tenir au cours du temps, mais pas non plus infaisables de sorte à pouvoir générer des clés. En termes de complexité, cela revient à trouver des

instances de problèmes prouvées ou supposées NP-difficiles, mais pour lesquelles on connaît une trappe permettant de résoudre efficacement (en temps polynomial) ce problème. Dans cette section, nous rappelons comment fonctionne une réduction de sécurité d'une primitive cryptographique à un problème, comment les jeux permettent d'effectuer de telles réductions, ainsi que les différents arguments et modèles existants.

Réduction de sécurité. Une réduction de sécurité d'une primitive cryptographique à un problème connu est un procédé, déterministe ou probabiliste, permettant de transformer un adversaire $\mathcal{A}$ qui résout cette primitive en un algorithme $\mathcal{A}^\star$ permettant de résoudre ledit problème. Par contraposée, si le problème est réputé difficile, alors la primitive cryptographique l'est aussi.

Jeux. Les réductions de sécurité passent par des jeux de sécurité, éventuellement interactifs, dans lesquels un adversaire $\mathcal{A}$ agissant comme une machine de Turing reçoit un challenge $\mathcal{C}$, effectue une succession d'opérations, avant de fournir une réponse $\mathcal{R}$ à ce challenge. Le jeu de sécurité précise dans quelle(s) mesure(s) la réponse fournie par l'adversaire est correcte, auquel cas l'adversaire remporte le jeu et de fait, casse la primitive cryptographique. Les nombres de requêtes et d'opérations faites par cet adversaire ainsi que sa probabilité de remporter le jeu sont importants pour les réductions de sécurité afin de pouvoir fixer des paramètres.

Argument hybride. Les jeux ne peuvent parfois pas confronter directement une primitive cryptographique à un problème bien connu. Il est alors nécessaire de construire une séquence de jeux, différant légèrement les uns des autres (généralement de par les distributions utilisées), et aboutissant à un problème connu. L'argument hybride consiste à montrer que deux jeux consécutifs sont indistinguables l'un de l'autre (en utilisant un problème connu ou un argument statistique), et donc que si le problème connu est difficile, alors la primitive cryptographique à l'origine de la séquence de jeux l'est aussi. Nous utiliserons ce type d'argument dans les chapitres 4 et 6.

Modèles classique et de l'oracle aléatoire. Random Oracle Model (ROM) en anglais, c'est une hypothèse selon laquelle les fonctions de hachage, qui sont des fonctions dites "à sens unique", se comportent comme des fonctions purement aléatoire, dont personne (honnête ou malhonnête) ne peut prédire la sortie. Le ROM est de plus en plus décrié car les fonctions de hachage ne sont pas réellement aléatoires, et bien que certaines constructions soient sures dans le ROM, elles ne le sont plus lorsque instanciées avec une fonction de hachage [LN09]. À l'inverse, les constructions dans le modèle classique se ramènent généralement directement à un problème difficile connu, ou du moins ne reposent par sur l'aspect aléatoire des fonctions de hachage.

2.1.4 Primitives Cryptographiques et Modèles de Sécurité

Dans cette section, nous décrivons les principales primitives sur lesquelles nous avons travaillé au cours de cette thèse, ainsi que les modèles de sécurité associés.

2.1.4.1 Chiffrement

Comme nous l'avons vu dans le chapitre 1, le chiffrement est la primitive historique de la cryptographie. Nous donnons ici une définition plus formelle du chiffrement asymétrique, suivie du modèle de sécurité associé. Ce modèle sera celui retenu pour le chapitre 6.

Définition 1.4 (Chiffrement asymétrique)**.** *Un schéma de chiffrement asymétrique est composé de quatre algorithmes polynomiaux* (Setup, KeyGen, Encrypt, Decrypt) *:*

0. $\mathsf{Setup}(1^\lambda)$ *: à partir du paramètre de sécurité* λ*, génère les paramètres globaux* param *du système*
1. $\mathsf{KeyGen}(\mathsf{param})$ *: à partir des paramètres du système, génère les clés publique* pk *et privée* sk
2. $\mathsf{Encrypt}(\mathsf{pk}, \mu, \theta)$ *: génère un chiffré* $\mathbf{c}$ *du message* μ *en utilisant l'aléa* θ
3. $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$ *: retourne le clair* $\boldsymbol{\mu}$ *correspondant à* $\mathbf{c}$ *ou* $\perp$*.*

Les schémas de chiffrement asymétriques doivent vérifier deux propriétés : la *correctness* (ou exactitude) qui garantie que les messages chiffrés avec la clé publique sont déchiffrés correctement en utilisant la clé secrète associée ; et l'*indistinguabilité contre les attaques à clairs choisis* (IND-CPA), qui garantie que le schéma de chiffrement reste sûr face à un adversaire polynomial ayant accès à des couples message clair/chiffré correspondants [2]. Ces deux notions sont capturées par les définitions formelles suivantes :

Définition 1.5 (Correctness)**.** *Pour toute paire de clé* (pk, sk) *générée par* KeyGen*, tout message* μ*, et tout aléa* θ*, le schéma de chiffrement doit vérifier*

$$\Pr\left[\mathsf{Decrypt}\left(\mathsf{sk}, \mathsf{Encrypt}\left(\mathsf{pk}, \boldsymbol{\mu}, \theta\right)\right) \neq \mu\right] < 1 - 2^{-\lambda}. \tag{2.6}$$

Définition 1.6 (INDistinguishability under Chosen Plaintext Attacks (IND-CPA) [GM84])**.**

Cette notion est formalisée par le jeu de sécurité ci-contre. L'avantage qu'un adversaire polynomial a de retrouver quel message a été chiffré doit être négligeable, même s'il a choisit les messages à la base. Formellement, l'avantage d'un adversaire est

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\lambda) = 1]\right|.$$

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1) \leftarrow \mathcal{A}(\textit{FIND} : \mathsf{pk})$
4. $\mathbf{c}^* \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \boldsymbol{\mu}_\mathbf{b}, \theta)$
5. $b' \leftarrow \mathcal{A}(\textit{GUESS} : \mathbf{c}^*)$
6. *RETURN* b'

Un schéma de chiffrement est IND-CPA si et seulement si $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \lambda^{-\omega(1)}$.

2.1.4.2 Chiffrement homomorphe

Un schéma de chiffrement (asymétrique) est dit homomorphe s'il existe une opération sur les chiffrés permettant de manipuler les clairs sous-jacents de manière contrôlée.

Définition 1.7 (Chiffrement homomorphe)**.** *Soient* $\mathcal{S}$ *un schéma de chiffrement,* (pk, sk) *les clés associées,* μ_1, μ_2 *deux textes clairs, et* c_1, c_2 *leur chiffrés associés* ($c_i = \mathsf{Encrypt}(\mathsf{pk}, \mu_i, \theta_i)$)*. Ce schéma* $\mathcal{S}$ *est homomorphe s'il existe un opérateur binaire* $\square$ *sur l'espace des chiffrés et un opérateur binaire* $\triangle$ *sur l'espace des clairs tels que :*

$$\mathsf{Decrypt}(c_1 \square c_2) = \mu_1 \triangle \mu_2. \tag{2.7}$$

$\mathcal{S}$ *sera dit* complètement *homomorphe s'il existe des couples de tels opérateurs permettant d'évaluer n'importe quelle fonction polynôme sur les données en clair.*

2. Il existe des notions de sécurité plus fortes au sens où l'adversaire a davantage de connaissances sur le schéma de chiffrement (voir IND-CCA et IND-CCA2)

Il s'est avéré que le premier schéma de chiffrement asymétrique [RSA78] était ("accidentellement") homomorphe multiplicativement. En effet, dans la version originale, le fait de multiplier entre eux deux chiffrés et de les réduire modulo N permettait d'obtenir un chiffré dont le clair sous-jacent était la multiplication (modulo N) des deux clairs initiaux. Cette propriété a rapidement amené la question de savoir s'il existait ou non un schéma complètement homomorphe [RAD78]. Cette question est restée plus de 30 ans en suspens avant que Gentry ne propose le premier schéma de chiffrement complètement homomorphe [Gen09a]. S'en est suivi un engouement pour les réseaux et d'autres schémas ont rapidement fait leur apparition.

Nous avons réalisé en 2012 un état de l'art des cryptosystèmes homomorphes [Den12]. Une implémentation de certains de ces cryptosystèmes fut proposée plus tard [HS14], intégrant les cryptosystèmes BGV, FV et YASHE [BLLN13] (ce dernier ne figurant pas dans le précédent état de l'art). Cette librairie de calcul sur les données chiffrées sera modifiée et utilisée dans le chapitre 5.

2.1.4.3 Signature

Parmi les principales primitives cryptographiques, la signature électronique permet d'assurer l'intégrité, l'authentification et/ou la non-répudiation. Nous rappelons ci-dessous la définition de la signature électronique ainsi que le modèle de sécurité associé.

Définition 1.8 (Signature)**.** *Un schéma de signature S est composé de trois algorithmes polynomiaux* `KeyGen`*,* `Sign` *et* `Verify`*. Un algorithme additionnel* `Setup` *peut être nécessaire pour définir les paramètres nécessaires à la génération de clés.*

0. [Setup] *: génère les paramètres du schéma de signature à partir du paramètre de sécurité ;*
1. KeyGen *: génère une clé secrète sk dite de signature et une clé publique pk dite de vérification à partir du paramètre de sécurité et éventuellement de ceux fournis par* `Setup` *;*
2. Sign *: génère une signature σ pour le message μ en utilisant la clé secrète sk ;*
3. Verify *: vérifie si une signature σ est valide ou non pour un message μ en utilisant la clé publique pk.*

La première propriété requise pour un schéma de signature est sa consistance. Toute signature légitime doit pouvoir passer l'algorithme de vérification. Cette propriété est formalisée comme suit :

Définition 1.9 (Consistance)**.** *Soit S un schéma de signature associé au paramètre de sécurité λ. Ce schéma est consistant si et seulement si:*

$$\forall (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda),\ \Pr[\mathsf{Verify}(pk, \mu, \mathsf{Sign}(sk, \mu)) = 1] > 1 - 2^{-\lambda}. \tag{2.8}$$

Les deux principales façons d'attaquer un schéma de signature sont soit de réussir à créer un couple message/signature valide (et différent de ceux déjà publiés), soit de retrouver la clé de signature directement à partir de la clé publique. Cette première attaque, nommée forgerie ou falsification, se formalise ainsi :

Définition 1.10 (Forgeries)**.** *Un schéma de signature S est dit résistant aux forgeries (ou falsifications) si pour tout adversaire $\mathcal{A}$ (s'exécutant en temps polynomial) ayant accès à la clé publique ainsi qu'à n couples messages/signatures (μ_i, σ_i) de son choix, $\mathcal{A}$ n'a qu'un avantage négligeable (en le paramètre de sécurité) pour produire un couple message/signature $(\mu, \sigma) \neq (\mu_i, \sigma_i)$ tel que* `Verify`$(\mu, \sigma) = 1$*, c'est-à-dire un couple valide.*

2.1.4.4 Signature de groupe, signature traçable

Introduite il y a 25 ans [Cv91], la signature de groupe est une primitive cryptographique permettant à n'importe quel membre appartenant à une entité de signer au nom de celle-ci, sans que la personne qui vérifie la signature ne sache qui au sein de cette entité a signé. Cette primitive a trouvé de nombreuses applications, telles que l'informatique de confiance (trust computing), la gestion d'accès à des zones réglementées, ou encore certains protocoles d'enchères. Cependant, l'administrateur de l'entité a le pouvoir de révéler quel membre a produit telle ou telle signature, même si cet utilisateur n'a pas agit malicieusement.

Le formalisme des signatures de groupe a été introduit en 2003 [BMW03], puis étendu au cas des groupes dynamiques (où le nombre d'utilisateurs varie au cours du temps) [BSZ05]. Ce dernier modèle de sécurité est fortement inspiré d'une extension des signatures de groupes : les signatures traçables [KTY04]. Celles-ci apportent plus de respect de la vie privée des membres de l'entité car l'administrateur ne peut pas "ouvrir" une signature (révéler l'identité du signataire), mais seulement vérifier si tel ou tel membre a produit ou non une signature donnée. De plus, cette capacité de "tracer" une signature peut être déléguée de manière efficace à des tiers s'exécutant en parallèle. Nous conserverons donc le modèle des signatures traçables, auquel nous nous référerons en tant que modèle Kiayias, Tsiounis, Yung (KTY).

Définition 1.11 (Signature traçable)**.** *Un schéma de signature traçable* $\mathcal{TS}$ *est composé d'un tuple d'algorithmes polynomiaux* $\mathcal{TS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verif}, \mathsf{Open}, \mathsf{Reveal}, \mathsf{Trace}, \mathsf{Claim})$ *définis comme suit :*

- $\mathsf{Setup}(1^\lambda)$ *: génère les paramètres globaux du système, la clé publique du groupe* pk *ainsi que les clés privées : la clé secrète maître* msk *pour l'administrateur du groupe (group manager* $\mathcal{GM}$*), et la clé d'ouverture* $\mathsf{sk_O}$ *pour l'autorité responsable des ouvertures de signatures (Opener).*
- $\mathsf{Join}(\mathcal{U}_i)$ *: protocole interactif entre un utilisateur* $\mathcal{U}_i$ *et le group manager (utilisant sa clé maître* msk*). À la fin du protocole, l'utilisateur obtient sa clé de signature* $\mathsf{usk}[i]$*, et le group manager ajoute l'utilisateur à la liste des utilisateurs enregistrés, en stockant certaines informations dans la table Reg. Nous dénotons* $\mathcal{S}$ *l'ensemble des indices des utilisateurs enregistrés.*
- $\mathsf{Sign}(\mathsf{pk}, i, \mu, \mathsf{usk}[i])$ *: protocole exécuté par un utilisateur enregistré* $\mathcal{U}_i$ *avec* $i \in \mathcal{S}$*, utilisant sa clé secrète* $\mathsf{usk}[i]$*. Produit une signature* σ *pour le message* μ*.*
- $\mathsf{Verif}(\mathsf{pk}, \mu, \sigma)$*: protocole exécuté par un vérifieur, utilisant la clé publique du groupe* pk*. Retourne 1 si et seulement si* σ *est une signature valide pour le message* μ*.*
- $\mathsf{Open}(\mathsf{pk}, \mu, \sigma, \mathsf{sk_O})$ *: Si* σ *est valide, l'Opener utilisant* $\mathsf{sk_O}$ *retourne l'indice* i *de l'utilisateur supposé avoir signé le message, conjointement avec une preuve* Π_O *publiquement vérifiable de cette affirmation.*
- $\mathsf{Reveal}(\mathsf{pk}, i, \mathsf{sk_O})$ *: sur l'entrée* $\mathsf{sk_O}$ *et un indice cible* i*, retourne une clé de traçage* $\mathsf{tk}[i]$ *propre à l'utilisateur* i*, avec une preuve* $\Pi_\mathcal{E}$ *certifiant que* $\mathsf{tk}[i]$ *permet bel et bien de tracer l'utilisateur* i*.*
- $\mathsf{Trace}(\mathsf{pk}, \mu, \sigma, \mathsf{tk}[i])$ *: en utilisant la clé de traçage* $\mathsf{tk}[i]$*, retourne 1 ssi* σ *est une signature valide produite par l'utilisateur* i*, avec une preuve* Π_{sO} *de cette affirmation.*
- $\mathsf{Claim}(\mathsf{pk}, \mu, \sigma, \mathsf{usk}[i])$ *: en utilisant la clé secrète* $\mathsf{usk}[i]$*, cet algorithme retourne 1 ssi* σ *est une signature valide produite par l'utilisateur* i*, avec une preuve* Π_c *de cette affirmation.*

L'ensemble des algorithmes d'une signature traçable ainsi que les différents acteurs sont synthétisés Figure 4.1.

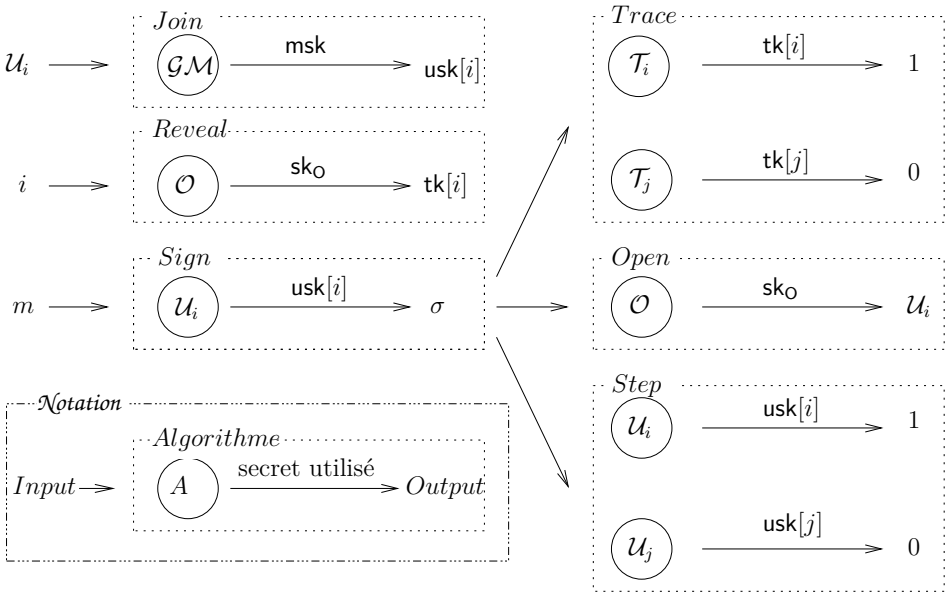


FIGURE 2.1 – Algorithmes et acteurs d'un schéma de signature traçable.

2.2 Réseaux Euclidiens

Dans cette section, nous présentons l'un des outils de base de cette thèse : les réseaux euclidiens (lattices en anglais). Les réseaux sont des sous-groupes discrets de $\mathbb{R}^n$, c'est-à-dire qu'ils sont stables par addition, possèdent un élément neutre pour cette opération, et il y a toujours un "blanc" entre deux points d'un réseau, aussi proches soient-ils. Nous commençons cette section par quelques définitions et propriétés des réseaux, avant de décrire les principaux problèmes sur lesquels reposeront les primitives cryptographiques des chapitres 3, 4, et 5.

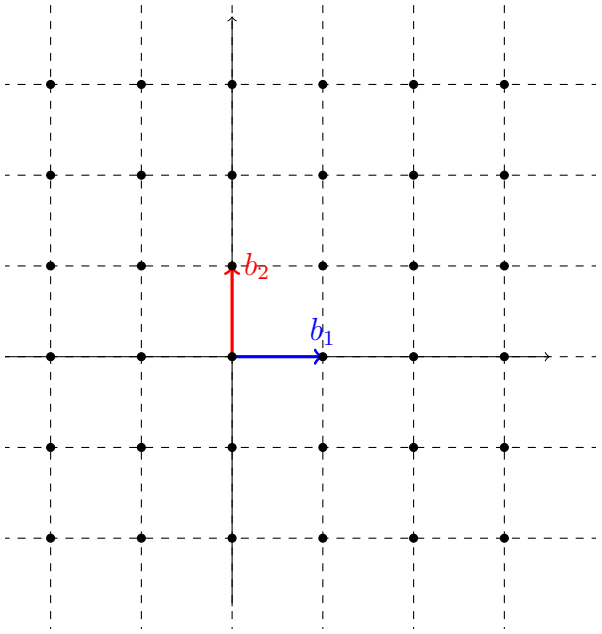
2.2.1 Définitions et propriétés

Formellement, nous adopterons la définition suivante d'un réseau euclidien :

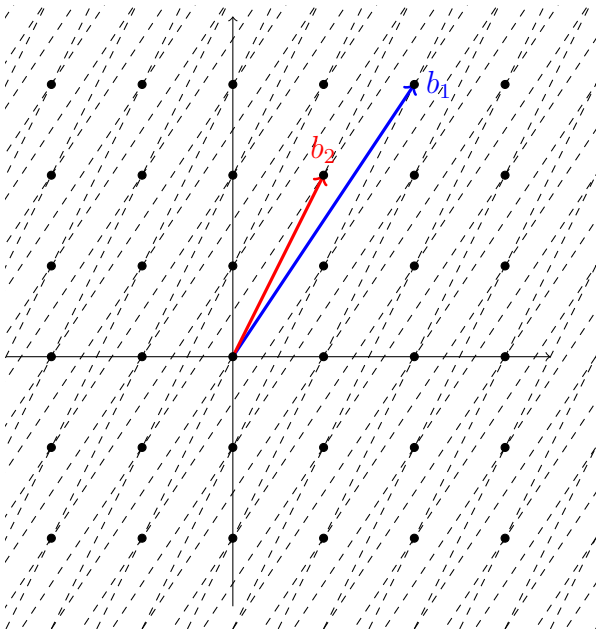
Définition 2.12 (Réseau)**.** *Soient $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ n vecteurs à m coordonnées. Le réseau engendré par ces vecteurs est donné par ses combinaisons linéaires à coefficients entiers :*

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}. \tag{2.9}$$

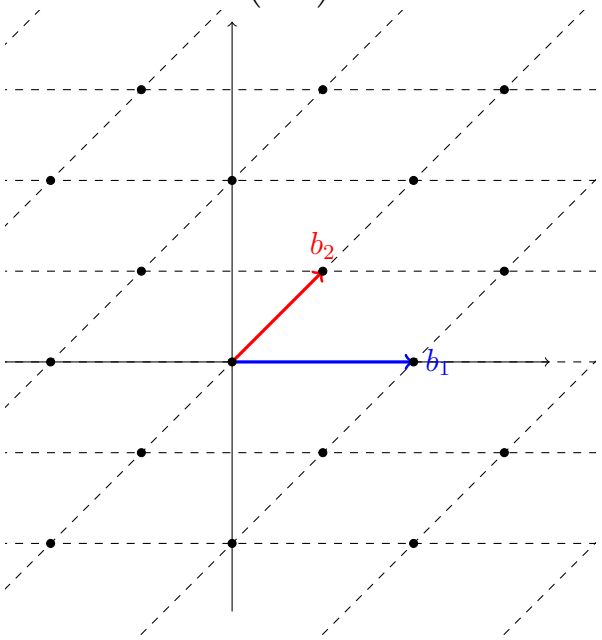
Bien que l'ensemble des vecteurs $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ forme la base du réseau*, il est fréquent de considérer la matrice $\mathbf{B} = \begin{pmatrix} b_{11} & b_{21} & \ldots & b_{n1} \\ b_{12} & b_{22} & \ldots & b_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1m} & b_{2m} & \ldots & b_{nm} \end{pmatrix} \in \mathbb{R}^{m \times n}$ formée des vecteurs $\mathbf{b}_1, \ldots, \mathbf{b}_n$ en colonne comme base du réseau. m est la* dimension *du réseau, et n est son* rang*. Lorsque $m = n$, on dit que le réseau est de* rang plein*.*



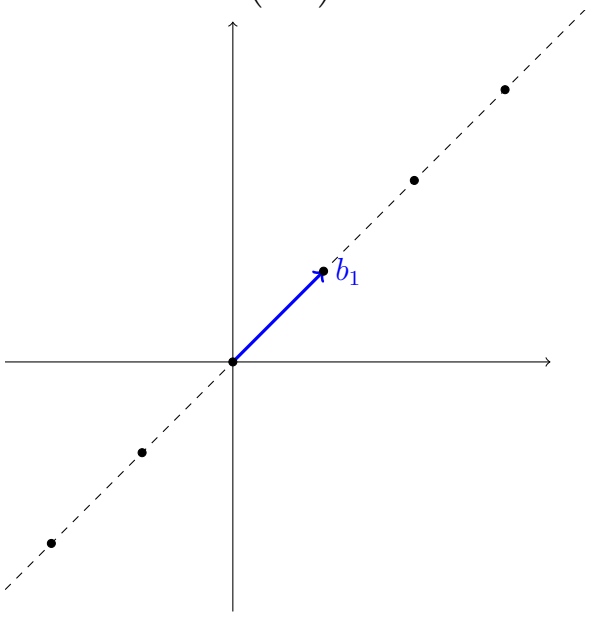
(a) $\mathbb{Z}^2$ avec $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ comme base.



(b) $\mathbb{Z}^2$ avec $\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$ comme base.



(c) Sous-réseau de $\mathbb{Z}^2$ avec $\begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ comme base.



(d) Sous-réseau de $\mathbb{Z}^2$ avec $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ comme base.

FIGURE 2.2 – Exemples de réseaux à deux dimensions.

La Figure 2.2 suivante présente des exemples de réseaux à 2 dimensions.

Bien qu'il n'y ait pas d'obligation, les coordonnées des vecteurs formant la base du réseau seront souvent entières. Formellement $\mathbf{b}_i \in \mathbb{Z}^m, \forall i \in \{1, \ldots, n\}$. Dans la suite, on écrira Vect($\mathcal{L}(\mathbf{B})$) ou juste $Vect(\mathbf{B})$ pour désigner l'espace vectoriel engendré par les vecteurs de la

base :

$$\mathrm{Vect}(\mathcal{L}(\mathbf{B})) = \mathrm{Vect}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} y_i \mathbf{b}_i \mid y_i \in \mathbb{R} \right\}. \tag{2.10}$$

Définition 2.13 (Parallélépipède fondamental)**.** *Soit $\mathcal{L}(\mathbf{B})$ un réseau euclidien. Le parallélépipède fondamental de $\mathcal{L}(\mathbf{B})$ est défini par :*

$$\mathcal{P} = \mathcal{P}(\mathbf{B}) = \{\mathbf{B}\mathbf{y} | \mathbf{y} \in [0,1[^n\} = \left\{ \mathbf{B}\mathbf{y} \mid \mathbf{y} \in \left[-\frac{1}{2}, \frac{1}{2}\right[^n \right\}. \tag{2.11}$$

La Figure 2.3 reprend les exemples donnés dans la Figure 2.2 et y intègre une copie du parallélépipède fondamental.

On remarque qu'en reportant une copie du parallélépipède fondamental à chaque point du réseau, on obtient l'espace vectoriel engendré par cette base.

Définition 2.14 (Réseaux équivalents)**.** *On dit que deux réseaux $\mathcal{L}(\mathbf{B}_1)$ et $\mathcal{L}(\mathbf{B}_2)$ sont équivalents si et seulement si il existe une matrice unimodulaire*[3] *$\mathbf{U}$ à coefficients dans $\mathbb{Z}$ telle que $\mathbf{B}_1 = \mathbf{B}_2\mathbf{U}$. On écrira $\mathcal{L}(\mathbf{B}_1) = \mathcal{L}(\mathbf{B}_2)$*

Définition 2.15 (Déterminant)**.** *Le déterminant $det(\mathcal{L}(\mathbf{B}))$ d'un réseau $\mathcal{L}(\mathbf{B})$ est défini comme étant le volume du parallélépipède fondamental $\mathcal{P}(\mathbf{B})$:*

$$\det(\mathcal{L}) = \mathrm{vol}(\mathcal{P}) = \sqrt{\det(\mathbf{B}^\top \mathbf{B})}. \tag{2.12}$$

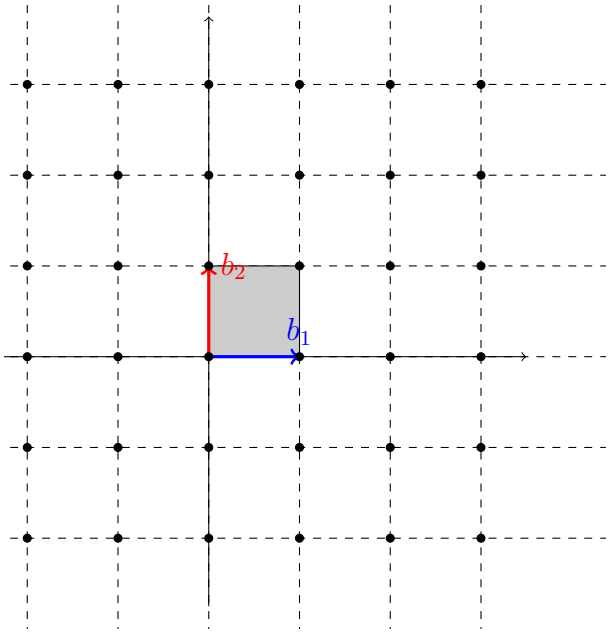
Proposition 2.1. *Bien que la notion de parallélépipède fondamental dépende de la base choisie, celle de déterminant a du sens car elle ne dépend pas de la base choisie.*

Démonstration. En effet, si $\mathbf{B}_1$ et $\mathbf{B}_2$ sont deux bases d'un même réseau, alors il existe $\mathbf{U}$ unimodulaire de sorte que $\mathbf{B}_1 = \mathbf{B}_2\mathbf{U}$, donc

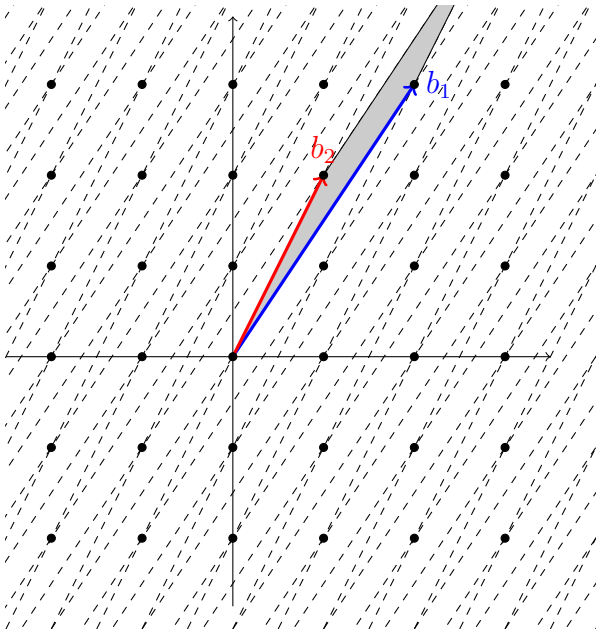
$$\begin{aligned}
\mathrm{vol}(\mathcal{P}(\mathbf{B}_1)) &= \sqrt{\det(\mathbf{B}_1^\top \mathbf{B}_1)} \\
&= \sqrt{\det((\mathbf{B}_2\mathbf{U})^\top (\mathbf{B}_2\mathbf{U}))} \\
&= \sqrt{\det(\mathbf{U}^\top) \cdot \det(\mathbf{B}_2^T \mathbf{B}_2) \cdot \det(\mathbf{U})} \\
&= \sqrt{\det(\mathbf{B}_2^\top \mathbf{B}_2) \cdot \det(\mathbf{U})^2} \\
&= \sqrt{\det(\mathbf{B}_2^\top \mathbf{B}_2)} \\
&= \mathrm{vol}(\mathcal{P}(\mathbf{B}_2)).
\end{aligned}$$

□

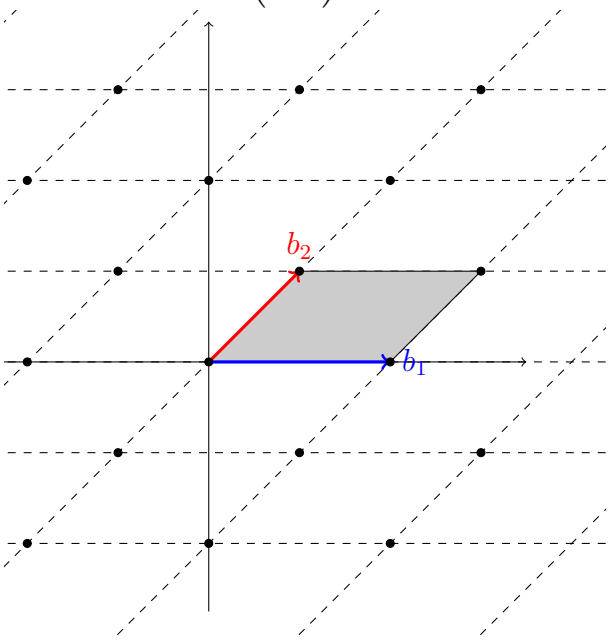
3. Une matrice est dite unimodulaire si son déterminant est ± 1, ce qui impose implicitement qu'elle soit carrée



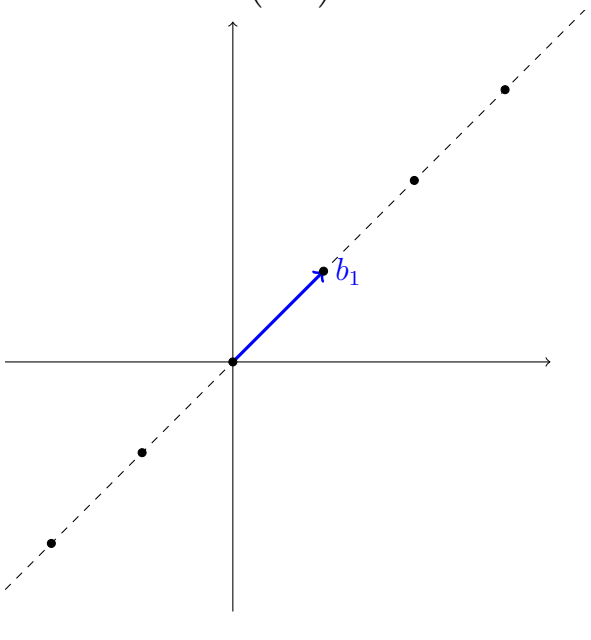
(a) $\mathbb{Z}^2$ avec $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ comme base.



(b) $\mathbb{Z}^2$ avec $\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$ comme base.



(c) Sous-réseau de $\mathbb{Z}^2$ avec $\begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ comme base.



(d) Sous-réseau de $\mathbb{Z}^2$ avec $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ comme base.

FIGURE 2.3 – Parallélépipèdes fondamentaux de réseaux deux dimensions.

Un des problèmes les plus connus en réseau est celui de la recherche du/des plus court(s) vecteur(s). Typiquement, dans un réseau $\mathcal{L}(\mathbf{B})$, tout vecteur non-nul $\mathbf{x}$ à une longueur strictement positive $\|\mathbf{x}\| = \ell_2(\mathbf{x}) > 0$. La question est de savoir si cette longueur est relativement petite par rapport aux autres vecteurs du réseau.

Définition 2.16 (Minima successifs)**.** *Soit* $\mathcal{L}(\mathbf{B})$ *un réseau de base* $\mathbf{B} \in \mathbb{R}^{m \times n}$*. Pour* $i \in \{1, \ldots, n\}$ *on définit le* i-ème minimum successif λ_i *par :*

$$\lambda_i = \inf \left\{ r \mid \dim(\mathrm{Vect}(\mathcal{L} \cap \bar{\mathcal{B}}_m(\mathbf{0}, r)) \geq i \right\}, \tag{2.13}$$

où $\bar{\mathcal{B}}_m(\mathbf{0}, r) = \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x}\| \leq r\}$ *désigne la boule fermée en dimension* m *centrée en* $\mathbf{0} \in \mathcal{L}(B)$ *et de rayon* r*.*

La Figure 2.4 ci-dessous montre un exemple de réseau à 2 dimensions en faisant apparaître ses minima successifs.

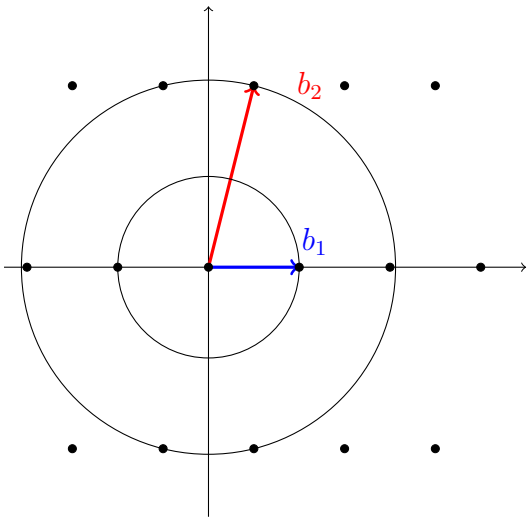


FIGURE 2.4 – Exemple de minima successifs : $\lambda_1 = 1$, $\lambda_2 = 2.06$.

$\lambda_1(\mathcal{L}(\mathbf{B}))$ n'est rien d'autre que la distance du plus petit vecteur du réseau $\mathcal{L}(\mathbf{B})$. Comme nous le verrons dans la section 2.2.2, trouver cette valeur n'est pas évident. Cependant, le théorème suivant donne une borne inférieure - hélas assez large - sur cette distance.

Théorème 2.2. *Avec les notations précédentes :*

$$\lambda_1(\mathcal{L}(\mathbf{B})) \geq \min_{i \in \{1,\dots,n\}} \|\tilde{\mathbf{b}}_i\|, \tag{2.14}$$

où les vecteurs $\tilde{\mathbf{b}}_i$ *correspondent à l'orthogonalisation de Gram-Schmidt des vecteurs* $\mathbf{b}_i$ *dans l'ordre lexicographique.*

Démonstration. Soient $\mathbf{x} \in \mathbb{Z}^n$ un vecteur non-nul quelconque, et $j \in \{1, \dots, n\}$ le plus grand entier tel que $x_j \neq 0$. Alors

$$|\langle \mathbf{Bx}, \tilde{\mathbf{b}}_j \rangle| = |\langle \sum_{i=1}^{j} x_i \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle| = |x_j| \langle \mathbf{b}_j, \tilde{\mathbf{b}}_j \rangle = |x_j| \langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle = |x_j| \cdot \|\tilde{\mathbf{b}}_j\|^2. \tag{2.15}$$

D'après Cauchy-Schwarz,

$$|\langle \mathbf{Bx}, \tilde{\mathbf{b}}_j \rangle| \leq \|\mathbf{Bx}\| \cdot \|\tilde{\mathbf{b}}_j\|. \tag{2.16}$$

En combinant les deux inégalités, on obtient

$$\|\mathbf{Bx}\| \geq |x_j| \cdot \|\tilde{\mathbf{b}}_j\| \geq \|\tilde{\mathbf{b}}_j\| \geq \min_{i \in \{1,\dots,n\}} \|\tilde{\mathbf{b}}_i\|. \tag{2.17}$$

□

De plus, les minima successifs sont atteints, dans le sens où $\forall i \in \{1, \dots, n\}$, il existe un vecteur $\mathbf{v}_i \in \mathcal{L}(\mathbf{B})$ tel que $\|\mathbf{v}_i\| = \lambda_i$

D'autre part, il existe des bornes supérieures pour les minima successifs, données par les théorèmes de Minkowski. Afin de présenter les deux théorèmes sur les bornes supérieures, nous aurons besoin des deux théorèmes suivant :

Théorème 2.3 (Blichfield)**.** *Pour tout réseau $\mathcal{L} \subseteq \mathbb{R}^n$ de rang plein, et tout ensemble $\mathcal{S} \subseteq \mathbb{R}^n$, si $\mathrm{vol}(\mathcal{S}) > \det(\mathcal{L})$, alors il existe deux points distincts $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{S}$, tels que $\mathbf{z}_1 - \mathbf{z}_2 \in \mathcal{L}$.*

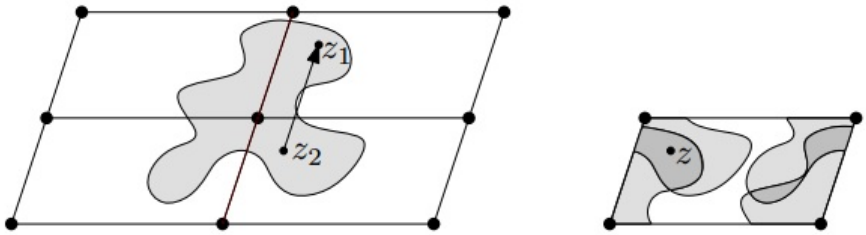


FIGURE 2.5 – Illustration du théorème de Blichfield. On est sûr de trouver deux points distincts $\mathbf{z}_1 \neq \mathbf{z}_2$ tels que $\mathbf{z}_1 - \mathbf{z}_2 \in \mathcal{L}$ dans les zones en gris foncé.

Démonstration. Soit $\mathbf{B}$ une base de $\mathcal{L} = \mathcal{L}(\mathbf{B})$. Pour $\mathbf{x} \in \mathcal{L}(\mathbf{B})$, les ensembles $\mathbf{x} + \mathcal{P}(\mathbf{B}) = \{\mathbf{x} + \mathbf{y} \mid \mathbf{y} \in \mathcal{P}(\mathbf{B})\}$ partitionnent $\mathbb{R}^n$. Soient $\mathcal{S}_\mathbf{x} = \mathcal{S} \cap (\mathbf{x} + \mathcal{P}(\mathbf{B}))$ et $\mathcal{S} = \bigcup_{\mathbf{x}\in\mathcal{L}} \mathcal{S}_\mathbf{x}$ (voir Figure 2.5). Comme cette union est disjointe, $\mathrm{vol}(\mathcal{S}) = \sum_{\mathbf{x}\in\mathcal{L}(\mathbf{B})} \mathrm{vol}(\mathcal{S}_\mathbf{x})$. Soit $\tilde{\mathcal{S}}_\mathbf{x} = \mathcal{S}_\mathbf{x} \backslash \{\mathbf{x}\} \subseteq \mathcal{P}(\mathbf{B})$. Alors $\mathrm{vol}(\tilde{\mathcal{S}}_\mathbf{x}) = \mathrm{vol}(\mathcal{S}_\mathbf{x})$ et donc

$$\sum_{\mathbf{x}\in\mathcal{L}} \mathrm{vol}(\tilde{\mathcal{S}}_\mathbf{x}) = \sum_{\mathbf{x}\in\mathcal{L}} \mathrm{vol}(\mathcal{S}_\mathbf{x}) = \mathrm{vol}(\mathcal{S}) > \mathrm{vol}(\mathcal{P}(\mathbf{B})). \tag{2.18}$$

Comme $\sum_{\mathbf{x}\in\mathcal{L}} \mathrm{vol}(\tilde{\mathcal{S}}_\mathbf{x}) = \mathrm{vol}(\mathcal{P}(\mathbf{B}))$, alors il existe deux points $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{L}$ tels que $\mathbf{z}_1 \neq \mathbf{z}_2$ et $\tilde{\mathcal{S}}_{\mathbf{z}_1} \cap \tilde{\mathcal{S}}_{\mathbf{z}_2} \neq \emptyset$. Soit $\mathbf{z} \in \tilde{\mathcal{S}}_{\mathbf{z}_1} \cap \tilde{\mathcal{S}}_{\mathbf{z}_2}$, alors $\mathbf{z} + \mathbf{z}_1 \in \mathcal{S}_{\mathbf{z}_1} \subseteq \mathcal{S}$ et $\mathbf{z} + \mathbf{z}_2 \in \mathcal{S}_{\mathbf{z}_2} \subseteq \mathcal{S}$ et $\mathbf{z}_1 - \mathbf{z}_2 = (\mathbf{z} + \mathbf{z}_1) - (\mathbf{z} + \mathbf{z}_2) \in \mathcal{L}$. □

Le *théorème du corps convexe de Minkowski* suivant peut être vu comme un corollaire du théorème de Blichfield. Il affirme que tout ensemble convexe centré en l'origine suffisamment large contient au moins un vecteur non nul du réseau. Pour rappel, un ensemble $\mathcal{S}$ est centré en l'origine si pour tout $\mathbf{x} \in \mathcal{S}$, $-\mathbf{x} \in \mathcal{S}$, et il est convexe si pour tout couple de point $(\mathbf{x}, \mathbf{y})$, $\mathcal{S}$ contient tous les points du segment reliant $\mathbf{x}$ à $\mathbf{y}$, formellement $\lambda \cdot \mathbf{x} + (1 - \lambda) \cdot \mathbf{y} \in \mathcal{S}$, $\forall \lambda \in [0, 1]$.

Théorème 2.4 (Corps convexe de Minkowski)**.** *Soit $\mathcal{L} \in \mathbb{R}^n$ un réseau de rang plein. Alors pour tout ensemble S convexe centré en l'origine, si $\mathrm{vol}(\mathcal{S}) > 2^n \cdot \det(\mathcal{L})$, alors $\mathcal{S}$ contient un vecteur $\mathbf{v} \in \mathcal{L}$ non-nul.*

Démonstration. Soit $\tilde{\mathcal{S}} = \frac{1}{2}\mathcal{S} = \{\mathbf{x} \mid 2\mathbf{x} \in \mathcal{S}\}$. Alors $\mathrm{vol}(\tilde{\mathcal{S}}) = 2^{-n}\mathrm{vol}(\mathcal{S}) > \det(\mathcal{L})$ par hypothèse. Donc d'après le théorème 2.3 de Blichfield, $\exists \mathbf{z}_1, \mathbf{z}_2 \in \tilde{\mathcal{S}} / \mathbf{z}_1 - \mathbf{z}_2 \in \mathcal{L} \backslash \{\mathbf{0}\}$. Par définition, $2\mathbf{z}_1, 2\mathbf{z}_2 \in \mathcal{S}$, donc $-2\mathbf{z}_2 \in \mathcal{S}$ car $\mathcal{S}$ est centré en l'origine. Enfin $\frac{1}{2}(2\mathbf{z}_1 - 2\mathbf{z}_2) = (\mathbf{z}_1 - \mathbf{z}_2) \in \mathcal{S}$ car $\mathcal{S}$ est convexe. La preuve de ce théorème est illustrée par la figure 2.6. □

Proposition 2.5. *Le volume de la boule de rayon r en dimension n vérifie*

$$\mathrm{vol}\left(\mathcal{B}_n(\mathbf{0}, r)\right) \geq \left(\frac{2r}{\sqrt{n}}\right)^n. \tag{2.19}$$

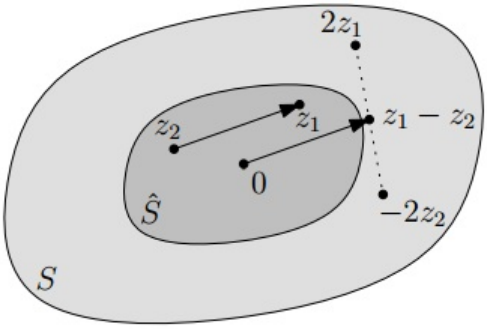


FIGURE 2.6 – Théorème du corps convexe de Minkowski

Démonstration. Il suffit de remarquer que cette boule contient le cube $\left\{\mathbf{x} \in \mathbb{R}^n \mid \forall i \in \{1,\dots,n\}, |x_i| < \frac{r}{\sqrt{n}}\right\}$. En effet soit $\mathbf{x}$ un tel élément, alors $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^n x_i^2} < \sqrt{\sum_{i=1}^n (\frac{r}{\sqrt{n}})^2} = \sqrt{\sum_{i=1}^n \frac{r^2}{n}} = \sqrt{n \cdot \frac{r^2}{n}} = \sqrt{r^2} = r$ car $r > 0$. De plus $\mathrm{vol}(\{\mathbf{x} \in \mathbb{R}^n \mid \forall i \in \{1,\dots,n\}, |x_i| < \frac{r}{\sqrt{n}}\}) = \mathrm{vol}(\{\mathbf{x} \in \mathbb{R}^n \mid \forall i \in \{1,\dots,n\}, |x_i| = \frac{r}{\sqrt{n}}\}) = (\frac{2r}{\sqrt{n}})^n$, car $|x_i| < \frac{r}{\sqrt{n}}$ et donc $-\frac{r}{\sqrt{n}} < |x_i| < \frac{r}{\sqrt{n}}$, et cet intervalle a pour longueur $\frac{r}{\sqrt{n}} - (-\frac{r}{\sqrt{n}}) = \frac{2r}{\sqrt{n}}$ et ce pour chaque $i \in \{1,\dots,n\}$. □

Le premier théorème de Minkowski est une conséquence de cette propriété :

Théorème 2.6 (1^{er} théorème de Minkowski)**.** *Pour tout réseau $\mathcal{L} \in \mathbb{R}^n$ de rang plein, $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \sqrt[n]{\det(\mathcal{L})}$.*

Démonstration. Par définition, $\mathcal{B}_n(\mathbf{0}, \lambda_1(\mathcal{L}))$ est un ensemble convexe centré en l'origine qui ne contient aucun point de $\mathcal{L}$. De plus,

$$\left(\frac{2\lambda_1(\mathcal{L})}{\sqrt{n}}\right)^n \leq \mathrm{vol}\left(\mathcal{B}_n(\mathbf{0}, \lambda_1(\mathcal{L}))\right) \leq 2^n \det(\mathcal{L}) \tag{2.20}$$

d'après le théorème 2.4 et la proposition 2.5. Donc $2^n \cdot \lambda_1(\mathcal{L})^n \leq 2^n \cdot \sqrt{n}^n \det(\mathcal{L})$ d'où $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \sqrt[n]{\det(\mathcal{L})}$ □

Le second théorème de Minkowski renforce la borne supérieure obtenue par le théorème 2.6 précédent :

Théorème 2.7 (2^{nd} théorème de Minkowski)**.** *Pour tout réseau $\mathcal{L} \in \mathbb{R}^n$ de rang plein,*

$$\sqrt[n]{\prod_{i=1}^n \lambda_i(\mathcal{L})} \leq \sqrt{n} \cdot \sqrt[n]{\det(\mathcal{L})}. \tag{2.21}$$

Démonstration. Soient $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{L}$ des vecteurs linéairement indépendants atteignant les minima successifs : $\|\mathbf{x}_i\| = \lambda_i$ et $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n$ leur orthogonalisation de Gram-Schmidt. Soit

$$T = \left\{\mathbf{y} \in \mathbb{R}^n \mid \sum_{i=1}^n \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{x}}_i \rangle}{\|\tilde{\mathbf{x}}_i\| \cdot \lambda_i}\right)^2 < 1\right\}. \tag{2.22}$$

Nous allons montrer qu'alors T ne contient aucun vecteur non-nul de $\mathcal{L}$. Soit $\mathbf{y} \in \mathcal{L} \setminus \{\mathbf{0}\}$ et soit $1 \leq k \leq n$ le plus petit entier tel que $\|\mathbf{y}\| \geq \lambda_k$. Alors $\mathbf{y} \in \mathrm{Vect}(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_k)$, sinon

$\mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{y}$ forme $k+1$ vecteurs linéairement indépendants de longueur inférieure à λ_{k+1}. Alors,

$$\sum_{i=1}^{n} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{x}}_i \rangle}{\|\tilde{\mathbf{x}}_i\| \cdot \lambda_i} \right)^2 = \sum_{i=1}^{k} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{x}}_i \rangle}{\|\tilde{\mathbf{x}}_i\| \cdot \lambda_i} \right)^2 \geq \frac{1}{\lambda_k^2} \cdot \sum_{i=1}^{k} \left(\frac{\langle \mathbf{y}, \tilde{\mathbf{x}}_i \rangle}{\|\tilde{\mathbf{x}}_i\|} \right)^2 = \frac{\|\mathbf{y}\|^2}{\lambda_k^2} \geq 1. \tag{2.23}$$

et donc $\mathbf{y} \notin T$. D'après le théorème du corps convexe, $\mathrm{vol}(T) \leq 2^n \cdot \det(\mathcal{L})$, cependant

$$\mathrm{vol}(T) = \left(\prod_{i=1}^{n} \lambda_i \right) \mathrm{vol}\left(\mathcal{B}(0^n, 1) \right) \geq \left(\prod_{i=1}^{n} \lambda_i \right) \left(\frac{2}{\sqrt{n}} \right)^n. \tag{2.24}$$

Donc $\left(\prod_{i=1}^{n} \lambda_i\right) \left(\frac{2}{\sqrt{n}}\right)^n \leq 2^n \det(\mathcal{L})$, d'où

$$\sqrt[n]{\prod_{i=1}^{n} \lambda_i(\mathcal{L})} \leq \sqrt{n} \cdot \sqrt[n]{\det(\mathcal{L})} \tag{2.25}$$

□

2.2.2 Problèmes sur les réseaux

Un des atouts majeurs de la cryptographie basée sur les réseaux est dû au travail séminal d'Ajtaï [Ajt96], établissant une connexion entre certains problèmes dans le cas-moyen et des problèmes sur les réseaux dans le pire-cas. Pour rappel, lorsqu'on dit qu'un problème est NP-difficile, on fait généralement référence aux instances les plus difficiles à résoudre de ce problème. Cependant pour une utilisation cryptographique, générer une telle instance peut-être compliqué, voir calculatoirement infaisable. Ainsi, pouvoir réduire la résolution d'un problème dans le pire-cas à celle d'un problème dans le cas-moyen est un avantage considérable pour la génération des instances, et donc des clés cryptographiques. La cryptographie basée sur les réseaux repose notamment sur la conjecture de Micciancio et Regev [MR09].

Conjecture 1. *Il n'existe aucun algorithme en temps polynomial qui puisse approximer jusqu'à un facteur polynomial les problèmes basés sur les réseaux.*

Dans cette section, nous présentons les problèmes dans le pire-cas ainsi que quelques problèmes dans le cas-moyen auxquels ces premiers peuvent se réduire.

2.2.2.1 Problèmes dans le pire-cas

Shortest Vector Problem (SVP). Le problème du vecteur le plus court dans un réseau est probablement le plus connu et le plus étudié de tous les problèmes sur les réseaux. La sécurité des cryptosystèmes actuels possédant une réduction s'y ramène de manière quasi-systématique.

Définition 2.17 (SVP_γ). *Étant donnés une base* $\mathbf{B}$ *d'un réseau* $\mathcal{L}$ *et un facteur d'approximation* $\gamma \in \mathbb{R}, \gamma \geq 1$*, le problème* (Search-)SVP_γ *consiste à trouver un vecteur non-nul* $\mathbf{v} \in \mathcal{L}$ *tel que* $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.

Shortest Independant Vector Problem (SIVP). Ce problème est relativement similaire à SVP, mis à part qu'ici on ne demande plus de trouver un vecteur court (éventuellement le plus court), mais plusieurs vecteurs courts, et linéairement indépendants :

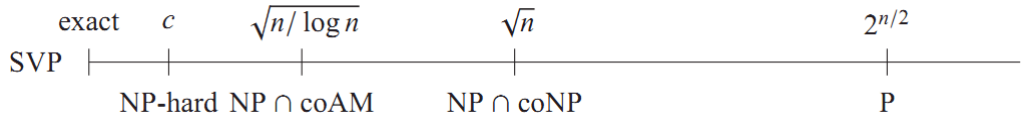


FIGURE 2.7 – Difficulté du problème du vecteur le plus court

Définition 2.18 (SIVP_γ). *Étant donnés une base* $\mathbf{B}$ *d'un réseau* $\mathcal{L}$ *et un facteur d'approximation* $\gamma \in \mathbb{R}, \gamma \geq 1$*, le problème* (Search-)SIVP_γ *consiste à trouver* n *vecteurs* $\mathbf{v}_i$ *non-nuls et linéairement indépendant tels que* $\mathbf{v}_i \in \mathcal{L}$ *et* $\|\mathbf{v}\|_i \leq \gamma \cdot \lambda_i(\mathcal{L})$.

Closest Vector Problem (CVP). Le problème du vecteur le plus proche est lui aussi relativement bien étudié, et en interaction forte avec le problème SVP. Nous adopterons la définition suivante :

Définition 2.19 (CVP_γ). *Étant donnés une base* $\mathbf{B}$ *d'un réseau* $\mathcal{L}$*, un vecteur cible* $\mathbf{t} \in \text{Vect}(\mathbf{B})$ *et* $\gamma \in \mathbb{R}, \gamma \geq 1$*, le problème* (Search-)CVP_γ *consiste à trouver un vecteur* $\mathbf{v} \in \mathcal{L}$ *tel que* $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$

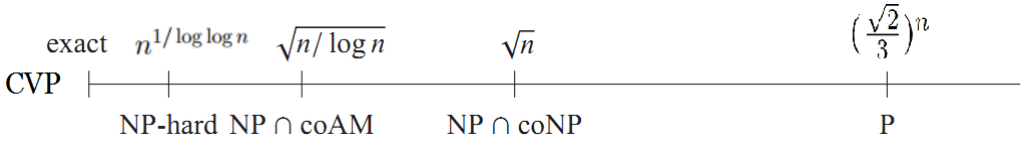


FIGURE 2.8 – Difficulté du problème du vecteur le plus proche

Absolute/Bounded Distance Decoding. Bien que ces deux problèmes soient intitulés différemment, il ne s'agît que d'instances particulières de CVP_γ :

— Bounded Distance Decoding (BDD) = $\text{CVP}_{1 \leq \gamma < \lambda_1/2}$
— Absolute Distance Decoding (ADD) = $\text{CVP}_{\gamma \geq \lambda_1}$

Ces deux valeurs ont une signification particulière ; dans BDD, si une solution existe, alors elle est unique, alors que dans ADD, l'existence d'une solution est assurée, mais pas son unicité...

Les versions exactes des problèmes décrits ci-dessus sont obtenues en prenant comme paramètre d'approximation $\gamma = 1$. Selon l'ordre de grandeur du paramètre γ, le problème SVP_γ repose dans différentes classes de complexité. Par ailleurs, Goldreich *et al.* [GMSS99] ont montré qu'approximer SVP n'était pas plus dur que d'approximer CVP.

2.2.2.2 Problèmes dans le cas-moyen

Les deux problèmes que nous présentons dans cette section sont ceux sur lesquels reposent la majorité des primitives cryptographiques modernes basées sur les réseaux. Le premier problème sert de brique de base aux signatures digitales tandis que le second sert au chiffrement. Nous décrirons les réductions dans la prochaine section.

La plupart des constructions cryptographiques avec une connexion au pire-cas de problèmes sur les réseaux suivent les traces des travaux d'Ajtaï, et utilisent des réseaux différents appelés "réseaux q-aires", où l'appartenance d'un vecteur au réseau est entièrement déterminée par sa réduction modulo q. Ces réseaux sont définis comme suit :

Définition 2.20 (Réseau q-aire)**.**

$$\begin{aligned}\Lambda_q^{\perp}(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}_q^m, \mathbf{Ax} = \mathbf{0} \pmod q\} \\ \Lambda_q(\mathbf{A}) &= \{\mathbf{y} = \mathbf{x}^{\top}\mathbf{A} \pmod q \in \mathbb{Z}_q^m, \mathbf{x} \in \mathbb{Z}_q^n\}\end{aligned} \tag{2.26}$$

Les schémas de signature basés sur SIS (voir ci-dessous) font également appel à des co-ensembles de $\Lambda_q^{\perp}(\mathbf{A})$. Pour tout $\mathbf{t} \in \mathbb{Z}_q^m$ et $\mathbf{u} \in \mathbb{Z}_q^n$ tels que $\mathbf{At} = \mathbf{u}$, on définit le réseau translaté

$$\Lambda_{q,\mathbf{u}}^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}_q^m, \mathbf{Ax} = \mathbf{u} \pmod q\} = \Lambda_q^{\perp}(\mathbf{A}) + \mathbf{t}. \tag{2.27}$$

Small Integer Solution (SIS). Le problème SIS consiste à trouver une solution non triviale à une équation linéaire modulo un entier, de sorte que les coefficients de cette solution ne soient pas trop grands.

Définition 2.21. *Pour tout $k \geq 0$, une instance du problème $k - SIS_{q,m,\beta}$ consiste en une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ et k vecteurs $\mathbf{e_1}, \ldots, \mathbf{e_k} \in \Lambda_q^{\perp}(\mathbf{A}) \subseteq \mathbb{Z}^m$ vérifiant $||\mathbf{e_i}|| \leq \beta,\ \forall\ 1 \leq i \leq k$. Le but est de trouver $\mathbf{v}\mathbb{Z}^m$ de sorte que :*

1. $||\mathbf{v}|| \leq \beta$
2. $\mathbf{A} \cdot \mathbf{v} = \mathbf{0}$ *mod* q, *c'est-à-dire* $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A})$ *et*
3. $\mathbf{v} \notin \mathbb{Q} - Vect(\mathbf{e_1}, \ldots, \mathbf{e_k})$

Le problème SIS correspond simplement au cas où $k = 0$.

De plus, les auteurs de [BF11] ont montré qu'un adversaire $\mathcal{A}$ qui résolvait le problème k-SIS en dimension m pouvait être utilisé pour résoudre le problème SIS en dimension $m - k$.

Learning With Erros (LWE). Si le problème SIS peut être vu comme une instance approchée du problème SVP sur le réseau $\Lambda_q^{\perp}$, le problème LWE est quant à lui un analogue du problème CVP sur le réseau Λ_q, ce qui justifie la "dualité" de ces deux problèmes.

Définition 2.22. *Pour un paramètre de sécurité λ fixé, soient $n = n(\lambda)$ une dimension entière, $f(x) = x^d + 1$, $d = 2^k$ un polynôme cyclotomique pour un $k \in \mathbb{N}$, $q = q(\lambda) \geq 2$ un module premier, $\mathcal{R} = \mathbb{Z}[x]/(f(x))$, $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$, $\chi = \chi(\lambda)$ une distribution sur $\mathcal{R}$ d'écart type σ.*
Le problème GLWE consiste à distinguer les distributions $(a_i, < a_i, s > +e) \in \mathcal{R}_q^{n+1}$ et $(a_i, r) \in \mathcal{R}_q^{n+1}$ pour $a_i, s \in \mathcal{R}_q^n$ et $b \in \mathcal{R}_q$ uniformément (ou sans perte de sécurité, $s \leftarrow \chi^n$) et $e \leftarrow \chi$.

Suivant la définition de [BGV12], le problème LWE correspond au problème GLWE instancié avec $d = 1$, alors que pour Ring-LWE, on choisit $n = 1$.

Regev a montré qu'un algorithme (éventuellement quantique) qui résolvait LWE dans le cas-moyen impliquait un algorithme quantique qui résoud les pire-cas de SIVP et gapSVP [Reg05]. Cette réduction a été "déquantumizée" récemment [BLP+13]. Lyubashevsky, Peikert et Regev ont montré qu'un algorithme qui résolvait Ring-LWE dans le cas-moyen impliquait un algorithme quantique qui résout les pire-cas de SVP [LPR10a].

Remarque 3. *Le problème SIS a été élargi pour inclure des instances où l'on possède des vecteurs solutions [BF11], ainsi qu'aux instances où la matrice en entrée est décomposée en plusieurs parties [NZZ15]. Nous proposerons une généralisation de ces différents problèmes et discuterons de sa sécurité dans le chapitre 4.*

Remarque 4. *La version binaire ($q = 2$) de LWE est communément appelée Learning Parity with Noise (LPN). Nous ferons un parallèle avec le problème du Syndrome Decoding (SD) dans le chapitre 6.*

Par analogie, et à ma connaissance, il n'existe pas d'équivalent de LPN en version anneau. Il pourrait être intéressant de créer le problème suivant :

Définition 2.23. *Pour un paramètre de sécurité λ fixé, soient $f(x) = x + 1$, $\mathcal{R}_2 = \mathbb{Z}_2[x]/(f(x))$, $\chi = \chi(\lambda)$ une distribution sur $\mathcal{R}$ d'écart type σ.*
Le problème Ring-LPN consiste à distinguer les distributions $(a_i, < a_i, s > +e) \in \mathcal{R}_2^2$ et $(a_i, r) \in \mathcal{R}_2^2$ pour $a_i, s \in \mathcal{R}_2$ et $b \in \mathcal{R}_2$ uniformément et $e \leftarrow \chi$.

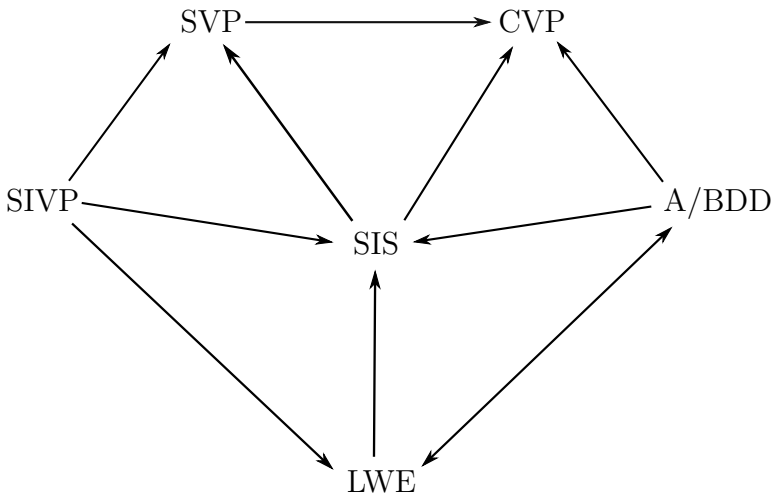


FIGURE 2.9 – Relations entre les différents problèmes sur les réseaux. Une flèche d'un problème A vers un problème B signifie qu'un algorithme qui résout le problème B permet également de résoudre le problème A (les paramètres d'approximation ont été omis).

2.2.3 Algorithmes pour la résolution de problèmes sur les réseaux

Comme nous venons de le voir, les problèmes basés sur les réseaux reposent principalement sur la difficulté de trouver un ou plusieurs vecteurs relativement courts dans un réseau à partir d'une base (apparemment) aléatoire/mauvaise, composée de vecteurs relativement longs et peu orthogonaux. Dans cette section nous décrivons les trois grandes familles [4] d'algorithmes pour la résolution des problèmes précédents.

2.2.3.1 Algorithmes d'énumération

Ces algorithmes par force-brute n'ont que peu d'intérêt utilisés seuls pour les dimensions et paramètres généralement employés dans les cryptosystèmes. Nous verrons qu'ils peuvent cependant s'avérer utiles pour améliorer la qualité des solutions fournies par la prochaine famille d'algorithmes.

4. Il existe une autre façon de résoudre de manière exacte ces problèmes : construire ce qu'on appelle la cellule de Voronoï. Nous ne présenterons pas ce type d'algorithmes dédiés aux versions exactes des problèmes, et redirigeons le lecteur intéressé vers [Vou11, Laa15].

2.2.3.2 Algorithmes de réduction par blocs

Les algorithmes de réduction de réseaux fonctionnent essentiellement de la même façon : à partir d'une mauvaise base d'un réseau, on cherche à obtenir des vecteurs de plus en plus courts et de plus en plus orthogonaux par combinaisons linéaires des vecteurs en entrée. Cette technique fut introduite en 1982 [LLL82] et généralise l'algorithme d'Euclide. De nombreuses améliorations ont été réalisées depuis

La qualité de leur sortie est mesurée par le root Hermite factor, défini comme suit :

Définition 2.24 (Root Hermite Factor)**.** *Le Root Hermite Factor d'un vecteur* $\mathbf{v}$ *du réseau* $\mathcal{L}$ *de dimension* n *est donné par*

$$\left(\frac{\|\mathbf{v}\|}{\det(\mathcal{L})^{1/n}}\right)^{1/n}. \tag{2.28}$$

Nous verrons en Sec. 2.2.3.4 comment le root Hermite factor permet d'évaluer la sécurité des primitives basées sur les réseaux.

2.2.3.3 Algorithmes de sieving

Les algorithmes de tamisage (sieving en anglais) fonctionnent essentiellement de la même façon : à partir d'une liste relativement longue de vecteurs aléatoires du réseau, construire des vecteurs plus courts par somme ou différence, et ré-itérer le procédé. Si la liste a été choisie assez longue, on peut raisonnablement espérer obtenir un vecteur relativement court du réseau [AKS01]. Les algorithmes de sieving constitue la meilleure approche pour résoudre SVP asymptotiquement en théorie, mais sont moins efficaces que les algorithmes de réductions en pratique.

2.2.3.4 Interactions, stratégie, et conséquences sur la sécurité

Les sections précédentes ont permis de mettre en évidence deux aspects caractéristiques des algorithmes pour la résolution des problèmes basés sur les réseaux :

1. Les algorithmes de réduction de réseaux par blocs sont relativement rapides, même en dimension élevée. Cependant la qualité de la solution retournée par ces algorithmes est inversement proportionnelle à leur efficacité.
2. Les algorithmes de sieving permettent de trouver des vecteurs extrêmement courts dans un réseau (voir même le plus court), mais leurs complexités à la fois en temps et en mémoire les rendent inutilisables en pratique au delà de la dimension 100 (voir même avant).

C'est pourquoi les meilleurs algorithmes pour la résolution de problèmes sur les réseaux (BKZ 2.0 notamment [CN11]) fonctionnent à l'aide de ces deux types de routines. Dans un premier temps la base est (LLL-)réduite, puis un algorithme de réduction par bloc (de sous-dimension < 100) est appelé. Chaque sous-bloc est résolu en faisant appel à un algorithme de sieving, et la solution est remontée dans l'algorithme par bloc, et ainsi de suite.

On estime à l'heure actuelle que BKZ 2.0 est le meilleur algorithme pour résoudre de tels problèmes en pratique. Les travaux de Lindner et Peikert [LP11] ont permis de faire l'analogie entre le root Hermite factor atteignable et son équivalent en sécurité symétrique, ces résultats sont reportés dans la table 2.1 suivante.

Root Hermite Factor							
Sécurité (bits)	64	80	100	112	128	196	256

TABLE 2.1 – Correspondance entre le root Hermite factor atteignable et le niveau de sécurité symétrique correspondant.

2.3 Codes Correcteurs d'Erreurs

Les codes correcteurs d'erreurs constituent un sujet de recherche actif, initié par la théorie du signal en ingénierie électrique, et perpétué par la cryptographie. En quelques mots, les codes correcteurs d'erreurs permettent comme leur nom l'indique de retrouver l'information émise après que le signal qui la transporte ait été perturbé par un bruit. Bien que les applications des codes soient multiples et variées (CD, DVD, Blu-ray, téléphonie, ordinateurs, ...), nous nous focaliserons sur leur utilisation en cryptographie.

Dans cette section, nous rappelons les principales définitions et propriétés des codes correcteurs d'erreurs linéaires. Ces définitions seront utiles et développées plus en détail dans le chapitre 6, où nous proposons un nouveau schéma de chiffrement. Nous conseillons au lecteur en quête d'approfondissement sur la théorie des codes correcteurs de se référer à [Ove07].

2.3.1 Définitions, métriques et propriétés

Lorsque Alice et Bob communiquent, il est préférable que l'information transmise par Bob coïncide avec celle reçue par Alice. Cependant, le canal utilisé pour cette communication est – comme tout canal – bruité, et chaque bit d'information a une probabilité p de se retrouver altéré. L'idée principale des codes correcteurs d'erreurs est de restreindre l'ensemble des mots que Bob peut utiliser de sorte que si Alice reçoit un mot hors de cet ensemble, mais relativement proche d'un mot valide, alors elle pourra tout de même comprendre quel mot Bob a voulu émettre. Cette restriction se fait en rajoutant de la redondance, d'une manière bien spécifique.

Nous utiliserons la définition suivante d'un code correcteur d'erreur linéaire :

Définition 3.25 (Code Linéaire)**.** *Soit $\mathcal{V}$ un espace vectoriel sur $\mathbb{R}$ de dimension n. Un* code linéaire *$\mathcal{C}$ de longueur n et de dimension k (noté $[n, k]$) est un sous-espace vectoriel de $\mathcal{V}$ de dimension k.*

En termes de vocabulaire, les éléments de $\mathcal{C}$ sont appelés les *mots du code*. En tant que sous-espace de dimension k sur $\mathcal{V}$, $\mathcal{C}$ peut être engendré par une matrice $k \times n$ à coefficients dans le corps $\mathbb{F}$, ce qui donnera lieu à la Définition 3.26. Toujours en tant que sous-espace, nous pouvons considérer l'espace dual de $\mathcal{C}$, noté $\mathcal{C}^\perp$. Alors $\mathcal{C}^\perp$ est aussi un code linéaire, de dimension $n - k$ sur $\mathcal{V}$, et peut être généré par une matrice $(n - k) \times n$ à coefficients dans $\mathbb{F}$, voir Définition 3.27.

Définition 3.26 (Matrice génératrice)**.** *On dit que $\mathbf{G} \in \mathbb{F}^{k \times n}$ est une* matrice génératrice *du code $\mathcal{C}[n, k]$ si*

$$\mathcal{C} = \left\{ \boldsymbol{\mu}\mathbf{G}, \text{ pour } \boldsymbol{\mu} \in \mathbb{F}^k \right\}. \tag{2.29}$$

Définition 3.27 (Matrice de parité)**.** *Étant donné in code $\mathcal{C}[n, k]$, on dit que $\mathbf{H} \in \mathbb{F}^{(n-k) \times n}$ est une* matrice de parité *de $\mathcal{C}$ si $\mathbf{H}$ est une matrice génératrice du code dual $\mathcal{C}^\perp$. Formellement,*

si

$$\mathcal{C}^{\perp} = \left\{ \mathbf{x} \in \mathbb{F}^n \text{ tel que } \mathbf{H}\mathbf{x}^{\top} = \mathbf{0}^{\top} \right\}. \tag{2.30}$$

Une des notions importantes en code est celle de distance minimale. C'est en quelque sorte un analogue du premier minimum en réseau. La distance minimale d'un code est définie comme suit :

Définition 3.28 (Distance minimale)**.** *Soit $\mathcal{C}[n,k]$ un code linéaire sur $\mathcal{V}$ et soit ω une norme sur $\mathcal{V}$. La* distance minimale *de $\mathcal{C}$ est définie par*

$$d = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C},\mathbf{x}\neq\mathbf{y}} \omega(\mathbf{x}-\mathbf{y}). \tag{2.31}$$

On notera par extension un tel code $[n,k,d]$, ils sont capables de décoder jusqu'à $\delta = \lfloor \frac{d-1}{2} \rfloor$ erreurs.

La cryptographie basée sur les codes souffre généralement de la taille des clés utilisées. Dans le chapitre 6, nous proposons un cryptosystème efficace, dont les clés restent de taille raisonnable grâce à la stratégie de Gaborit [Gab05], qui conduit à l'utilisation de codes Quasi-Cyclique (QC). Les codes QC sont extrêmement utiles dans un contexte cryptographique, puisque leur description compacte permet de réduire considérablement la taille des clés utilisées.

Définition 3.29 (Code quasi-cyclique [MTSB13])**.** *Un code linéaire $\mathcal{C}[n,k,d]$ sur $\mathcal{V}$ est* quasi-cyclique (QC) *d'ordre s (divisant n) si pour tout $\mathbf{c} \in \mathcal{C}$, le vecteur obtenu après rotation de s coordonnées est également un mot du code.*

Formellement, en considérant $\mathbf{c}$ comme un polynôme dans $\mathcal{R} = \mathbb{F}[X]/(X^n - 1)$, $\mathcal{C}$ est un code QC d'ordre s si $\forall \mathbf{c} \in \mathcal{C}$, on a $X^s \cdot \mathbf{c} \in \mathcal{C}$.

Le cas particulier où $s = 2$ correspond à des codes communément appelés *doublement circulants*, admettant une matrice génératrice de la forme $(\mathbf{I}_n \mid \mathbf{A})$ avec $\mathbf{A}$ une matrice circulante. De tels codes ont été utilisés pendant presque 10 ans en cryptographie [GG07] et ont prouvé leur efficacité plus récemment [MTSB13] (où des codes QC d'ordre 3 sont également considérés).

Pour tout couple d'éléments $\mathbf{x}, \mathbf{y} \in \mathcal{V}$, on définit leur produit de la même façon que dans $\mathcal{R}$, c'est-à-dire que $\mathbf{x} \cdot \mathbf{y} = \mathbf{c} \in \mathcal{V}$ avec

$$c_k = \sum_{i+j \equiv k \mod n} x_i y_j, \text{ for } k \in \{0, 1, \ldots, n-1\}. \tag{2.32}$$

On remarquera qu'en tant que produit dans l'anneau *commutatif* $\mathcal{R}$, on a $\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x}$. Nous allons étendre cette propriété en définissant les matrices circulantes.

Définition 3.30 (Matrice circulante)**.** *Soit $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$. La* matrice circulante *induite par $\mathbf{x}$ est définie et notée comme suit :*

$$\mathbf{rot}(\mathbf{x}) = \begin{pmatrix} x_1 & x_n & \ldots & x_2 \\ x_2 & x_1 & \ldots & x_3 \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_{n-1} & \ldots & x_1 \end{pmatrix} \in \mathbb{F}^{n \times n}. \tag{2.33}$$

Par conséquent, on vérifie aisément que le produit de deux éléments $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ peut être exprimé en tant que produit matrice-vecteur en utilisant l'opérateur **rot** :

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x} \cdot \mathbf{rot}(\mathbf{y})^\top = \left(\mathbf{rot}(\mathbf{x}) \cdot \mathbf{y}^\top\right)^\top = \mathbf{y} \cdot \mathbf{rot}(\mathbf{x})^\top = \mathbf{y} \cdot \mathbf{x}. \tag{2.34}$$

Nous venons de donner plusieurs définitions faisant intervenir une norme sur un espace vectoriel. Beaucoup d'analogies peuvent être faites avec les réseaux lorsque cette norme est la norme euclidienne. Dans la suite de cette section, nous nous focaliserons sur deux autres métriques propres à la théorie des codes correcteurs d'erreurs : la métrique de Hamming dans le cas binaire, et la métrique Rang lorsque nous travaillerons sur des extensions de corps finis. Bien qu'il existe d'autres métriques pour les codes correcteurs (voir [Gab12]), elles sont hors du cadre de ce mémoire.

Métrique de Hamming. Nous commençons par rappeler deux bornes essentielles en métrique de Hamming, à savoir la borne de Singleton qui donne une majoration de la distance minimale d'un code d'une part, et la borne de Gilbert-Varshamov qui en donne une minoration d'autre part.

Définition 3.31 (Borne de Singleton [Sin64])**.** *Soit $\mathcal{C}[n,k]$ un code linéaire de distance minimale d, alors $d \leq n - k + 1$.*

Définition 3.32 (Distance de Gilbert-Varshamov [Gil52, Var57])**.** *Soit d_{GV} la* distance de Gilbert-Varshamov *définie comme le plus grand des entiers d vérifiant*

$$\sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \leq q^{n-k}. \tag{2.35}$$

Alors pour tout code $\mathcal{C}[n,k]$ de distance minimale d, $d \geq d_{GV}$.

La borne de Gilbert-Varshamov garantie l'existence de codes de taux $\tau = \frac{k}{n}$ admettant pour distance minimale au moins d_{GV}. Dans le cas des codes linéaires binaires de taux $\frac{1}{2}$, on a $d_{GV} \approx 0.11n$. Il est possible de montrer qu'asymptotiquement, les codes linéaires aléatoires atteignent cette borne avec une forte probabilité.

Les paramètres obtenus pour HQC – notre cryptosystème du chapitre 6 en métrique de Hamming – sont tout à fait raisonnables, mais il est possible de les améliorer un peu plus en optant pour la métrique rang. Nous rappelons donc quelques définitions et propriétés qui font de la métrique rang un véritable atout pour notre cryptosystème, et redirigeons le lecteur vers [Loi06] pour plus de détails sur cette métrique.

Métrique Rang. Soit $\mathbb{F}$ une extension d'un corps fini, *i.e.* $\mathbb{F} = \mathbb{F}_{q^m}$ avec q premier, et soit $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ un élément d'un espace vectoriel $\mathcal{V}$ de dimension n sur $\mathbb{F}_{q^m}$. Une des propriétés élémentaires des extensions de corps finis est qu'ils peuvent être vus comme espaces vectoriels sur le corps qu'ils étendent. Ainsi, en considérant $\mathbb{F}_{q^m}$ comme un espace vectoriel de dimension m sur $\mathbb{F}_q$, et étant donnée une base $(\mathbf{e_1}, \ldots, \mathbf{e_m}) \in \left(\mathbb{F}_q^n\right)^m$, on peut exprimer chaque x_i comme

$$x_i = \sum_{j=1}^{m} x_{j,i}\mathbf{e_j} \left(\Leftrightarrow x_i = (x_{1,i}, \ldots, x_{m,i}) \right). \tag{2.36}$$

En utilisant cette formulation, on peut étendre $\mathbf{x} \in \mathbb{F}_{q^m}^n$ sous forme matricielle $\mathbf{E}(\mathbf{x})$ avec :

$$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \in \mathbb{F}_{q^m}^n \tag{2.37}$$

$$\mathbf{E}(\mathbf{x}) = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix} \in \mathbb{F}_q^{m \times n}. \tag{2.38}$$

Pour les codes en métrique Rang, la norme ω est définie comme le *rang* de la matrice obtenue Eq. (2.38). Dans la suite $\mathcal{C}$ désignera un code en métrique rang, de longueur n et de dimension k sur $\mathbb{F}_{q^m}$, où $q = p^\eta$ pour un premier p et un entier $\eta > 1$. La matrice $\mathbf{G}$ de taille $k \times n$ sera la matrice génératrice de $\mathcal{C}$ et $\mathbf{H}$ sa matrice de parité. Le code $\mathcal{C}$ est un sous-espace de dimension k de $\mathbb{F}_{q^m}$ muni de la métrique rang. La *distance rang minimale* du code $\mathcal{C}$ est le plus petit rang de tous les mots du code non-nuls. On considère également le produit scalaire usuel qui permet de définir le code dual.

Une des notions importantes en métrique rang différant de la métrique de Hamming est celle de support. Soit $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{F}_{q^m}^n$ un vecteur de rang r. On note $E = \langle x_1, \dots, x_n \rangle$ le $\mathbb{F}_q$-sous-espace de $\mathbb{F}_{q^m}$ généré par les coordonnées de $\mathbf{x}$, *i.e.* $E = \text{Vect}\,(x_1, \dots, x_n)$. L'espace vectoriel E est appelé le *support* de $\mathbf{x}$ et noté $\text{Supp}(\mathbf{x})$. Enfin, la notion d'*isométrie* qui, en métrique de Hamming correspond à l'action d'un code sur les matrices $n \times n$ de permutations, est remplacée en métrique rang par l'action du code sur les matrice $n \times n$ inversibles sur le corps de base $\mathbb{F}_q$.

Bornes en métrique rang. Les bornes classiques en métrique de Hamming ont des analogues assez directs en métrique rang. La borne de Singleton pour un code linéaire $[n, k]$ de rang r sur $\mathbb{F}_{q^m}$ s'applique naturellement en métrique rang, et fonctionne de manière similaire à la métrique de Hamming (en trouvant des ensembles d'information) : $r \leq 1 + n - k$. Lorsque $n > m$ cette borne peut être ré-écrite [Loi06]

$$r \leq 1 + \left\lfloor \frac{(n-k)m}{n} \right\rfloor. \tag{2.39}$$

Les codes atteignant cette borne sont appelés Maximum Rank Distance (MRD).

Décodage déterministe. À la différence des codes en métrique de Hamming, il y a peu de familles de codes capables de décoder des erreurs jusqu'à un poids donné de manière déterministe. Essentiellement, seuls les codes de Gabidulin [Gab85] peuvent le faire. Ces codes sont l'analogue en métrique rang des codes de Reed-Solomon [RS60], où les polynômes sont remplacés par des q-polynômes. Ces codes sont définis sur $\mathbb{F}_{q^m}$ et pour $k \leq n \leq m$, les codes de Gabidulin de longueur n et dimension k sont optimaux et satisfont la borne de Singleton pour $m = n$ et $d = n - k + 1$. Ils peuvent décoder jusqu'à $\lfloor \frac{n-k}{2} \rfloor$ erreurs en métrique rang de manière déterministe.

Décodage probabiliste. Il existe également une famille de codes simples qui a été décrite pour la métrique du sous-espace [SKK10] et qui s'adapte de manière directe à la métrique rang. Ces codes atteignent asymptotiquement l'équivalent de la borne de Gilbert-Varshamov en métrique rang. Cependant, leur probabilité non-négligeable d'erreur de décodage rend ces codes moins intéressants dans le contexte et les paramètres (q petit) du schéma de chiffrement en métrique rang (RQC) présenté chapitre 6.

2.3.2 Problèmes sur les codes

Dans cette section sont décrits les problèmes relatifs à la cryptographie basée sur les codes. Les définitions ci-dessous seront données de manière générique et bien qu'elles soient généralement instanciées en métrique de Hamming, elles s'adaptent en métrique rang sans perte de généralité. Enfin, nous aborderons la complexité de ces problèmes.

Définition 3.33 (Distribution SD)**.** *Pour n, k, et w des entiers positifs, la* Distribution $\mathsf{SD}(n, k, w)$ *consiste à choisir* $\mathbf{H} \xleftarrow{\$} \mathbb{F}^{(n-k)\times n}$ *et* $\mathbf{x} \xleftarrow{\$} \mathbb{F}^n$ *tel que* $\omega(\mathbf{x}) = w$*, et retourner* $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.

Définition 3.34 (Search SD)**.** *Soit ω une norme sur $\mathcal{V}$. Étant donné* $(\mathbf{H}, \mathbf{y}^\top) \in \mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)}$ *issu de la distribution* SD*, le problème du* Search-$\mathsf{SD}(n, k, w)$ *consiste à trouver* $\mathbf{x} \in \mathbb{F}^n$ *tel que* $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$*, avec* $\omega(\mathbf{x}) = w$.

En fonction de la métrique utilisée, le problème ci-dessus sera noté SD pour la métrique de Hamming ou Rank-SD (RSD) pour la métrique rang.

De manière équivalente, le problème SD peut être formulé comme un problème de décodage. Pour la métrique de Hamming, le problème SD a été prouvé NP-complet [BMvT78]. Ce problème peut également être vu comme problème LPN avec un nombre fixé d'échantillons [AIK07]. Le problème RSD a récemment été prouvé difficile avec une réduction probabiliste à la métrique de Hamming [GZ14]. Pour des fins cryptographiques, nous aurons besoin d'une version décisionnelle de ce problème, donnée par la Définition suivante :

Définition 3.35 (Decisional SD)**.** *Étant donné* $(\mathbf{H}, \mathbf{y}^\top) \xleftarrow{\$} \mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)}$*, le problème* décisionnel-SD (n, k, w) (DSD) *consiste à décider avec un avantage non-négligeable si* $(\mathbf{H}, \mathbf{y}^\top)$ *provient de la distribution* $\mathsf{SD}(n, k, w)$ *ou de la distribution uniforme sur* $\mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)}$.

Comme nous l'avons abordé plus haut, ce problème peut être reformulé en problème de décodage, en tant que problème LPN avec un nombre fixe d'échantillons et sous cette forme, le problème DSD a été montré polynomialement équivalent à sa version Search [AIK07]. La version en métrique rang de ce problème est notée DRSD ; en appliquant la transformation décrite dans [GZ14], il est possible de montrer que ce problème se réduit à un problème Search en métrique de Hamming. Ainsi, même si la réduction n'est pas optimale, elle apporte de la confiance en la difficulté du problème.

Enfin, comme le schéma que nous présenterons chapitre 6 utilise des codes QC pour les deux métriques présentées, nous définissons explicitement le problème sur lequel repose notre construction. Le Définition ci-après décrit donc le problème DSD dans la configuration QC, et combine les Définitions 3.29 et 3.35.

Définition 3.36 (Distribution s-QCSD)**.** *Soient n, k, w et s des entiers positifs. La distribution* s-$\mathsf{QCSD}(n, k, w, s)$ *consiste à choisir uniformément aléatoirement une matrice de parité* $\mathbf{H} \xleftarrow{\$} \mathbb{F}^{(n-k)\times n}$ *d'un code QC* $\mathcal{C}$ *d'ordre s et* $\mathbf{x} \xleftarrow{\$} \mathbb{F}^n$ *tel que* $\omega(\mathbf{x}) = w$*, et à retourner* $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.

Définition 3.37 (Problème (Search) s-QCSD)**.** *Soient n, k, w et s des entiers positifs,* $\mathbf{H}$ *une matrice de parité aléatoire d'un code QC* $\mathcal{C}$ *et* $\mathbf{y} \xleftarrow{\$} \mathbb{F}^{n-k}$*. Le problème* Search s-$\mathsf{QCSD}(n, k, w)$ *consiste à trouver* $\mathbf{x} \in \mathbb{F}^n$ *de poids* $\omega(\mathbf{x}) \leq \delta$ *tel que* $\mathbf{y} = \mathbf{x}\mathbf{H}^\top$.

Bien qu'il n'existe pas de résultat général de complexité pour les codes QC, le décodage de tels codes est considéré comme difficile par la communauté. Il existe des attaques génériques

qui utilisent la structure cyclique d'un code [Sen11, HT15], mais ces attaques n'ont qu'un impact très limité sur la complexité de ces problèmes, de sorte qu'en pratique, les meilleures attaques sont celles pour les codes non-circulant, jusqu'à un petit facteur d'approximation.

Le problème QCSD possède également une forme décisionnelle à laquelle nous ramènerons la sécurité de notre cryptosystème chapitre 6.

Définition 3.38 (Problème Décisionnel s-DQCSD)**.** *Soient n, k, w et s des entiers positifs, $\mathbf{H}$ une matrice de parité aléatoire d'un code QC $\mathcal{C}$ et $\mathbf{y} \xleftarrow{\$} \mathbb{F}^{n-k}$. Le problème s-DQCSD(n,k,w) consiste à décider avec un avantage non-négligeable si $(\mathbf{H}, \mathbf{y}^\top)$ provient de la distribution s-QCSD(n,k,w) ou de la distribution uniforme sur $\mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{n-k}$.*

Comme pour le problème ring-LPN, il n'existe pas à notre connaissance de réduction du problème s-DQCSD search au problème décisionnel. L'approche de [AIK07] ne peut s'adapter directement dans ce cas. Cependant, les meilleures attaques connues sur le problème décisionnel restent celles qui cherchent à résoudre le problème search. La situation est similaire pour les versions en métrique rank de ces problèmes (notés s-RQCSD et s-DRQCSD).

2.3.3 Algorithmes

La complexité en pratique du problème SD pour la métrique de Hamming a été largement étudiée depuis plus de 50 ans. Pour des poids relativement petits [5], les meilleures attaques ont une complexité exponentielle en le poids du mot du code recherché [BJMM12].

Le problème RSD est moins connu, sans être pour autant complètement nouveau aux yeux de la communauté cryptographique puisque le premier cryptosystème en métrique rang fut proposé en 1991 [GPT91]. Nous discutons maintenant de la complexité du problème RSD.

Pour la métrique rang, la complexité des attaques croît très vite avec la taille des paramètres, pour une raison assez fondamentale : les attaques en métrique de Hamming se ramènent à compter le nombre de mots de poids t parmi tous ceux de longueur n, nombre donné par le coefficient de Newton $\binom{n}{t}$, qui est exponentiel et majoré par 2^n. Pour la métrique rang, énumérer les supports de taille r pour un code $\mathcal{C}[n,k]$ sur $\mathbb{F}_{q^m}$ revient à compter le nombre de sous-espaces de dimension r parmi ceux de dimension n. Ce nombre est cette fois-ci donné par le coefficient de Gauss $\begin{bmatrix} m \\ r \end{bmatrix}_q$ qui est aussi exponentiel mais d'ordre $q^{m(m-r)}$, c'est-à-dire avec un terme quadratique dans l'exposant.

Il existe deux catégories d'attaques pour les cryptosystèmes en métrique rang :

— **Combinatoires** : ce sont généralement les meilleures attaques pour de petites valeurs de q (typiquement $q = 2$) et lorsque n et k ne sont pas trop petits, lorsque q augmente, l'aspect combinatoire de ces attaques les rend moins efficaces. La meilleure attaque combinatoire a récemment été améliorée en $(n-k)^3 m^3 q^{(r-1)\lfloor \frac{(k+1)m}{n} \rfloor}$, en prenant en compte le paramètre n [GRS16].

— **Algébriques** : la nature particulière de la métrique rang en fait un domaine naturel pour les attaques algébriques utilisant les bases de Gröbner, puisque ces attaques sont complètement indépendantes de la valeur de q et parfois même du degré m de l'extension. Ces attaques s'avèrent les meilleures en termes d'efficacité surtout lorsque q augmente. Dans les cas que nous considérerons au chapitre 6 où q est relativement petit (2 ou 4), le coût de ce type d'attaque est amplement supérieur à celui des attaques combinatoires (voir [LdVP06, FLDVP08, GRS16]).

5. Légèrement plus petits que la borne de Gilbert-Varshamov

L'efficacité de ces attaques dépend donc étroitement des paramètres choisis (une discussion étendue sur ce sujet peut être trouvée dans [GRS16]). Globalement, les attaques combinatoires sont les plus efficaces pour des petits alphabets, et les attaques algébriques reprennent progressivement l'avantage au fur et à mesure que l'alphabet grossit.

Une question reste en suspens quant aux améliorations potentielles des attaques génériques pour la métriques rang. La structure particulière du support d'un mot de code pour la métrique rang (où le support d'un mot n'est pas relié directement à ses coordonnées comme pour la métrique de Hamming) semble montrer que des attaques basées sur le paradoxe des anniversaires seraient difficiles à obtenir. Cette difficulté supplémentaire apporte d'autant plus de confiance en la métrique rang que les récentes améliorations pour décoder des codes aléatoires en métrique de Hamming utilisent des attaques par paradoxes des anniversaires améliorées. En quelque sorte, la métrique rang serait immune à ce type d'améliorations.

2.3.4 Principaux schémas de chiffrement

Dans cette section, nous présentons trois des schémas de chiffrement les plus employés pour les métriques considérées dans ce mémoire, à savoir les cryptosystèmes de McEliece [McE78] et MDPC [MTSB13] pour la métrique de Hamming, ainsi que le cryptosystème LRPC [GMRZ13] pour la métrique rang. Ces cryptosystèmes seront comparés à celui que nous proposons dans le chapitre 6 à cet endroit.

2.3.4.1 McEliece

Dans la course au développement de primitives de chiffrement asymétrique McEliece a rapidement proposé après RSA le premier cryptosystème à clé publique basé sur les codes. Dans sa version originale, ce dernier propose d'utiliser comme famille de codes les Goppa binaires. Il s'est avéré que toutes les autres instanciations utilisant des familles différentes ont été cassées.

— $\mathsf{KeyGen}(1^\lambda)$: génère un code $\mathcal{C}$ de longueur $n(\lambda)$ et de dimension $k(\lambda)$ avec un algorithme $\mathcal{D}$ de décodage efficace pour $\mathcal{C}$ capable de corriger jusqu'à t erreurs, ainsi que sa matrice génératrice $\mathbf{G} \in \mathbb{F}_q^{k\times n}$. Soient $\mathbf{S} \xleftarrow{\$} \mathbb{F}_q^{k\times k}$ une matrice inversible et $\mathbf{P} \xleftarrow{\$} \mathbb{F}_2^{n\times n}$ une matrice de permutation. L'algorithme retourne $\mathsf{pk} = (\tilde{\mathbf{G}} = \mathbf{SGP}, t)$ et $\mathsf{sk} = (\mathbf{S}, \mathbf{G}, \mathbf{P})$

— $\mathsf{Encrypt}(\mathsf{pk}, \mu)$: pour chiffrer $\mu \in \mathbb{F}_q^k$, génère $\mathbf{e} \xleftarrow{\$} \mathbb{F}_q^n$ tel que $\omega(\mathbf{e}) = t$, et retourne $\mathbf{c} = \mu\tilde{\mathbf{G}} + \mathbf{e}$.

— $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: calculer $\tilde{\mathbf{c}} = \mathbf{cP}^{-1}$ et utiliser l'algorithme de décodage pour trouver $\tilde{\mu} = \mathcal{D}(\tilde{\mathbf{c}})$. Retourner $\tilde{\mu}\mathbf{S}^{-1}$.

En 1986, Niederreiter a proposé une version duale du cryptosystème de McEliece [Nie86], permettant de réduire légèrement la taille des clés utilisées. Nous n'aborderons pas plus en détail ce schéma.

2.3.4.2 MDPC

En 2012, Barreto *et al.* ont proposé deux variantes de McEliece, utilisant des codes Moderate Density Parity-Check (MDPC) et QC-MDPC. Afin de ne pas surcharger les notations, nous désignerons simplement par MDPC l'instanciation QC de leur cryptosystème, qui est par ailleurs la plus efficace.

— $\mathsf{KeyGen}(1^\lambda)$: génère une matrice $\mathbf{H} \in \mathbb{F}_2^{k\times n}$ de parité d'un code $\mathcal{C}$ MDPC pour lequel un algorithme $\mathcal{D}_{\mathbf{H}}$ permettant de décoder jusqu'à t erreurs est connu, ainsi la matrice génératrice $\mathbf{G} \in \mathbb{F}_2^{(n-k)\times n}$ de ce code sous forme échelonnée réduite. Retourne $\mathsf{pk} = (\mathbf{G}, t)$ et $\mathsf{sk} = \mathbf{H}$.

— $\mathsf{Encrypt}(\mathsf{pk}, \mu)$: pour chiffrer $\mu \in \mathbb{F}_2^{n-k}$, génère $\mathbf{e} \xleftarrow{\$} \mathbb{F}_2^n$ tel que $\omega(\mathbf{e}) \leq t$, et retourne $\mathbf{c} = \mu\mathbf{G} + \mathbf{e}$.

— $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: calculer $\tilde{mu} = \mu\mathbf{G} \leftarrow \mathcal{D}_{\mathbf{H}}(\mu\mathbf{G} + \mathbf{e})$ et extraire μ à partir des $n-k$ premières positions de $\tilde{\mu}$.

Malgré des tailles de clés tout à fait raisonnables, ce schéma souffre d'une probabilité d'échec de déchiffrement non-négligeable, ce qui a récemment conduit à des attaques pratiques [GJS16]. Les trois solutions proposées par les auteurs pour remédier à ce problème sont :

1. Choisir les paramètres de manière conservatrice : augmenter le nombre d'erreurs que le code peut corriger de sorte à ce que cette probabilité devienne négligeable. Cela résulte malheureusement en des clés qui perdent leur avantage de taille,
2. Passer à des algorithmes de décodages plus élaborés : qui permettraient de décoder plus d'erreurs. Cela a un impact significatif sur les performances du cryptosystème,
3. Autoriser à re-chiffrer : dans certaines applications, la configuration peut le permettre, mais il faut alors rajouter des hypothèses de sécurité.

En résumé, aucune de ces contre-mesures n'apparaît acceptable à la fois en termes de sécurité, d'efficacité, et de performance. Le schéma que nous proposons dans le chapitre 6 présente une analyse détaillée de la probabilité d'erreur de déchiffrement, et grâce à sa meilleure complexité asymptotique, il est possible d'augmenter les paramètres du cryptosystème pour diminuer arbitrairement cette probabilité tout en restant efficace. En d'autres termes, la solution *1.* devient envisageable.

2.3.4.3 LRPC

En 2013, Gaborit *et al.* ont proposé d'utiliser des codes en métrique rang à faible distance pour leur cryptosystème : les codes Low Rank Parity-Check (LRPC). Dans sa version originale, ce schéma est également présenté avec une instance QC permettant de réduire la taille des clés.

— $\mathsf{KeyGen}(1^\lambda)$: génère un code LRPC $\mathcal{C}$ de support $\mathcal{S}$ de petit rang r, de matrice de parité $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ et de matrice génératrice $\mathbf{G} \in \mathbb{F}_{q^m}^{k\times n}$. Retourne $\mathsf{pk} = (\mathbf{G}, r)$ et $\mathsf{sk} = \mathbf{H}$

— $\mathsf{Encrypt}(\mathsf{pk}, \mu)$: pour chiffrer $\mathbf{x} \in \mathbb{F}_{q^m}^k$, génère $\mathbf{e} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ de rang $\omega(\mathbf{e}) \leq r$ et retourne $\mathbf{c} = \mu\mathbf{G} + \mathbf{e}$.

— $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: calculer $\mathbf{s} = \mathbf{H}\mathbf{c}^\top$, et retrouver l'erreur $\mathbf{e}$ en décodant $\mathbf{s}$. Calculer $\mu\mathbf{G} = \mathbf{c} - \mathbf{e}$ puis retrouver et retourner μ.

Deuxième partie

Cryptographie basée sur les Réseaux Euclidiens

Chapitre 3

Réparation Pratique de NTRUSign

Sommaire

Contexte

Les premières tentatives pour obtenir des schémas de signature basés sur les réseaux étaient étroitement liées à la réduction de vecteurs modulo le parallélépipède fondamental de la base secrète (comme GGH [GGH97], ou NTRUSign [HHGP+03]). Cette approche laissait fuir de l'information sur le secret utilisé à savoir la forme du parallélépipède, qui a pu être exploitée par des attaques pratiques [NR06]. NTRUSign était un schéma extrêmement efficace, et beaucoup d'intérêt a été porté à de possibles contre-mesures, avec cependant peu de succès [DN12].

Dans [GPV08], les auteurs proposent une version randomisée de l'algorithme du plan le plus proche de Babaï, de sorte que la distribution du vecteur réduit modulo le parallélépipède secret ne dépend que de la taille de la base utilisée. En utilisant ce nouvel algorithme, et en générant de larges clés, proches de l'uniforme, les auteurs parviennent à obtenir des signatures de type GGH basées sur les réseaux et prouvées sûres. Récemment, Stehlé et Steinfeld ont également obtenu un schéma très proche de NTRUSign (d'un point de vue théorique) et prouvé sûr [SS13].

Dans ce chapitre, nous présentons une approche alternative pour combler la fuite d'information de NTRUSign. Plutôt que de modifier les réseaux et les algorithmes utilisés, nous utilisons une signature NTRUSign classique laissant fuir de l'information, et masquons cette fuite en utilisant un bruit gaussien, comme pour la signature de Lyubashevsky [Lyu12]. Nos principales contributions sont donc un ensemble de paramètres forts pour NTRUSign, obtenu

en prenant en compte les dernières attaques connues, un moyen statistique de masquer la fuite d'information des signatures classiques NTRUSign, de sorte que cette instance particulière de signature basée sur le problème CVP résiste aux forgeries, sous l'hypothèse que le problème $\tilde{\mathcal{O}}(N^{1.5})$-SIVP sur les réseaux de type NTRU soit dur dans le pire-cas. Enfin, nous donnons des paramètres concrets pour évaluer les performances du schéma de signature résultant.

Ce travail est issu d'une collaboration avec Carlos Aguilar Melchor, Xavier Boyen, et Philippe Gaborit, et a été publié dans la conférence internationale PQCrypto 2014 [Mos14].

3.1 Introduction

Lattice based cryptography has met growing interest since the seminal work of Ajtai [Ajt96] which introduced the so called worst-case to average-case reductions. Based upon this work, a long list of cryptographic primitives such as One Way Functions, Collision-Resistant Hash Functions, Digital Signatures, or Identification schemes have been revisited to provide more confidence about security. The most efficient known digital signature scheme provably secure is BLISS [DDLL13] [1] which leads to signatures of about 5kb [2] for a security level of 128 bits.

Digital signatures have shown great promise since 1997, when was introduced GGH [GGH97]. The most famous particular instantiation of GGH is NTRUSign, which uses convolution modular lattices. The particularity of those schemes is their lack of strong worst-case to average-case security reductions, but they offer amazing performances regarding classical schemes based on number theory or discrete logarithm. For instance, for a 128 bit security level, a NTRUSign signature would be only 1784 bits long (see [HHGPW09]).

NTRUSign, first known as NSS [HPS01], was first introduced at EuroCrypt'01 by Hoffstein, Pipher and Silverman. It was amazingly fast and benefited from small keys due to the cyclic structure of the underlying convolution modular lattices that were used. The authors were aware that their scheme was vulnerable to transcript attacks *i.e.* wasn't zero-knowledge, but unfortunately they overestimated its length, and Nguyen and Regev succeeded in breaking the scheme in 2006 [NR06] by a nice gradient descent over the even moment polynomials. Initially, their attack required about 100.000 NTRU signatures to recover the hidden parallelepiped that reveals the secret basis, but still due to the cyclic structure of convolution modular lattices, they were able to shrink this threshold to about only 400 signatures for a claimed security level of 80 bits. In order to tackle this issue, several heuristical countermeasures were proposed such as the use of perturbations [HHGP+03] and the deformation of the fundamental parallelepiped [HWH08], but none of them were capable to resist to the improved attack by Ducas and Nguyen [DN12].

3.1.1 Our contribution

We revisit NTRUSign in order to provide it with a zero-knowledge proof. Our technique is inspired from Lyubashevsky's scheme [Lyu12], where the secret key $\mathbf{S}$ consists of a matrix in $\{-d, \ldots, d\}^{m \times k}$, the message is hashed to a vector $\mathbf{c} \leftarrow \{-1, 0, 1\}^k$ such that $\|\mathbf{c}\|_1 \leq \kappa$, and the signature consists of $\mathbf{Sc}$ shifted by a mask $\mathbf{y} \xleftarrow{\$} D_\sigma^m$ where D_σ^m represents the discrete gaussian distribution in dimension m with standard deviation σ.

Instead of hiding $\mathbf{Sc}$, we get a leaky signature from NTRUSign, and then use this signature as the secret and hide it with a well chosen $\mathbf{y}$. The critical technicality resides in the choice

1. which improves [Lyu12] with a better rejection sampling
2. for the space-optimized version, see Table 3 of [DDLL13] for more details

of the dimension N and the standard deviation σ : if it was chosen too small, the secret isn't properly hidden, and our modification doesn't seal any leak, if σ is too big, so will be our signatures and our scheme loses in efficiency and practicality.

We note that unlike other provably secure signature schemes such as GPV [GPV08] or [SS11], we do not modify the initial NTRU signature scheme, except by choosing public parameters more conservatively, and thus keep its inherent size and computational efficiency. Of course masking the signature in a second step comes at a price, but we manage to get signature of size $\approx 10kb$ together with public and secret keys respectively around 7000 and 1500 kb.

We choose to hide NTRU signatures with a noise based on the assumption that the leak in the signatures is exploitable but that there are no structural attacks against the public NTRU key (and thus we suppose that sealing the leak is enough to secure the scheme). This is based on the observation that the research community has published no noticeable structural attacks on NTRU lattices in the last decade and that problems such as SIS do not seem to be easier than in random lattices (if we take into account the gap induced by the small secret key).

3.1.2 Road map

In section 3.2, we present the basic elements and notations used in NTRUSign and Lyubashevsky's signature scheme, then describe these schemes respectively in sections 3.3 and 3.4. Finally, we present the scheme we propose in section 3.5 along with its security proofs and sets of parameters.

3.2 Additional Background

Most of the necessary notions have already been introduced in the preliminaries chapter 2, but we will need several additional lemmas for our fix.

The next lemma gives us an idea of how big the standard deviation must be to ensure that the inner product of two vectors doesn't overflow a certain amount. This lemma is crucial to determine our signature size in table 3.3 with overwhelming probability.

Lemma 2.1 ([Lyu12])**.**

$$\forall \mathbf{v} \in \mathbb{R}^{2N}, \forall \sigma, r > 0, \textit{we have } Pr\left[\mid \langle \mathbf{x}, \mathbf{v} \rangle \mid > r; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N} \right] \leq 2e^{-\frac{r^2}{2\|\mathbf{v}\|^2\sigma^2}}.$$

Optimally, we will set $r = \alpha \cdot \|\mathbf{v}\|\sigma$. Table 3.1 shows how big α should be to ensure k bits of security. We also need a few more material to prove that our NTRU signature will be correctly hidden by our mask. This material is given by the following lemma.

Lemma 2.2 ([Lyu12])**.**

1. $\forall \alpha > 0, Pr\left[|x| > \alpha\sigma; x \xleftarrow{\$} D_\sigma^1 \right] \leq 2e^{-\frac{\alpha^2}{2}}$
2. $\forall \eta \geq 1, Pr\left[\|\mathbf{x}\| > \eta\sigma\sqrt{2N}; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N} \right] < \eta^{2N} e^{N(1-\eta^2))}$
3. $\forall \mathbf{x} \in \mathbb{Z}^{2N}$ *and* $\sigma \geq \frac{3}{\sqrt{2\pi}}$, *we have* $D_\sigma^{2N}(\mathbf{x}) \leq 2^{-2N}$

Security parameter k	80	100	112	128	160
Gap factor α	11	12	13	14	15

TABLE 3.1 – $\alpha = \lceil \sqrt{2(k+1)\ln(2)} \rceil$ as a function of the security level k

For our purposes, as the mask $\mathbf{y}$ is sampled from a Discrete Normal Distribution, we might have to re-sample several times before obtaining a valid signature, but still, we want our signature procedure to terminate, in a reasonable (polynomial) time. This is ensured by the next lemma, whose proof is essentially detailed in [Lyu12] for a security level of $k = 100$ bits. We extend the proof of this lemma (in appendix 3.A.1) to make it fitting better with different security levels.

Lemma 2.3 ([Lyu12] extended)**.** *For any $\mathbf{v} \in \mathbb{Z}^{2N}$ and $\sigma = \omega(\|\mathbf{v}\|_2 \sqrt{\log_2(2N)})$, we have*

$$Pr\left[D_\sigma^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \mathcal{O}(1); \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] = 1 - 2^{-\omega(\log_2(2N))}$$

and more concretely, $\forall \mathbf{v} \in \mathbb{Z}^{2N}$, if $\sigma = \alpha\|\mathbf{v}\|$ for some positive α, then

$$Pr\left[D_\sigma^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) < e^{1+1/(2\alpha^2)}; \mathbf{x} \xleftarrow{\$} D_\sigma^{2N}\right] > 1 - 2^{-k}$$

This lemma ensures us that Lyubashevsky's layer of the signing procedure will be called at most $M = e^{1+1/(2\alpha^2)}$ times with probability at least $1 - 2^{-k}$. Keeping this repetition rate down is of major importance especially as this layer involves a `NTRUSign` procedure which is itself also a loop. In table 3.3, we provide two versions of parameters for each security level. In the first one, the `NTRUSign` part is generated in only one round with overwhelming probability, before applying the rejection step with $M \approx 2.8$, leading to a speed-optimized version. In the second one, we allow the generation of the `NTRU` signature to take at most 5 rounds whilst reducing its norm. This implies more rejection steps ($M \approx 7.5$) but allows us to shrink the signature sizes by approximately 15%.

To prove the security of our scheme, we also need the rejection sampling lemma 1.1, which will be used in the proof of theorem 3.2.1 that will help us getting our security reduction to SIS. The next theorem is a direct consequence of lemmas 2.3 and 1.1 by replacing V by the subset of $\mathbb{Z}^{2N}$ of vector $\mathbf{v}$ of length at most T, f by D_σ^{2N} and g_v by $D_{\mathbf{v},\sigma}^{2N}$.

Theorem 3.2.1 ([Lyu12])**.** *Let $V = \left\{\mathbf{v} \in \mathbb{Z}^{2N}; \|\mathbf{v}\| \leq T\right\}$, $\sigma = \omega(T\sqrt{\log 2N}) \in \mathbb{R}$ and $h : V \to \mathbb{R}$ a probability distribution. Then $\exists M = \mathcal{O}(1)$ such that distributions of algorithms $\mathcal{A}$ and $\mathcal{F}$ below are within statistical distance $\frac{2^{-\omega(\log 2N)}}{M}$. Moreover, $\mathcal{A}$ outputs something with probability at least $\frac{1-2^{-\omega(\log 2N)}}{M}$*

Algorithm $\mathcal{A}$
1: $\mathbf{v} \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} g_{\mathbf{v}}$
3: *output* $(\mathbf{z}, \mathbf{v})$ *with probability* $\min\left(\frac{f(\mathbf{z})}{M g_{\mathbf{v}(\mathbf{z})}}, 1\right)$

Algorithm $\mathcal{F}$
1: $\mathbf{v} \xleftarrow{\$} h$
2: $\mathbf{z} \xleftarrow{\$} f$
3: *output* $(\mathbf{z}, \mathbf{v})$ *with probability* $1/M$

3.3 General overview of NTRUSign

In this section, we briefly describe the NTRUSign scheme. For a complete description of the scheme, we refer the reader to [HHgP+05, HHGPW09]. The basic set for NTRUSign is $\mathcal{R}_q = \mathbb{Z}_q/(X^N - 1)$ with addition and polynomial multiplication modulo $X^N - 1$, also known as convolution product and denoted by $*$:

$$(\mathbf{f} * \mathbf{g})(X) = \sum_{k=0}^{N-1} \left(\sum_{i+j \equiv k \mod N} f_i g_i \right) X^k \tag{3.1}$$

The public and private keys will be matrices $\mathbf{P}$, and $\mathbf{S}$ defined by :

$$\mathbf{P} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_0 & h_{N-1} & \dots & h_1 \\ 0 & 1 & \dots & 0 & h_1 & h_0 & \dots & h_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_{N-1} & h_{N-2} & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right) \quad \mathbf{S} = \left(\begin{array}{cccc|cccc} f_0 & f_{N-1} & \dots & f_1 & F_0 & F_{N-1} & \dots & F_1 \\ f_1 & f_0 & \dots & f_2 & F_1 & F_0 & \dots & F_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{N-1} & f_{N-2} & \dots & f_0 & F_{N-1} & F_{N-2} & \dots & F_0 \\ \hline g_0 & g_{N-1} & \dots & g_1 & G_0 & G_{N-1} & \dots & G_1 \\ g_1 & g_0 & \dots & g_2 & G_1 & G_0 & \dots & G_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{N-1} & g_{N-2} & \dots & g_0 & G_{N-1} & G_{N-2} & \dots & G_0 \end{array}\right) \tag{3.2}$$

where $\mathbf{h} = \mathbf{f}^{-1} * \mathbf{F} \mod q$ for $\mathbf{f}, \mathbf{g} \in \mathcal{R}_q^*$, $\mathbf{F}, \mathbf{G} \in \mathcal{R}_q$ are given by the *keyGen* algorithm 1, and verify $\mathbf{f} * \mathbf{G} - \mathbf{g} * \mathbf{F} = q$. As operations are performed modulo q and as $(\mathbf{F}, \mathbf{G})$ can be obtained efficiently from $(\mathbf{f}, \mathbf{g})$ using [HHGP+03], we will denote $\mathbf{P} = (\mathbf{1}\ \mathbf{h})$ and $\mathbf{S} = (\mathbf{f}\ \mathbf{F})$ for short. The NTRU lattice is :

$$\Lambda_q(\mathbf{A}) = \{(\mathbf{y}_1, \mathbf{y}_2) / \mathbf{y}_2 = \mathbf{y}_1 * \mathbf{h} \mod q\} \tag{3.3}$$

Algorithms. We now recall the algorithms used in NTRUSign. More sophisticated versions of this signature scheme have been elaborated, such as the one with perturbations [HHGP+03] or the deformation of the fundamental parallelepiped [HWH08] to counter the attack of [NR06], but all these upgrades have been broken with the later improved attack of [DN12]. Therefore, we will further use the basic instantiation of NTRUSign for our scheme, which offers greater performances together with smaller keys. We will discuss the security of our scheme in section 3.5.

Key generation and signing procedures are described respectively in algorithms 1 and 2. The NTRU signature $\mathbf{s} \in \mathcal{R}_q$ is a simple Babaï's round-off [Bab85] of the target $(\mathbf{0}, \mathbf{m})$ using the secret key sk, where $\mathbf{m} = H(\mu)$ is the hash of the message μ to be signed by $H : \{0,1\}^* \to \mathcal{R}_q$. In order to process this round-off, for any $x \in \mathbb{R}$ we will denote by $\lfloor x \rceil$ the nearest integer to x so that $\{x\} = x - \lfloor x \rceil \in (-\frac{1}{2}, \frac{1}{2}]$. By extension, for any $\mathbf{x} \in \mathcal{R}_q$, $\{\mathbf{x}\}$ will denote the previous operation applied to every x_i. Due to the particular structure of the NTRU lattice and to the fact that the NTRU signature is a lattice vector, giving $\mathbf{s}$ as the signature suffices to reconstruct the right part using the public key. This trick permits to save half of the space needed to represent the NTRU signature.

Polynomials $\mathbf{F}$ and $\mathbf{G}$ play an important role in the size of the error when rounding-off the target. The technique of [HHGP+03] permits to find those polynomials in such a way that $\|\mathbf{F}\| \approx \|\mathbf{G}\| \approx \sqrt{\frac{N}{12}}\|\mathbf{f}\|$ so that the error when signing using sk is of size approximately $(\sqrt{\frac{N}{6}} + \nu\frac{N}{6\sqrt{2}})\|\mathbf{f}\|$. As a comparison, a invalid signer trying to sign using pk instead of sk would generate an error of magnitude $\nu\sqrt{\frac{N}{12}}q$.

Algorithme 1 : KeyGen(N, q, d, $\mathcal{N}$, ν)

Input : N, q, d, $\mathcal{N}$, and ν
Output : $pk = \mathbf{h} = \mathbf{f}^{-1} * \mathbf{F} \mod q$ and $sk = \mathbf{f}, \mathbf{g}$

begin
  repeat
    $\mathbf{f} \xleftarrow{\$} \mathcal{T}(d)$, $\mathbf{g} \xleftarrow{\$} \mathcal{T}(d)$;
    Generate $\mathbf{F}$ and $\mathbf{G}$ using [HHGP+03];
  until $\mathbf{f}$ *and* $\mathbf{g}$ *are invertible in* $\mathcal{R}_q$;
  $\mathbf{h} = \mathbf{g} * \mathbf{f}^{-1}$;
  return $pk = \begin{pmatrix} \mathbf{1} & \mathbf{h} \\ \mathbf{0} & \mathbf{q} \end{pmatrix}$, $sk = \begin{pmatrix} \mathbf{f} & \mathbf{F} \\ \mathbf{g} & \mathbf{G} \end{pmatrix}$;

Algorithme 2 : NTRUSign(pk, sk, μ)

Input : Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output : ($\mathbf{s}$,cpt) the NTRU signature

begin
  $cpt \leftarrow 0$;
  repeat
    $cpt \leftarrow cpt + 1$;
    $\mathbf{m} \leftarrow H(\mu, cpt) \in \mathcal{R}_q$;
    $(\mathbf{x}, \mathbf{y}) = (\mathbf{0}, \mathbf{m}) \cdot \begin{pmatrix} \mathbf{G} & -\mathbf{F} \\ -\mathbf{g} & \mathbf{f} \end{pmatrix} / q$;
    $\mathbf{s} = -\{\mathbf{x}\} * \mathbf{f} - \{\mathbf{y}\} * \mathbf{g}$;
  until $\|(\mathbf{s}, \mathbf{s} * \mathbf{h} - \mathbf{m})\|_\nu \leq \mathcal{N}$;
  return $(\mathbf{s}, cpt)$;

Algorithme 3 : Verify($pk = \mathbf{h}, \mathbf{s}, cpt, \mu$)

Input : Public key pk, the signature $\mathbf{s}$, and the message $\mu \in \{0,1\}^*$
Output : ***true*** if and only if $\mathbf{s}$ is a valid signature of μ

begin
  $\mathbf{m} = H(\mu, cpt)$;
  if $\|(\mathbf{s}, \mathbf{s} * \mathbf{h} - \mathbf{m})\|_\nu \leq \mathcal{N}$ **then**
    return ***true***;
  else
    return ***false***;

Parameters. Setting concrete NTRUSign parameters for a given security level seems to be an unclear task to perform. Nevertheless, the authors of [HHgP+05] provide a generic algorithm to generate such parameters, given the security parameter k, the signing tolerance ρ[3], and an upper bound N_{max} on the degree of the polynomials $\mathbf{f}$ and $\mathbf{g}$. Even if this algorithm doesn't take into account best known attacks, it can provide one with a hint of how the parameters should look like, relatively to one another. Therefore we will use it to get N, q, d, $\mathcal{N}$, and ν, and then check that best known attacks are out of range. New strong NTRUSign parameters have been generated and are presented in Table 3.2. We will not care about the transcript length as the fix we propose hides the leaky part of the signature, and an adversary would not learn anything more from issued signatures.

3.4 General overview of Lyubashevsky's scheme

In this section, we recall briefly the signature scheme presented by Lyubashevsky at EuroCrypt'12, and refer the reader to the original paper [Lyu12] for more details. The most efficient instantiations of this scheme rely on the average-case hardness of two problems:

3. if $\mathcal{E}$ is the expected size of a signature, the verifying process should fail for every signature whose size is greater than $\rho\mathcal{E}$. Notice that the author also use one in [Lyu12], namely η. We will be tempted to set $\rho = \eta$ in next sections.

k	N	d	q	ν	$\mathcal{N}$	ω_{cmb}	c	ω_{lk}	ω_{frg}	γ	ω_{lf}	R	L
100	431	34	2^{10}	0.16686	141	167	3.714	187	172	0.0516	415	131	180
112	479	42	2^{10}	0.15828	165	200	4.232	209	137	0.0558	470	157	200
128	541	61	2^{11}	0.14894	211	269	3.711	239	329	0.0460	541	207	226
160	617	57	2^{11}	0.13946	217	269	3.709	272	360	0.0431	627	210	258

TABLE 3.2 – New `NTRUSign` parameters, $\rho = 1$, no perturbation.

the $\text{SIS}_{q,n,m,d}$ decisional problem and the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem, which are at least as hard as the worst-case of the $\mathcal{O}(n^{1.5})$-SIVP [Ajt96].

Algorithme 4 : KeyGen(n, m, k, q)

Input : n, m, k, q
Output : $pk = (\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times k}$ and $sk = \mathbf{S} \in \mathbb{Z}_q^{m\times k}$

begin
  $\mathbf{S} \xleftarrow{\$} \{-d, \ldots, 0, \ldots, d\}^{m\times k}$;
  $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$;
  $\mathbf{T} \leftarrow \mathbf{AS}$;
  return $pk = (\mathbf{A}, \mathbf{T}), sk = \mathbf{S}$;

Algorithme 5 : Sign(pk, sk, μ)

Input : Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output : $(\mathbf{z}, \mathbf{c})$ the signature

begin
  $\mathbf{y} \xleftarrow{\$} D_\sigma^m$;
  $\mathbf{c} \leftarrow H(\mathbf{Ay}, \mu)$;
  $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$;
  return $(\mathbf{z}, \mathbf{c})$ *with probability* $\min\left(\frac{D_\sigma^m(\mathbf{z})}{M \cdot D_{\mathbf{Sc},\sigma}^m(\mathbf{z})}, 1\right)$;

Algorithme 6 : Verify$(pk, (\mathbf{z}, \mathbf{c}), \mu)$

Input : Public key, message μ, and the signature $(\mathbf{z}, \mathbf{c})$ to check
Output : ***true*** if and only if $(\mathbf{z}, \mathbf{c})$ is a valid signature of μ

begin
  if $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$ ***and*** $\|\mathbf{z}\| \leq \eta\sigma\sqrt{m}$ **then**
    return ***true***;
  else
    return ***false***;

As mentioned by the author, key sizes can be shrunk by a factor k using more structured matrices and relying on the ring version of the SIS problem, but we will skip this detail in this section for simplicity. Public and private keys are respectively uniformly random matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{S} \in \{-d, \ldots, 0, \ldots, d\}^{m\times k}$ and the signature process invokes a random oracle $H : \{0,1\}^* \to \left\{\mathbf{v} : \mathbf{v} \in \{0,1\}^k, \|\mathbf{v}\|_1 \leq \kappa\right\}$. A signature $(\mathbf{z}, \mathbf{c})$ of a message μ corresponds to a combination of the secret key and the hash of this message, shifted by a commitment value also used in the random oracle.

3.5 Description of our scheme

3.5.1 Putting the pieces together

Before exposing our scheme, we want to recall an important property over the NTRU lattice that we will make use of. We denote :

$$\Lambda_q^{\perp}(\mathbf{P}) = \left\{(\mathbf{y_1}, \mathbf{y_2}) / (\mathbf{1} \quad \mathbf{h}) \cdot (\mathbf{y_1} \quad \mathbf{y_2})^t = \mathbf{0} \mod q\right\} = \left\{(-\mathbf{h} * \mathbf{x} \mod q, \mathbf{x}), \mathbf{x} \in \mathbb{Z}_q^N\right\} \quad (3.4)$$

Then one can see $\Lambda_q^{\perp}(\mathbf{P}) = q \cdot \Lambda_q(\mathbf{P})^*$, and $\Lambda_q(\mathbf{P}) = q \cdot \Lambda_q^{\perp}(\mathbf{P})^*$. If we borrow the notation from code-based cryptography, if $\Lambda_q(\mathbf{P})$ is generate by $\mathbf{P} = (\mathbf{1}, \mathbf{h})$ then $\Lambda_q^{\perp}(\mathbf{P})$ is generated by $(\mathbf{h}, -\mathbf{1})$.

As the key generation part of our scheme is exclusively constituted by the NTRUSign key generation process, we use the algorithm described in [HHgP+05] to get N, q, d, $\mathcal{N}$, and ν, then invoke algorithm 1 for our keyGen procedure, to get the public and private matrices $\mathbf{P}$ and $\mathbf{S}$ as depicted in algorithm 7.

To sign a message $\mu \in \{0,1\}^*$, we will need a regular random oracle $H : \{0,1\}^* \to \mathcal{R}_q$. To add some randomness to our signature, the oracle's input will be an element of $\mathcal{R}_q$ represented under a bit string concatenated with our message μ. We then NTRUSign the oracle's output to get our leaky sample, which we shift by a mask $(\mathbf{y_1}, \mathbf{y_2})$ large enough to statistically hide this leak. Finally, we apply a rejection sampling step to ensure that the overall signature follows the expected distribution.

Algorithme 7 : KeyGen(N, q, d, $\mathcal{N}$, and ν)

Input : N, q, d, $\mathcal{N}$, and ν
Output : $pk = \mathbf{h} = \mathbf{f}^{-1} * \mathbf{F} \mod q$ and $sk = \mathbf{f}, \mathbf{F}$

begin
  repeat
    $\mathbf{f} \xleftarrow{\$} \mathcal{T}(d)$, $\mathbf{g} \xleftarrow{\$} \mathcal{T}(d)$;
    Generate $\mathbf{F}$ and $\mathbf{G}$ using [HHGP+03];
  until $\mathbf{f}$ *and* $\mathbf{g}$ *are invertible in* $\mathcal{R}_q$;
  $\mathbf{h} = \mathbf{f}^{-1} * \mathbf{F}$;
  return $\mathbf{P} = (-\mathbf{h}, \mathbf{1})$, $\mathbf{S} = (\mathbf{f}, \mathbf{F})$;

Algorithme 8 : Sign($\mathbf{P}, \mathbf{S}, \mu$)

Input : Public and private keys, and $\mu \in \{0,1\}^*$ the message to sign
Output : $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ the signature

begin
  $\mathbf{y_1} \xleftarrow{\$} D_\sigma^N$, $\mathbf{y_2} \xleftarrow{\$} D_\sigma^N$;
  $\mathbf{e} = H(\mathbf{P}(\mathbf{y_1}, \mathbf{y_2}), \mu) = H(\mathbf{y_2} - \mathbf{h} * \mathbf{y_1}, \mu)$;
  $(\mathbf{s}, \mathbf{t}) = \mathtt{NTRUSign}_{\mathbf{S}}(\mathbf{0}, \mathbf{e})$;
  $(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t}) + (\mathbf{y_1}, \mathbf{y_2})$;
return $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ *with probability* $\min\left(\frac{D_\sigma^{2N}(\mathbf{x})}{M \cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1\right)$;

We insist on the fact that in the original scheme *with perturbations*, the aim was to sign the message μ enough times so that the transcript an adversary could collect with couples of messages and signatures is short enough to make all secret key recovery techniques fail. The main difference in our scheme consists in hiding the leaky part with something larger so that it becomes indistinguishable, rather than signing it again and again. In other words, the leaky part of NTRUSign plays the role of the secret key in [Lyu12].

3.5.2 Sets of parameters

Hereafter is a set of parameters for our signature scheme, given different security levels. One interesting aspect of those sets is that we had to raise q and N for our NTRUSign part to

Algorithme 9 : Verify$(\mathbf{P}, (\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}, \mu)$

Input : Public key $\mathbf{P}$, a signature $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$, and a message μ
Output : ***true*** if and only if $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$ is a valid signature of μ

```
begin
    if ||(x1, x2)||_2 ≤ ησ√(2N) and H(P·(x1, x2) − e, μ) = e then
        return true;
    else
        return false;
```

be secure, but due to the small norm of our NTRU signature, this q is not raised as much as in [Lyu12]. This results in the nice property of lowering key and signature sizes.

Security parameter (bits) k	100	100	112	112	128	128	160	160
Optimized for	Size	Speed	Size	Speed	Size	Speed	Size	Speed
N	431	431	479	479	541	541	617	617
d	34	34	42	42	61	61	57	57
$\log_2(q)$	16	16	16	16	16	16	16	16
η (lemma 2.2)	1.296	1.296	1.297	1.297	1.299	1.299	1.314	1.314
ν	0.16686	0.16686	0.15828	0.15828	0.14894	0.14894	0.13946	0.13946
$\mathcal{N}$	109	139	128	165	160	213	165	218
α (lemma 2.1)	6	12	6.5	13	7	14	7.5	15
$\sigma = \eta\alpha\mathcal{N}$	848	2162	1080	2783	1455	3874	1627	4297
$M = e^{1+1/(2\alpha^2)}$ (lemma 2.3)	7.492	2.728	7.477	2.726	7.465	2.725	7.455	2.724
signature size (bits) $\approx 2N\log_2(\alpha\sigma)$	10700	12700	12300	14600	14500	17100	16800	19800
pk size (bits) $\approx N\log_2(q)$	6900	6900	7700	7700	8700	8700	9900	9900
sk size (bits) $\approx 2N\log_2(3)$	1400	1400	1550	1550	1750	1750	2000	2000

TABLE 3.3 – Parameters, signature and key sizes for our scheme, given the security level k

3.5.3 Security of our scheme

In this section, k will represent the security parameter, typically $k = 80$ for a "toy" security, $k = 100$ or 112 for a current security, and $k = 128$ to 160 for a "strong" security. Due to space restrictions, we will only mention the different known kinds of attack the reader can find in the literature. For further details, we refer to [HHGPW09, MS01, DN12, HG07] for the NTRUSign part, and to [Lyu12, LP11, MR09] for Lyubashevsky's scheme. Due to the hybridness of our scheme, potential attacks could be of three types, that we exposed in what follows, before tackling them.

The first one consists in attacking the NTRU lattice by trying to find back the private key $(\mathbf{f}, \mathbf{F})$ only from the public key $\mathbf{h} = \mathbf{f}^{-1} * \mathbf{F}$ (and eventually some signatures after Lyubashevsky's layer). Even if there is no theoretical proof on the intractability of this attack, there hasn't been (to the best of our knowledge) any efficient way to do so neither. Parameters given in table 3.3 have been chosen so that someone succeeding in doing so would achieve a lattice reduction with better Hermit factors than those described in 3.4 respectively to the security parameter k. Such a good algorithm could obviously be used to solve worst-case of lattice problems on general convolution modular lattices. A second way to break our signature scheme, still by finding out the secret key, could be trying to isolate the NTRU signature inside our signature to find enough leaky parts to then proceed to a [DN12]-like attack. This issue

k	100	112	128	160
δ	1.00696	1.00652	1.00602	1.00521
α	12	13	14	15

TABLE 3.4 – δ and α as a function of the security level k

is addressed by Theorem 3.2.1. Finally, we show that if an adversary succeed in creating a forgery in polynomial-time, then we can use this forgery to solve the SIS problem, which is the main theorem (3.5.1) of this section.

Regarding attacks against the `NTRUSign`, all parameters have been heighten so they ensure way more than k bits of security. We are aware that some attacks might lower the security level [MS01, GN08, HG07, DN12], but also due to our lack of knowledge on how to benefit from the singular structure of `NTRU` lattices, we take a conservative gap between claimed and effective security. Nevertheless, all parameters given in table 3.2 were set in such a way that lattice reduction techniques are meant to fail, either by finding a short vector too long, either by a computational complexity blow up. Also due to recent attacks such as [HG07, GN08, DN12], the `NTRUSign` parameters presented in [HHGPW09] don't reach the claimed security. Therefore, we ran the Baseline Parameter Generation Algorithm of [HHgP+05], and integrated the most recent known attacks. As one can notice, we intentionally took a "huge" degree N, and a big q for two reasons. It first gives more confidence about the security of the underlying `NTRU` lattice, and it was also necessary for proofs to work after applying Lyubashevsky's layer to our scheme.

As far as we know, lattice-reduction over $\Lambda_q^\perp(\mathbf{A})$ is the most efficient technique to solve random instances of knapsack problems. Experiments in [GN08] led to the ability of finding a vector $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A})$ whose norm is at most $\|\mathbf{v}\|_2 \leq \delta^{2N} \cdot \sqrt{q}$, for δ depending on the lattice-reduction algorithm which is used (see below). Experiments from Micciancio and Regev [MR09] conducted to a minimum of $\delta^m \cdot q^{n/m} \approx \min(q, 2^{2\sqrt{N \log_2(q) \log_2(\delta)}})$ for $m \approx \sqrt{N \log_2(q)/\log_2(\delta)}$.

In 2011, Lindner and Peikert [LP11] achieved to give an approximation of the best δ reachable for a given security level k, using a conservative approximation on BKZ's running time :

$$t_{BKZ}(\delta) = \log_2(T_{BKZ}(\delta)) = 1.8/\log_2(\delta) - 110 \tag{3.5}$$

where $T_{BKZ}(\delta)$ is the running time of BKZ in second, on their machine. So assuming one can achieve 2^{30} operations per second on a "standard" computer, to determine δ given the security parameter k, we have :

$$\log_2(\delta) := \frac{1.8}{\log_2(\frac{T_{BKZ}(\delta)}{2^{30}}) + 110} = \frac{1.8}{k - 30 + 110} = \frac{1.8}{k + 80} \tag{3.6}$$

This equation gives us a way to get δ as a function of the security parameter k, see table 3.4. Similarly to [Lyu12], in order to hide properly our leaky part $(\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t})$, we will use Lemmas 2.1 and 2.2 to get a proper α.

Against forgeries. In this section, we give a short overview of the material that will be needed to base our signature scheme upon the SIS problem over random `NTRU` lattices. This leads to a signature scheme based on the worst-case hardness of the $\tilde{\mathcal{O}}(N^{1.5})$-SIVP problem over general convolutional modular lattices.

We now expose the core of the reduction, which allows us to base the security of our signature scheme upon the ℓ_2-SIS$_{q,N,2N,\beta}$ Problem of general `NTRU` lattices. Our main theorem

Hybrid 1

Sign$(\mathbf{P}, \mathbf{S}, \mu)$

1. $\mathbf{y_1} \xleftarrow{\$} D_\sigma^N$, $\mathbf{y_2} \xleftarrow{\$} D_\sigma^N$
2. $\mathbf{e} \xleftarrow{\$} \mathcal{R}_q$
3. $(\mathbf{s}, \mathbf{t}) = \texttt{NTRUSign}_{\mathbf{S}}(\mathbf{0}, \mathbf{e})$
4. $(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{0}, \mathbf{e}) - (\mathbf{s}, \mathbf{t}) + (\mathbf{y_1}, \mathbf{y_2})$
5. with probability $\min(\frac{D_\sigma^{2N}(\mathbf{x})}{M \cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1)$:
   — Output $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$
   — Program $H(\mathbf{P} \cdot (\mathbf{x_1}, \mathbf{x_2}) - \mathbf{e}, \mu) = \mathbf{e}$

Hybrid 2

Sign$(\mathbf{P}, \mathbf{S}, \mu)$

1. $\mathbf{e} \xleftarrow{\$} \mathcal{R}_q$
2. $(\mathbf{x_1}, \mathbf{x_2}) \xleftarrow{\$} D_\sigma^{2N}$
3. with probability $1/M$:
   — Output $(\mathbf{x_1}, \mathbf{x_2}), \mathbf{e}$
   — Program $H(\mathbf{P} \cdot (\mathbf{x_1}, \mathbf{x_2}) - \mathbf{e}, \mu) = \mathbf{e}$

FIGURE 3.1 – Signing Hybrids

will be proved by two lemmas, mostly proved in [Lyu12], but revisited in appendix 3.A in some of the details in order to fit best with our sets of parameters.

Theorem 3.5.1 ([Lyu12] revisited)**.** *Assume there is polynomial-time forger $\mathcal{F}$, which makes at most s (resp. h) queries to the signing (resp. random) oracle, who breaks our signature scheme (with parameters such those in Table 3.3), then there is a polynomial-time algorithm to solve the ℓ_2-$SIS_{q,N,2N,\beta}$ Problem for $\beta = 2\eta\sigma\sqrt{2N}$ with probability $\approx \frac{\delta^2}{h+s}$. Moreover, the signing algorithm 8 produces a signature with probability $\approx \frac{1}{M}$ and the verifying algorithm 9 accepts the signature produced by an honest signer with probability at least $1 - 2^{-2N}$.*

Proof. We begin the proof by showing that our signature algorithm 8 is statistically close (within distance $\epsilon = s(h+s) \cdot 2^{-N+1} + s \cdot \frac{2^{-\omega(\log_2 2N)}}{M}$ by Lemma 5.1) to the one in Hybrid 2 in Figure 3.1. Given that Hybrid 2 outputs something with probability $1/M$, our signing algorithm will output something too with probability $(1-\epsilon)/M$. Then by Lemma 5.2, we show that if a forger $\mathcal{F}$ succeeds in forging with probability δ when the signing algorithm is replaced by the one in Hybrid 2, then we can use $\mathcal{F}$ to come up with a non-zero lattice vector $\mathbf{v}$ such that $\|\mathbf{v}\| \leq 2\eta\sigma\sqrt{2N}$ and $\mathbf{Pv} = 0$ with probability at least $\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right)$. □

Lemma 5.1 ([Lyu12] revisited)**.** *Let $\mathcal{D}$ be a distinguisher who can query the random oracle H and either the actual signing algorithm 8 or Hybrid 2 in Figure 3.1. If he makes h queries to H and s queries to the signing algorithm that he has access to, then for all but a $e^{-\Omega(N)}$ fraction of all possible matrices $\mathbf{P}$, his advantage of distinguishing the actual signing algorithm from the one in Hybrid 2 is at most $s(h+s) \cdot 2^{-N+1} + s \cdot \frac{2^{-\omega(\log_2 2N)}}{M}$.*

Lemma 5.2 ([Lyu12] revisited)**.** *Suppose there exists a polynomial-time forger $\mathcal{F}$ who makes at most h queries to the signer in Hybrid 2, s queries to the random oracle H, and succeeds in forging with probability δ. Then there exists an algorithm of the same time-complexity as $\mathcal{F}$ that for a given $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{N \times 2N}$ finds a non-zero $\mathbf{v}$ such that $\|\mathbf{v}\|_2 \leq 2\eta\sigma\sqrt{2N}$ and $\mathbf{Pv} = \mathbf{0}$ with probability at least*

$$\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right).$$

Conclusion

In this work, we described a method for sealing `NTRUSign` signatures' leak, based on the worst-case hardness of standard problems over ideal lattices. This method differs from existing heuristic countermeasures such the use of perturbations [HHGP+03] or the deformation of the parallelepiped [HWH08] - both broken [DN12] - but also from provably secure modifications of `NTRUSign` like [SS13] which uses gaussian sampling techniques in order to not disclose the secret basis [GPV08]. Moreover, this technique seems to be sufficiently generic to be applied on GGH signatures. Details on this will be provided in a longer version of this paper.

We show that it is actually possible to use the rejection sampling technique from [Lyu12] instead of gaussian sampling to achieve zero-knowledgeness, while keeping most of `NTRUSign`'s efficiency. Moreover, parameter refinements allowed us to lower the rejection rate, leading to performance improvements regarding [Lyu12], together with smaller signature and secret key sizes.

It might be possible to improve the rejection sampling procedure even more using techniques such those in [DDLL13], but it seems necessary to break the public key's particular shape to do so. Therefore, it is still an open question whether the resulting benefit in the signature size would worth the key sizes growth.

Acknowledment. The authors would like to thank Léo Ducas for helpful discussions on rejection sampling, as well as Paulo S. L. M. Barreto for pointing out a typo on **h** in the original paper [HHGP+03, Sec.2, KeyGen, point 2.(c)] also present in the conference version of this chapter[4], and the anonymous PQCrypto reviewers for their valuable comments.

4. The ePrint version should be updated soon.

Appendix

3.A Proofs

3.A.1 Section 3.2

Most of the lemmas of Section 3.2 are proved in [Lyu12]. We therefore refer the reader to the original paper for these proofs. Nevertheless, we adapted lemma 2.3 to make bounds tighter with respect to different security levels. We prove the correctness of our modification :

Proof.

$$D_{\sigma}^{2N}(\mathbf{x})/D_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \rho_{\sigma}^{2N}(\mathbf{x})/\rho_{\mathbf{v},\sigma}^{2N}(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{v}\|^2 - \|\mathbf{x}\|^2}{2\sigma^2}\right) = \exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle\mathbf{x},\mathbf{v}\rangle}{2\sigma^2}\right)$$

By lemma 2.1 and using the fact that $\sigma = \omega(\|\mathbf{v}\|\sqrt{\log(2N)})$, with probability $1 - 2^{-\omega(\log(2N))}$ we have

$$\exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle\mathbf{x},\mathbf{v}\rangle}{2\sigma^2}\right) < \exp\left(\frac{\|\mathbf{v}\|^2 + \omega(\sigma\|\mathbf{v}\|\sqrt{\log(2N)})}{2\sigma^2}\right) = \mathcal{O}(1).$$

And more precisely, by setting $r = \alpha\|\mathbf{v}\|\sigma$ in lemma 2.1 with α determined by the security parameter k in table 3.1, we obtain with probability $1 - 2^{-k}$ that

$$\exp\left(\frac{\|\mathbf{v}\|^2 - 2\langle\mathbf{x},\mathbf{v}\rangle}{2\sigma^2}\right) < \exp\left(\frac{\|\mathbf{v}\|^2 + 2\alpha\|\mathbf{v}\|\sigma}{2\sigma^2}\right) = \exp\left(\frac{\|\mathbf{v}\|^2}{2\sigma^2} + \frac{\alpha\|\mathbf{v}\|}{\sigma}\right) \overset{\sigma=\alpha\|\mathbf{v}\|}{=} e^{1+1/(2\alpha^2)}.$$

□

3.A.2 Proofs of Section 3.5

We begin with the proof of lemma 5.1, which states that our actual signing algorithm 8 is indistinguishable from Hybrid 2 depicted in Figure 3.1, using Hybrid 1 as an intermediate step.

Proof. First, let us prove that $\mathcal{D}$ has an advantage of at most $s(h+s)\cdot 2^{-N+1}$ of distinguishing between the actual signature scheme 8 and Hybrid 1. The only difference between those algorithms is the output of the random oracle H. It is chosen uniformly at random from $\mathcal{R}_q$ in Hybrid 1, rather than according to $H(\mathbf{Py},\mu)$ for $y \xleftarrow{\$} D_q^{2N}$ in the real signing algorithm. Random oracle in Hybrid 1 is then programmed to answer $H(\mathbf{Px}-\mathbf{e},\mu) = H(\mathbf{Py},\mu)$ without checking whether $(\mathbf{Py},\mu)$ was already queried or not. Since $\mathcal{D}$ calls H (resp. algorithm 8) h (resp s) times, at most $s+h$ values of $(\mathbf{Py},\mu)$ will be set. We now bound the probability of generating such an already set value. Using lemma 2.2, we can see that for any $\mathbf{t}\in\mathbb{Z}_q^N$,

$$Pr[\mathbf{Py}=\mathbf{t};\mathbf{y}\xleftarrow{\$}D_q^{2N}] = Pr[\mathbf{y_1} = (\mathbf{t}-\mathbf{h}*\mathbf{y_0});\mathbf{y}\xleftarrow{\$}D_q^{2N}] \leq \max_{\mathbf{t'}\in\mathbb{Z}_q^N} Pr[\mathbf{y_1}=\mathbf{t'};\mathbf{y_1}\xleftarrow{\$}D_q^N] \leq 2^{-N}.$$

Therefore, if Hybrid 1 is called s times with the probability of getting a collision begging less than $(s+h)\cdot 2^{-N+1}$ for each call, then the probability of coming up with a collision after s calls is at most $s(s+h)\cdot 2^{-N+1}$.

We pursue by showing that the outputs of Hybrids 1 and 2 are statistically within distance $\frac{2^{-\omega(\log_2 2N)}}{M}$. As noticed in [Lyu12], this is an almost straightforward consequence of theorem 3.2.1 : assuming both Hybrids output $(\mathbf{x}, (-\mathbf{s}, \mathbf{e}-\mathbf{t}))$ with respective probabilities $\min\left(\frac{D_\sigma^{2N}(\mathbf{x})}{M\cdot D^{2N}_{(-\mathbf{s},\mathbf{e}-\mathbf{t}),\sigma}(\mathbf{x})}, 1\right)$ for Hybrid 1 and $1/M$ for Hybrid 2, they respectively play the role of $\mathcal{A}$ and $\mathcal{F}$ (with $T = \eta\alpha\mathcal{N}$). Even if both Hybrids only output $\mathbf{e}$, this does not increase the statistical distance because given $\mathbf{e}$, one can generate $(-\mathbf{s}, \mathbf{e}-\mathbf{t})$ such that $\mathbf{P}(-\mathbf{s}, \mathbf{e}-\mathbf{t}) = \mathbf{e}$ simply by `NTRUSigning` $(\mathbf{0}, \mathbf{e})$, and this will have the exact same distribution as the $\mathbf{e}$ in both Hybrids. Finally, as the signing oracle is called s times, the statistical distance between the two Hybrids is at most $s\cdot\frac{2^{-\omega(\log_2 2N)}}{M}$, or more concretely $s\cdot\frac{2^{-k}}{M}$. The claim in the lemma is obtained by summing both distances. □

We now prove lemma 5.2, which provides us with a ℓ_2-SIS$_{q,N,2N,\beta}$ solver using a polynomial-time successful forger.

Proof. Let $t = h + s$ be the number of calls to the random oracle H during $\mathcal{F}$'s attack. H can be either queried by the forger or programmed by the signing algorithm when $\mathcal{F}$ asks for some message to be signed. We pick random coins ϕ (resp. ψ) for the forger (resp. the signer), along with $\mathbf{r}_1, \ldots, \mathbf{r}_t \leftarrow \mathcal{R}_q$, which will correspond to the H's responses. We now consider a subroutine $\mathcal{A}$, which on input $(\mathbf{P}, \phi, \psi, \mathbf{r}_1, \ldots, \mathbf{r}_t)$ initializes $\mathcal{F}$ by giving it $\mathbf{P}$ and ϕ and run it. Each time $\mathcal{F}$ asks for a signature, $\mathcal{A}$ runs Hybrid 2 using the signer's coins ψ to get a signature, and H is programmed to answer with the first unused $\mathbf{r}_i \in (\mathbf{r}_1, \ldots, \mathbf{r}_t)$. $\mathcal{A}$ keeps track of the answered $\mathbf{r}_i$ in case $\mathcal{F}$ queries the same message to be signed again. Similarly, if $\mathcal{F}$ queries directly the random oracle, H will answer with the first unused $\mathbf{r}_i \in (\mathbf{r}_1, \ldots, \mathbf{r}_t)$, unless the query was already made. When $\mathcal{F}$ ends and eventually come up with an output (with probability δ), $\mathcal{A}$ simply forwards $\mathcal{F}$'s output.

With probability δ, $\mathcal{F}$ succeeds in forging, coming up with $(\mathbf{x}, \mathbf{e})$ satisfying $\|\mathbf{x}\| \leq \eta\sigma\sqrt{2N}$ and $H(\mathbf{P}\mathbf{x} - \mathbf{e}, \mu) = \mathbf{e}$ for some message μ. If H was not queried nor programmed on some input $\mathbf{w} = \mathbf{P}\mathbf{x} - \mathbf{e}$, then $\mathcal{F}$ has only a $1/|\mathcal{R}_q| = q^{-N}$ (*i.e. negligible*) chance of generating a $\mathbf{e}$ such that $\mathbf{e} = H(\mathbf{w}, \mu)$. Therefore, $\mathcal{F}$ has at least a $\delta - q^{-N}$ chance of succeeding in a forgery with $\mathbf{e}$ being one of the $\mathbf{r}_i$'s. Assume $\mathbf{e} = \mathbf{r}_j$, we are left with two cases : $\mathbf{r}_j$ is a response to a random oracle query made by $\mathcal{F}$, or it was program during the signing procedure invoked by $\mathcal{A}$.

Let first assume that the random oracle was programmed to answer $H(\mathbf{P}\mathbf{x}' - \mathbf{e}, \mu') = \mathbf{e}$ on input μ'. If $\mathcal{F}$ succeeds in forging $(\mathbf{x}, \mathbf{e})$ for some (possibly different) message μ, then $H(\mathbf{P}\mathbf{x}' - \mathbf{e}, \mu') = H(\mathbf{P}\mathbf{x} - \mathbf{e}, \mu)$. If $\mu \neq \mu'$ or $\mathbf{P}\mathbf{x}' - \mathbf{e} \neq \mathbf{P}\mathbf{x} - \mathbf{e}$, then $\mathcal{F}$ found a pre-image of $\mathbf{r}_j$. Therefore, $\mu = \mu'$ and $\mathbf{P}\mathbf{x}' - \mathbf{e} = \mathbf{P}\mathbf{x} - \mathbf{e}$, so that $\mathbf{P}(\mathbf{x} - \mathbf{x}') = \mathbf{0}$. We know that $\mathbf{x} - \mathbf{x}' \neq \mathbf{0}$ (because otherwise $(\mathbf{x}, \mathbf{e})$ and $(\mathbf{x}', \mathbf{e})$ sign the same message μ), and since $\|\mathbf{x}\|_2, \|\mathbf{x}'\|_2 \leq \eta\sigma\sqrt{2N}$, we have that $\|\mathbf{x} - \mathbf{x}'\| \leq 2\eta\sigma\sqrt{2N}$.

Let now assume that $\mathbf{r}_j$ was a response of the random oracle invoked by $\mathcal{F}$. We start by recording $\mathcal{F}$'s output $(\mathbf{x}, \mathbf{r}_j)$ for the message μ, then generate fresh random elements $\mathbf{r}'_j, \ldots, \mathbf{r}'_t \leftarrow \mathcal{R}_q$. We then run $\mathcal{A}$ again with input $(\mathbf{P}, \phi, \psi, \mathbf{r}_1, \ldots, \mathbf{r}_{j-1}, \mathbf{r}'_j, \ldots, \mathbf{r}'_t)$, and by the General Forking Lemma [BN06], we obtain that the probability that $\mathbf{r}'_j \neq \mathbf{r}_j$ and the forger uses the random oracle response $\mathbf{r}'_j$ (and the query associated to it) in its forgery is at least

$$\left(\delta - \frac{1}{|\mathcal{R}_q|}\right)\left(\frac{\delta - 1/|\mathcal{R}_q|}{t} - \frac{1}{|\mathcal{R}_q|}\right),$$

and thus with the above probability, $\mathcal{F}$ outputs a signature $(\mathbf{x}', \mathbf{r}'_j)$ of the message μ and $\mathbf{Px}-\mathbf{e} = \mathbf{Px}'-\mathbf{e}'$ where we let $\mathbf{e} = \mathbf{r}_j$ and $\mathbf{e}' = \mathbf{r}'_j$. By rearranging terms in the above equality we obtain

$$\begin{aligned} \mathbf{P}(\mathbf{x}-\mathbf{x}') - \overbrace{(\mathbf{e}-\mathbf{e}')}^{\mathbf{P}((\mathbf{0},\mathbf{e})-(\mathbf{0},\mathbf{e}')-((\mathbf{s},\mathbf{t})-(\mathbf{s}',\mathbf{t}')))} &= \mathbf{0} \\ \mathbf{P}\left(\mathbf{y}-\mathbf{y}'\right) &= \mathbf{0} \end{aligned} \tag{3.7}$$

But since $H(\mathbf{Py}, \mu) = \mathbf{e} = \mathbf{r}_j \neq \mathbf{r}'_j = \mathbf{e}' = H(\mathbf{Py}', \mu)$, necessarily $\mathbf{y} \neq \mathbf{y}'$, and as $\|\mathbf{y}\|_2, \|\mathbf{y}'\|_2 \leq \eta\sigma\sqrt{2N}$, we finally have that $\|\mathbf{y}-\mathbf{y}'\|_2 \leq 2\eta\sigma\sqrt{2N}$ with probability

$$\left(\delta - 2^{-k}\right)\left(\frac{\delta - 2^{-k}}{h+s} - 2^{-k}\right).$$

□

Chapitre 4

Un Schéma de Signature Traçable Non-Frameable

Sommaire

Contexte

Les signatures traçables furent introduites en 2004 comme une généralisation des signatures de groupe permettant d'améliorer la vie privée de ses utilisateurs. Dans ce chapitre, nous présentons le premier schéma de ce type basé sur les réseaux disposant des fonctionnalités suivantes :

— Non-framabilité : en cas de coalition (pouvant même impliquer le group manager), aucun utilisateur honnête de peut être accusé d'être l'auteur d'une signature qu'il n'a pas produite,

— Enrôlement dynamique : l'appartenance d'un utilisateur au groupe peut évoluer au cours du temps, sans avoir à réinitialiser le système en entier, et

— Traçabilité : le group manager peut tracer les signatures produites par un utilisateur spécifique, sans avoir à ouvrir les signatures d'utilisateurs honnêtes. De plus, cette capacité peut être déléguée efficacement à des agents externes s'exécutant en parallèle.

Ce schéma représente à travers les propriétés sus-mentionnées une réelle avancée par rapport aux précédents schémas basés sur les réseaux, qui sont tous statiques, framable et/ou invasifs (même un utilisateur honnête peut voir ses signatures ouvertes).

Nous proposons une nouvelle approche pour obtenir une signature traçable : les preuves testables à faible divulgation de connaissance (Testable weak Zero Knowledge (TwZK) en

anglais). Celles-ci permettent à un vérifieur de tester si un secret en particulier a été utilisé par le prouveur, sans apprendre autre chose de la preuve. Nous donnons un nouveau modèle de sécurité associé à ce type de preuves, qui s'avère particulièrement approprié pour les signatures traçables.

Nous formulons également une généralisation de deux problèmes sur les réseaux – le problème k-SIS [BF11] et le problème Split-SIS [NZZ15] – et montrons que notre schéma de signature traçable reposant sur ce nouveau problème généralisé (sur des matrices de type NTRU) est sûr dans le ROM, en assumant la difficulté dans le pire-cas de certains problèmes sur les réseaux.

Ce travail est issu d'une collaboration avec Carlos Aguilar Melchor, Olivier Blazy, et Philippe Gaborit.

4.1 Introduction

Since their introduction at EuroCrypt'91 by Chaum and van Heyst [Cv91], Group Signatures have met growing interest for the multitude of applications they provide. Roughly speaking, such schemes allow members of some entity/community/corporation to issue signatures on behalf of this latter both anonymously (*anonymity*) and accountably (*traceability*). Those desirable features easily find real-life applications in trusted computing, restricted areas access, or even auction protocols. In such schemes, the aforesaid community is administrated by a group manager ($\mathcal{GM}$), who owns two additional privileges compared to standard users: (1) Adding/Removing users to/from the group, and (2) Link signatures to their original issuer. Formal security requirements for such schemes were first introduced in [BMW03] (BMW model), then extended in [BSZ05, KY06] to the case of dynamic groups, where the number of users is not restricted to some pre-established bound during the initialization process. According to the BMW framework, a group signature scheme has to satisfy both *full-anonymity* and *full-traceability*. It is also worth enabling membership revocation in case of misbehaving users. As discussed in [LLNW14] the Verifier-Local Revocation (VLR) approach – which requires the verifier to have an up-to-date list of revoked users – seems to be the most convenient strategy to do so, both in term of efficiency and practicality regarding the already deployed PKI infrastructures. Indeed, in order to produce and verify a signature correctly, both the user and the verifier require an up-to-date public key. The VLR approach fixes the public key (pk does not need to be updated) and introduces a revocation list handled by the group manager. Before checking a signature, a verifier needs to ensure he has the latest revocation list. The VLR approach allows *offline* signing: a user willing to produce a signature on some digital document does not require any connection to check whether he has the correct version of pk. Therefore, our construction will also make use of the VLR to deal with dishonest users. The first lattice-based group signature scheme was proposed by Gordon *et al.* [GKV10] at AsiaCrypt'10 and since then, extensive efforts have been made to get efficient schemes whose keys and signature sizes increase only logarithmically in the number of members in the group (*e.g.* [LLLS13, LLNW14, LNW15, NZZ15, LLNW16]).

Besides, most group signature schemes link signatures back to their original issuer by opening *all* signatures. This is both unnecessarily inefficient as *only* the group manager can perform the openings, and unfair as honest users also have their signatures opened. At EuroCrypt'04, Kiayias *et al.* proposed an improved generalization of group signatures called traceable signatures, where "traceability" refers to the ability to retrieve all the signatures issued by a specific user [KTY04]. The crux here being that the Open algorithm reveals the identity of the issuer (potentially honest) whereas the Trace algorithm just confirms (or

invalidates) that a given signature was produced by the specific user being traced. The tracing key can be outsourced to tracing agents (sub-openers) which can independently (*i.e.* in parallel) monitor the activity of the associated user. Additionally, such signatures provide an efficient way for honest users to *claim* the paternity of a signature, without having to store all the one time random values (nonce) used during signing.

From a different point of view, although traceable/group signature schemes based on number theory suffer from worse asymptotic efficiency [1] and attacks in the eventuality of a quantum computer, they also enjoy additional properties such as *non-frameability*. This nice property ensures that in case of a collusion involving malicious users (and even the group manager), the resulting signature *can not* trace to an honest user outside the coalition. Once again, we are not aware of any lattice-based group signature scheme with such a feature. This can be partly explained as follows: on the one hand, the most binding part in group signature schemes is the group manager's privileges. As he is responsible for the key establishment and distribution, every single user's secret signing key is escrowed. Consequently, if the group manager is corrupted or attacked, anyone even outside the group that manages to get a secret signing key can issue signatures on behalf of the community, but also of a user, thereby accounting him for signing. On the other hand, every group signature scheme (in BMW and BSZ models at least) requires the group manager to be able to *link* signatures to their issuer in case of misbehaviours and therefore, the group manager still needs additional rights/information.

Our contributions. We fill the aforesaid void by introducing the first traceable signature scheme that benefits from the desirable features described above (traceable, dynamic enrollment, non-frameable), that is secure in the Random Oracle Model (ROM), assuming the worst-case hardness of lattice problems. Up to now, only pairing-based constructions achieved such features [Gro07, LY09, BP12], our construction is (to our knowledge) the first lattice-based (and more generally post-quantum) solution gathering these nice properties. Traceability is achieved through a technique due to Alamélou *et al.* [ABCG15] called *Testable weak Zero-Knowledge* (TwZK for short), that allows to test whether a specific secret was used during a proof of knowledge, without revealing additional information on this secret [ABCG15]. This technique will be recalled and modified to fit our scheme in Sec. 4.3. We also extend and merge two recent lattice problems: the k-SIS from [BF11] and the Split-SIS from [NZZ15]. The former corresponds to a classical SIS instance, but with k hints given on a valid solution, while the latter is an SIS instance on a matrix decomposed into several parts. Our resulting new problem – straightforwardly named (k, η)-Split-SIS – asks for an SIS solution of a matrix split into η parts given k hints. We prove this new problem to be polynomially equivalent to the SIS problem, for some parameters and up to some scaling. We believe this new problem can serve as a building block for future constructions. Finally, we generalize the nice SternExt protocol from [LNSW13] to the case where two matrices are used instead of just one, and one part is testable. The resulting modified protocol mSternExt is presented in Fig. 4.2. Our traceable signature scheme uses the NTRUSign fix from [ABDG14] (using techniques from [Lyu12]) so that the security relies on an SIS problem on NTRU-like matrix (similarly to [DLP14]).

1. Keys and signature sizes currently outperform all existing algebra-like group signature schemes, but the sub-exponential best-known attacks against those schemes make their efficiency decrease dramatically when the security level increases, mostly due to the large arithmetic involved.

Overview. Our approach through getting a group signature scheme is in the same vein as previous constructions: each user possesses a *short* secret that, once multiplied by a random public matrix, gives a public syndrome. A signature on some digital document is simply (a transcript of) a proof of knowledge on this secret. Technical details to achieve advanced features are discussed in the next paragraph. The group public key is a tuple of matrices $(\mathbf{P}, \mathbf{B})$ together with a public syndrome $\mathbf{s}$, where $\mathbf{B}$ is a random matrix and $\mathbf{P}$ is the public key of the modified NTRUSign scheme from [ABDG14] (which we denote mNTRUSign and is depected in details in chapter 3). Users' secret signing keys are relatively short pre-images $(\mathbf{x}_i, \mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i))$ of the same public syndrome $\mathbf{s}$.[2] A high level overview of the scheme is provided in Figure 4.1.

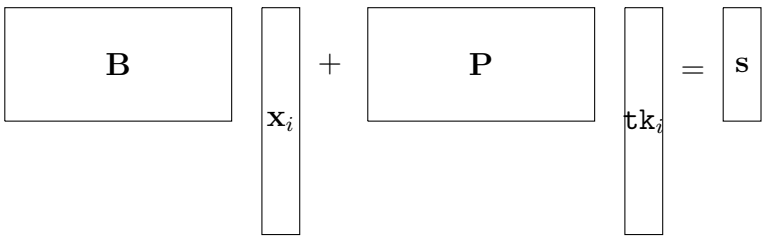


FIGURE 4.1 – Anonymity equation for our traceable signature scheme. Every registered group user own a private signing key $(\mathbf{x}_i, \mathtt{tk}_i)$ which is a pre-image (by public matrices $\mathbf{B}$ and $\mathbf{P}$) of the public syndrome $\mathbf{s}$.

The non-frameability feature is achieved through techniques similar to those in traitor tracing applications (see [LPSS14] for instance): every user $\mathcal{U}_i$ willing to join the group has to collaborate with the group manager to produce a valid secret signing key, but only part of this key is escrowed by the group manager (with respect to the public NTRU matrix $\mathbf{P} \in \mathbb{Z}_q^{n \times m}$). Therefore, the latter cannot misuse $\mathcal{U}_i$'s secret signing key, and hence frame him.

The traceability (in the sense of [KTY04]) is handled through Testable weak Zero-Knowledge proofs of knowledge, where the group manager owning the tracing keys can test whether a specific signing key was used or not. Users' tracing keys are non-leaking NTRUSign signatures (from [ABDG14]).

Finally, anonymity is guaranteed through the equation depicted in Fig. 4.1: each secret key $(\mathbf{x}_i, \mathtt{tk}_i)$ satisfies this equation.

Verification simply consists in checking that the transcript of the proof of knowledge is correct.

Techniques. Our construction mainly relies on three primitives. The first one is the NTRUSign fix by Aguilar *et al.* [MP12], which in the context of our traceable scheme allows to provide users' tracing keys. These keys serve two purposes: they provide group members with anonymity, and they constitute the witness being tested in the second and third primitive. The second primitive is a modification of Stern's protocol [Ste94] on lattices, as the original SternExt of [LNW15] only supports one matrix with a special distribution in infinite norm. Our modified protocol mSternExt supports two matrices, and is compatible with the last primitive: the notion of *Testable weak Zero-Knowledge* from [ABCG15]. Indeed, mSternExt allows for testing the tracing key being used during this protocol. Hence anybody

2. This *non-zero* syndrome $\mathbf{s}$ is mainly here to prevent the zero vector and the negation of a valid secret key from being valid secret keys.

in possession of a secret tracing key $\mathtt{tk}_i$ (associated to $\mathcal{U}_i$) can check whether user i signed a given message or not.

Limitations. Unfortunately, the new features that brings our traceable signature scheme comes at the cost of allowing only a sublinear number of users, namely $\mathcal{O}(N^{\frac{1}{4}})$ where N can be thought as the security parameter. This limited number of users comes from the security reduction from $k-\mathsf{SIS}$ to SIS from [BF11]. Attempts to define a new "One-More" type problem for SIS better suited for our scheme and reduce it to the standard SIS problem were unfruitful, but recent results from [LPSS14] spread the hope to improve this bound up to $\mathcal{O}(\frac{N}{\log N})$.

Road Map. The following Section is dedicated to the generalized SIS problem, and recalls about TwZK from [ABCG15]) used throughout this chapter. Detailed building blocks for our new construction are presented in Section 4.3. We then give formal security models for traceable signatures and propose a new lattice-based scheme both dynamic, traceable and non-frameable, together with its security analysis in Section 4.4. We finally end the paper with a conclusion in Section 4.5 together with open questions.

4.2 Preliminaries

Most of the necessary background to understand this chapter was introduce in chapter 2. Additional details regarding advanced cryptographic primitives are developped in the next section. We extend the SIS problem in this section.

k-SIS is known to be as hard as the standard SIS problem, up to a *constant* k [BF11], inherently making lattice-based group signature schemes *static*. We exploit a new result by [LPSS14] making k *polynomial*, thanks to the Rényi Divergence (RD), together with another recent problem.

Nguyen *et al.* recently introduce the Split-SIS problem, a version of the SIS problem where the matrix is split into two parts [NZZ15]. As our construction uses not two but three matrices, we generalize their definition of the Split-SIS problem to η matrices and extend the reduction to SIS. We believe this contribution is of general interest and could serve as a building block to other constructions.

Definition 2.39 (η-Split-SIS problem ([NZZ15] extended))**.** *Given η matrices $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\beta = (\beta_1, \ldots, \beta_\eta)$ and N, the η-Split-$\mathsf{SIS}_{q,n,m,\beta,N}$ asks to find $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_\eta) \in \mathbb{Z}_q^{m\eta} \setminus \{\mathbf{0}\}$ and $\mathbf{h} = (h_1, \ldots, h_\eta) \in \mathbb{Z}_N^\eta \setminus \{\mathbf{0}\}$ such that $\|\mathbf{x}_i\| \leq \beta_i$ and $\sum_{i=1}^{\eta} h_i \cdot \mathbf{A}_i\mathbf{x}_i = \mathbf{0}$.*

For certain parameters, it was proven in [NZZ15] that the 2-Split-SIS problem is as hard as the corresponding SIS problem in higher dimension $2m$. The following theorem naturally extends this proof to the η matrices configuration.

Theorem 4.2.1. *Let $m = m(n), N = N(n)$ be polynomials, $\beta_1 = \cdots = \beta_\eta = \beta(n)$ and $q \geq \beta \cdot \omega(\sqrt{n \log n}) > N$ a prime integer. Then the η-Split-$\mathsf{SIS}_{q,n,m,\beta,N}$ problem is polynomially equivalent to the $\mathsf{SIS}_{q,n,\eta m,\beta\sqrt{\eta}}$.*

Proof. η-Split-$\mathsf{SIS}_{q,n,m,\beta}$ trivially reduces to $\mathsf{SIS}_{q,n,\eta m,\beta\sqrt{\eta}}$ by splitting any SIS solution into η parts. Now assume there is a probabilistic polynomial time algorithm $\mathcal{A}$ that solves η-Split-$\mathsf{SIS}_{q,n,m,\beta}$ with probability ϵ, and consider an $\mathsf{SIS}_{q,n,\eta m,\beta\sqrt{\eta}}$ instance $\mathbf{A} = [\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_\eta]$. We construct an algorithm $\mathcal{B}$ that solves the $\mathsf{SIS}_{q,n,\eta m,\beta\sqrt{\eta}}$ with probability at least ϵ/N^η. $\mathcal{B}$ starts by sampling $\mathbf{h} = (h_1, \ldots, h_\eta) \xleftarrow{\$} \mathbb{Z}_N^\eta \setminus \{\mathbf{0}\}$, sets $h_i^* = 1$ if $h_i = 0$, h_i otherwise, and $\hat{\mathbf{A}} = [h_1^*\mathbf{A}_1 \mid \cdots \mid h_\eta^*\mathbf{A}_\eta]$. $\mathcal{B}$ uses $\mathcal{A}$ to get $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_\eta) \in \mathbb{Z}_q^{m\eta} \setminus \{\mathbf{0}\}$ and $\hat{\mathbf{h}} = (\hat{h}_1, \ldots, \hat{h}_\eta) \in \mathbb{Z}_N^\eta$ such that $\|\mathbf{x}_i\| \leq \beta$ and $\sum_{i=1}^\eta \hat{h}_i \cdot \mathbf{A}_i\mathbf{x}_i = \mathbf{0}$. Now if $\hat{\mathbf{h}} \neq \mathbf{h}$, $\mathcal{B}$ aborts. Otherwise (with probability $1/N^\eta$), $\mathcal{B}$ outputs $\hat{\mathbf{x}} = (\tilde{h}_1\mathbf{x}_1, \ldots, \tilde{h}_\eta\mathbf{x}_\eta)$, where $\tilde{h}_i \leftarrow 0$ if $h_i = 0$, 1 otherwise. Then $\hat{\mathbf{x}} \neq \mathbf{0}$ and $\|\hat{\mathbf{x}}\| \leq \beta\sqrt{\eta}$, and $\hat{\mathbf{A}}\hat{\mathbf{x}} = \mathbf{0}$. □ □

Definition 2.40 ((k,η)-Split-SIS problem)**.** *Given $k, \eta \in \mathbb{N}$, an instance of the η-Split-SIS together with k linearly independent solutions $\mathbf{x}_1, \ldots, \mathbf{x}_k$ to it, the (k,η)-Split-SIS problem asks for a new solution that does not lie in the span of the previous ones. Formally, for all $i \in \{1, \ldots, k\}$, $\sum_{j=1}^\eta h_j\mathbf{A}_j\mathbf{x}_{i,j} = \mathbf{0}$ and $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_\eta) \notin \mathbb{Q}\text{-span}(\mathbf{x}_1, \ldots, \mathbf{x}_k)$.*

Theorem 4.2.2. *For $q = \tilde{O}(n^3)$, $m \geq 2n\log q$, $\beta = \tilde{O}(n)$, and $k = O(n)$, the (k,η)-Split-$\mathsf{SIS}_{q,n,m,\beta}$ problem is polynomially equivalent to $\mathsf{SIS}_{q,n,\eta(m-k),\beta'}$ for some $\beta' = poly(n)$.*

Proof. The reduction from (k,η)-Split-$\mathsf{SIS}_{q,n,m,\beta}$ to $\mathsf{SIS}_{q,n,\eta(m-k),\beta'}$ is easy: solving SIS implies solving η-Split-SIS (with a little change in dimension), which also implies solving (k,η)-Split-$\mathsf{SIS}_{q,n,m,\beta}$ (if one can solve the problem without hints, he can also solve it given hints).

The other way around is more technical, but essentially results from [NZZ15, Theorem 1], and using [LPSS14, Theorem 11] with a polynomial dimension reduction from ηm to $\eta(m-k)$ using the Rényi Divergence. Indeed, the first step is to reduce the $\mathsf{SIS}_{q,n,\eta(m-k),\beta'}$ to the η-Split-$\mathsf{SIS}_{q,n,m-k,\beta''}$ for some $\beta'' = \tilde{O}(\beta')$ using Theorem 4.2.1. The second one is to apply [LPSS14, Theorem 11] to finish the reduction to (k,η)-Split-$\mathsf{SIS}_{q,n,m,\beta}$ with $\tilde{O}(\beta') = \text{poly}(n)$. □ □

By the reduction induced by Theorem 4.2.2, our traceable signature scheme is secure (in the ROM) based on the worst-case hardness of the SIVP problem on general lattices. The reader is referred to Section 4.4.3 for complete proofs of the claimed security.

Interactive Proofs of Knowledge. For interactive proofs, we denote $\mathcal{P}$ the prover, $\mathcal{V}$ the verifier, and (Cmt, Ch, Rsp) the transcript "commitment - challenge - response". Formal definitions and security models are exposed in Section 4.3.

4.3 Cryptographic Primitives

Proofs of Knowledge. Zero-Knowledge (ZK) Proofs were introduced by Goldwasser, Micali and Rackoff in [GMR88]. Roughly, such proofs allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the veracity of a statement $\mathcal{S}$, without leaking any additional information on the proof but its length. Since its introduction, this kind of proofs has been widely applied to digital identification protocols, and by extension signature schemes using the Fiat-Shamir paradigm [FS86]. These proofs satisfy the following three properties:

— *Completeness:* If $\mathcal{S}$ is true, the honest verifier will be convinced except with negligible probability,

— *Soundness:* If $\mathcal{S}$ is false, no cheating prover can convince the honest verifier that it is true except with negligible probability,

— *Zero-knowledge:* Anything that is feasibly computable from the proof is also feasibly computable from the assertion itself.

A classical variant of ZK proofs generally used is *Witness-Indistinguishable* (WI) proofs, where the ZK requirement is roughly weakened into "seeing the proof does not provide any information on the witness used". More formally, assuming a language $\mathcal{L}$ and witnesses w_0, w_1 satisfying the language, WI requires distributions D_0 and D_1 to be indistinguishable where $D_b = \{(w_0, w_1, \mathsf{Prove}(w_b, \mathcal{L}; \rho))\,,$ for random strings $\rho\}$, while not requiring proof simulatability anymore.

Nevertheless, our construction is going to need somewhat opposite requirements: we want to be able to build a simulator capable of simulating proofs, while allowing anyone possessing a witness in the language to test if a given proof was generated using this witness. Proofs of knowledge meeting these requirements are called *Testable weak Zero-Knowledge* (TwZK for short) and were introduced in [ABCG15]. The authors notice there that deterministic instances of Stern's protocol belong to this type of proofs (see App. 4.A for a complete proof). Indeed, knowing the witness being used allows to recompute the inner part of the proof and check the consistency. The general idea behind TwZK is that there is a little information leaking compared to classical ZK: the fact that an attacker can test whether a particular witness was used in a proof. Now if the set of witnesses is either too large or if it is difficult to find a witness, the underlying security model shows that this information can only be used to test a particular known witness. If the witness is not known, an attacker should either test all possible witnesses or find a particular witness, which in both cases is hard. Hence an attacker who breaks the security of our scheme either breaks ROM security or is able to find the correct witness. Concretely, in case of a signature scheme derived through Fiat-Shamir, for the original Stern scheme it means that an attacker is able to find a given vector of small weight associated to a given syndrome, in a very large set of such possible vectors, and hence he is able to solve the Syndrome Decoding problem. The following is a formal definition of the new concept of [ABCG15].

Definition 3.41 (Testable weak Zero-Knowledge Proofs)**.** *A Testable weak Zero-Knowledge Proof of Knowledge is defined through the following algorithms:*

— $\mathsf{Setup}(1^\lambda)$*: Generates the public parameters of the system, possibly with a simulation trapdoor* stk*.*

— $\mathsf{Prove}(w, \mathcal{L}; \rho)$*: Generates a proof* π *that a word* w *is in the language* $\mathcal{L}$ *using some random string* ρ*.*

— $\mathsf{Verif}(\pi, \mathcal{L})$*: Checks the validity of a proof* π *with respect to the language* $\mathcal{L}$*.*

— $\mathsf{TestWit}(w, \pi, \mathcal{L})$*: Checks if a valid proof* π *was generated using the word* $w \in \mathcal{L}$*.*

— $\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \rho)$ *Generates a simulated proof of knowledge of a word in* $\mathcal{L}$ *using trapdoor* stk *and random string* ρ*.*

In addition to classical correctness and soundness properties, we want weak indistinguishability of proof simulation:

For a non empty language $\mathcal{L}$ a proof generated using a witness (case S_0), is indistinguishable from one using the trapdoor (case S_1). $S_0 = \{\mathsf{Prove}(w, \mathcal{L}; \rho) | w \in \mathcal{L}\}, S_1 = \{\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \rho)\}$.

$\mathsf{Exp}^{\mathsf{wS}-b}_{\mathcal{P},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $\mathsf{IF}(b = 0)$, $\pi^* \xleftarrow{\$} \{\mathsf{Prove}(w, \mathcal{L}; \rho)\}$
3. ELSE $\pi^* \xleftarrow{\$} \{\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \rho)\}$
4. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \pi^*, \mathsf{TestWit}(.))$
5. $\mathtt{RETURN}$ b'

Compared to what might be expected for Classical Zero-Knowledge, the previous property can only be achieved for some languages. Namely those where guessing the correct witness associated with a proof is hard. (Either because there is no efficient way to search through the set of witnesses, or simply because witnesses are hard to sample). The proof that the original Stern protocol meets our requirements is given in Appendix 4.A. (Notice that our results show that the original Stern protocol is actually not ZK but TwZK, although straightforward randomization can make it ZK.)

A lattice-based Stern protocol. We recall for completeness the extended lattice-based Stern protocol from [LNSW13]. One round of the protocol is described in Figure 4.1, as well as the verification process. In the following, COM denotes a commitment scheme (see [LNSW13] for details), any collision-resistant hash function could possibly be used. Given public inputs $(\mathbf{M}, \mathbf{t}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ and secret $\mathbf{s}$ such that $\|\mathbf{s}\| \leq \beta$ and $\mathbf{Ms} = \mathbf{t}$, it allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that the former knows such an $\mathbf{s}$, *without* disclosing it. Matrix $\mathbf{M}$ is first extended to $\mathbf{M}' \in \mathbb{Z}_q^{n\times 3m}$ by concatenating $2m$ zero columns to the original matrix. The witness is then processed to very short vectors in B_{3m} : the set of $3m$-dimensional trinary vectors with the same number (m) of entries -1, 0, and 1. In order to get a transformed extended witness in B_{3m}, we need the two following randomized routines:

— Decompose, which on input short $\mathbf{s} \in \mathbb{Z}_q^m$, decompose $s_i = \sum_{j=0}^{k-1} 2^j b_{i,j}$ with $b_{i,j} \in \{-1, 0, 1\}$ and returns the k vectors $\tilde{\mathbf{s}}_j = (b_{0,j}, \ldots, b_{m-1,j})$ such that $\mathbf{s} = \sum_{j=0}^{k-1} 2^j \tilde{\mathbf{s}}_j$.

— Extend, which on input $\mathbf{S} = \mathsf{Decompose}(\mathbf{s}) \in \{-1, 0, 1\}^{k\times m}$, concatenates $2m$ k-rows columns to $\mathbf{S}$ so that each row of the resulting matrix $\mathbf{S}'$ lies in B_{3m}. Hence, we have the relation:

$$\mathbf{M}' \sum_{i=0}^{k-1} 2^i \mathbf{S}'_i = \mathbf{t} \pmod q \quad \Leftrightarrow \quad \mathbf{Ms} = \mathbf{t} \pmod q.$$

In the SternExt protocol, the above functions Extend and Decompose are used to pre-process any short input secret to prove into a sequence of very short vectors in B_{3m}. The SternExt protocol is therefore called with inputs $\mathbf{S}' = \mathsf{Extend}(\mathsf{Decompose}(\mathbf{s}))$, extended matrix $\mathbf{M}'$ (with $2m$ zero columns from $\mathbf{M}$), and syndrome $\mathbf{t}$. At this point, the careful reader could have noticed that if the SternExt protocol were using *deterministic* versions of the Extend and Decompose routines, the resulting protocol would be TwZK in two cases over three. Indeed assuming such versions of the routines, if $\textsc{Ch} = 1$, $\mathcal{V}$ (who checks $\mathbf{c}_0$ and $\mathbf{c}_2$) can additionally check whether $\mathbf{c}_1 = \mathrm{COM}\left(\pi_0(\mathbf{z}_0 - \mathbf{s}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{z}_{k-1} - \mathbf{s}_{k-1})\right)$ or not. If $\textsc{Ch} = 2$, $\mathcal{V}$ (who checks $\mathbf{c}_0$ and $\mathbf{c}_1$) can also check if $\mathbf{c}_2 = \mathrm{COM}\left(\pi_0(\mathbf{r}_0) + \pi_0(\mathbf{s}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{r}_{k-1}) + \pi_{k-1}(\mathbf{s}_{k-1})\right)$.

Following the last remark, we now describe the aforementioned modified SternExt protocol.

SternExt revisited. In order to provide all the aforementioned features, our traceable signature scheme requires two matrices. Therefore, we modify the SternExt protocol to support multiple matrices in the protocol. Instead of performing two independent instances of the SternExt protocol (which would correspond to two independent secrets $\mathbf{x}$ and $\mathtt{tk}$), we link those instances together by appending commitments relative to each part of a longer secret $(\mathbf{x}, \mathtt{tk})$.

The modifications we propose constitute our modified protocol (which we call mSternExt), and is described in Fig. 4.2.

1. [**Commitment**] $\mathcal{P}$ samples k couples $(\mathbf{r}_i, \pi_i) \xleftarrow{\$} \mathbb{Z}_q^{3m} \times \mathcal{S}_{3m}$ and sends $\textsc{Cmt} = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$ to $\mathcal{V}$, where:
$$\mathbf{c}_0 \leftarrow \mathrm{COM}\left(\pi_0 \mid \cdots \mid \pi_{k-1} \mid \mathbf{M}'\left(\textstyle\sum_{i=0}^{k-1} 2^i \mathbf{r}_i\right) \mod q\right),$$
$$\mathbf{c}_1 \leftarrow \mathrm{COM}\left(\pi_0(\mathbf{r}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{r}_{k-1})\right), \text{ and}$$
$$\mathbf{c}_2 \leftarrow \mathrm{COM}\left(\pi_0(\mathbf{s}_0 + \mathbf{r}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{s}_{k-1} + \mathbf{r}_{k-1})\right)$$
2. [**Challenge**] On receiving $\textsc{Cmt}$, $\mathcal{V}$ generates and sends $\textsc{Ch} \xleftarrow{\$} \{0, 1, 2\}$ to $\mathcal{P}$.
3. [**Response**] $\mathcal{P}$ generates $\textsc{Rsp}$ as below and finally sends it to $\mathcal{V}$.

$\textsc{Ch} = 0$	$\textsc{Ch} = 1$	$\textsc{Ch} = 2$
For $i \in \{0, \ldots, k-1\}$, $\mathbf{v}_i \leftarrow \pi_i(\mathbf{s}_i)$, $\mathbf{w}_i \leftarrow \pi_i(\mathbf{r}_i)$ $\textsc{Rsp} \leftarrow (\mathbf{c}_1, \mathbf{c}_2, \{\mathbf{v}_i\}, \{\mathbf{w}_i\})$	For $i \in \{0, \ldots, k-1\}$, let $\mathbf{z}_i \leftarrow \mathbf{s}_i + \mathbf{r}_i$. $\textsc{Rsp} \leftarrow (\mathbf{c}_0, \mathbf{c}_2, \{\pi_i\}, \{\mathbf{z}_i\})$	$\textsc{Rsp} \leftarrow (\mathbf{c}_0, \mathbf{c}_1, \{\pi_i\}, \{\mathbf{r}_i\})$

Verification: $\mathcal{V}$ outputs Accept if the following holds:
- If $\textsc{Ch} = 0$, $\mathbf{v}_i \in \mathrm{B}_{3m}$ $\forall i \in \{0, \ldots, k-1\}$, $\mathbf{c}_1 = \mathrm{COM}\left(\mathbf{w}_0 \mid \cdots \mid \mathbf{w}_{k-1}\right)$, and $\mathbf{c}_2 = \mathrm{COM}\left(\mathbf{v}_0 + \mathbf{w}_0 \mid \cdots \mid \mathbf{v}_{k-1} + \mathbf{w}_{k-1}\right)$,
- If $\textsc{Ch} = 1$, $\mathbf{c}_0 = \mathrm{COM}\left(\pi_0 \mid \cdots \mid \pi_{k-1} \mid \mathbf{M}'\left(\sum_{i=0}^{k-1} 2^i \mathbf{z}_i\right) - \mathbf{t} \mod q\right)$, and $\mathbf{c}_2 = \mathrm{COM}\left(\pi_0(\mathbf{z}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{z}_{k-1})\right)$,
- If $\textsc{Ch} = 2$, $\mathbf{c}_0 = \mathrm{COM}\left(\pi_0 \mid \cdots \mid \pi_{k-1} \mid \mathbf{M}'\left(\sum_{i=0}^{k-1} 2^i \mathbf{r}_i\right) \mod q\right)$, and $\mathbf{c}_1 = \mathrm{COM}\left(\pi_0(\mathbf{r}_0) \mid \cdots \mid \pi_{k-1}(\mathbf{r}_{k-1})\right)$,

Otherwise, $\mathcal{V}$ outputs Reject.

FIGURE 4.1 – The interactive $\mathsf{SternExt}(\mathbf{S}', \mathbf{M}')$ proof system.

Rényi divergence. Similarly to the results of Micciancio and Regev showing that discrete gaussians better suit lattices than the uniform distribution [MR07], recent results using the Rényi Divergence (RD) tend to prove the same compared to the statistical distance (see [BLL+15] for instance). For two distributions D_1 and D_2 such that $\mathsf{supp}(D_1) \subset \mathsf{supp}(D_2)$ over a countable domain D, their RD (of order 2) is defined by $\mathrm{RD}(D_1 \,||\, D_2) = \sum_{x \in X} \frac{D_1(x)^2}{D_2(x)}$. (The reader is referred to [BLL+15] and [LPSS14] for more details on RD.)

As mentioned before, our construction uses two instances of the SternExt protocol: one to ensure traceability and the other one for non-frameability. For this latter property to hold, an adversary – even helped with corrupted users – should not be able to come up with a valid secret signing key that would trace outside the group or worse, to an honest user. Corrupted signing keys are solutions to an SIS problem and therefore, an adversary exploiting corrupted users has to solve this instance using k hints for this instance, *i.e.* he has to solve a k-SIS instance. Boneh and Freeman first showed a reduction from solving SIS to solving k-SIS [BF11], giving intuition about the entropy contained in those hints but unfortunately, their reduction involved an exponential dimension reduction in the number of hints, yielding easy SIS problems for values of k one is willing to consider in real-life applications.

Last year at CRYPTO'14, Ling *et al.* presented a similar result for the Learning With Errors problem [LPSS14], where they achieve to reduce the dimension loss from exponentially to polynomially, thanks to the RD. As their result also applies to the SIS problem we base non-frameability upon, the RD now allows us to handle polynomially many users inside the group instead of a constant number previously.

1. [**Commitment**] $\mathcal{P}$ samples k tuples $(\phi_i, \psi_i) \xleftarrow{\$} \mathcal{S}_{3m} \times \mathcal{S}_{3m}$, $(\mathbf{u}_i, \mathbf{v}_i) \xleftarrow{\$} \mathrm{B}_{3m} \times \mathrm{B}_{3m}$. For short, we denote $\mathbf{sums} = \left\{\sum_{i=0}^{k-1} \mathbf{P}2^i\mathbf{u}_i + \mathbf{B}'2^i\mathbf{v}_i\right\}_{i=0}^{k-1}$. $\mathcal{V}$ sends $\textsc{Cmt} = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)$ to $\mathcal{V}$, where:

$$\begin{aligned}
\mathbf{a}_0 &= \mathrm{COM}(\{\phi_i\} \mid \mathbf{sums} \mid \mu) \\
\mathbf{a}_1 &= \mathrm{COM}(\{\phi_i(\mathbf{u}_i)\} \mid \mu) \\
\mathbf{a}_2 &= \mathrm{COM}(\{\phi_i((\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i) + \mathbf{u}_i)\} \mid \mu)
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}_0 &= \mathrm{COM}(\{\psi_i\} \mid \mathbf{sums} \mid \mu) \\
\mathbf{b}_1 &= \mathrm{COM}(\{\psi_i(\mathbf{v}_i)\} \mid \mu) \\
\mathbf{b}_2 &= \mathrm{COM}(\{\psi_i(\mathbf{x}_i + \mathbf{v}_i)\} \mid \mu)
\end{aligned}$$

2. [**Challenge**] On receiving $\textsc{Cmt}$, $\mathcal{V}$ generates and sends $\textsc{Ch} \xleftarrow{\$} \{0, 1, 2\}$ to $\mathcal{P}$.
3. [**Response**] $\mathcal{P}$ generates $\textsc{Rsp}$ as below and finally sends it to $\mathcal{V}$.

$\textsc{Ch} = 0$	$\textsc{Ch} = 1$	$\textsc{Ch} = 2$
$\textsc{Rsp} \leftarrow$	$\textsc{Rsp} \leftarrow$	$\textsc{Rsp} \leftarrow$
$(\{\phi_i, \psi_i, \mathbf{u}_i, \mathbf{v}_i\})$	$(\{\phi_i, \psi_i, (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i) + \mathbf{u}_i, \mathbf{x}_i + \mathbf{v}_i\})$	$(\{\phi_i((\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i)), \phi_i(\mathbf{u}_i), \psi_i(\mathbf{x}_i), \psi_i(\mathbf{v}_i)\})$

Verification: $\mathcal{V}$ outputs Accept if the following holds:

- If $\textsc{Ch} = 0$: $\mathbf{u}_i, \mathbf{v}_i \in \mathrm{B}_{3m}$, and commitments $\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_0$, and $\mathbf{b}_1$ were generated according to step 1.
- If $\textsc{Ch} = 1$: $\mathbf{a}_0, \mathbf{a}_2, \mathbf{b}_0$, and $\mathbf{b}_2$ were generated according to step 1.
- If $\textsc{Ch} = 2$: $\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1$, and $\mathbf{b}_2$ were generated according to step 1.

Otherwise, $\mathcal{V}$ outputs Reject.

FIGURE 4.2 – The modified interactive $\mathsf{mSternExt}(\mathbf{B}', \mathbf{P}, \mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i), \mathbf{X}'_i, \mu)$ protocol which proves the knowledge of short vectors $(\mathbf{x}_1, \mathbf{x}_2), \mathbf{e}$ and $\mathbf{x}$ such that $\mathbf{P}(\mathbf{x}_1, \mathbf{x}_2 - \mathbf{e}) + \mathbf{B}\mathbf{x} = \mathbf{s}$

Non-leaking NTRUSign. Among secure fixes of the NTRUSign signature scheme, we choose to build our new traceable signature scheme upon the one of Aguilar *et al.* [ABDG14] (see chapter 3). It benefits from a rather good efficiency given the security reduction, it has concrete parameters (which unfortunately is rather scarce for lattice-based schemes), and uses most of the rejection sampling framework of [Lyu12] which ensures that issued signatures follows the expected distribution. We denote this NTRUSign fix by mNTRUSign, and its routine by mNTRUSign.KeyGen, mNTRUSign.Sign and mNTRUSign.Verify.

4.4 A Non-Frameable Lattice-Based Traceable Signature Scheme

4.4.1 Formal Model

Traceable signatures were introduced by Kiayias, Tsiounis and Yung in [KTY04] as an improvement of group signatures (defined in [Cv91]). In addition to the classical properties of a group signature scheme, that allows users to sign in the name of the group, while the Opener only is able to trace back the actual signer, traceable signatures allow the Opener to delegate the tracing decision for a specific user without revoking the anonymity of the other users: the Opener can delegate its tracing capability to sub-openers, but against specific

signers without letting them trace other users. This gives two crucial advantages: on the one hand tracing agents (sub-openers) can run in parallel ; on the other hand, honest users do not have to fear for their anonymity if authorities are looking for signatures produced by misbehaving users only. This is in the same vein as searchable encryption [ABC+05], where a trapdoor, specific to a keyword, allows to decide whether a ciphertext contains this keyword or not, and provides no information about ciphertexts related to other keywords.

In a traceable signature scheme, there are several users and two authorities: the *Group Manager*: it issues certificates for users to grant access to the group, and the *Opener*: it is able to *open* or *trace* any signature. The former means that it can learn who is the actual signer of a given signature while the latter decides, on a given signature and an alleged signer, whether the signature has really been generated by this signer or not. It is also able to delegate the latter tracing capability but for specific users only. To this aim, it *reveals* a trapdoor to a *Sub-Opener*. The latter gets the ability to trace a specific user only (decide whether the signer associated to the trapdoor is the actual signer of a signature) without learning anything about the other users.

In the following, those two authorities are going to be the same party, but the sub-tracing capacities can be delegated to anyone. Also notice that many security models consider an additional party called "Judge" that verifies all the claims. As anyone will be able to directly verify the claims (*i.e.* play the role of this Judge), we omit this detail for the sake of clarity.

A traceable signature scheme (with stepping capacities) is defined by a sequence of (interactive) protocols $\mathcal{TS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verif}, \mathsf{Open}, \mathsf{Reveal}, \mathsf{Trace}, \mathsf{Claim})$. While Setup, Join, Sign, Verif, Open are similar to those in Group Signature schemes, Reveal, Trace, Claim are new. All these algorithms are summed up in Figure 4.1 :

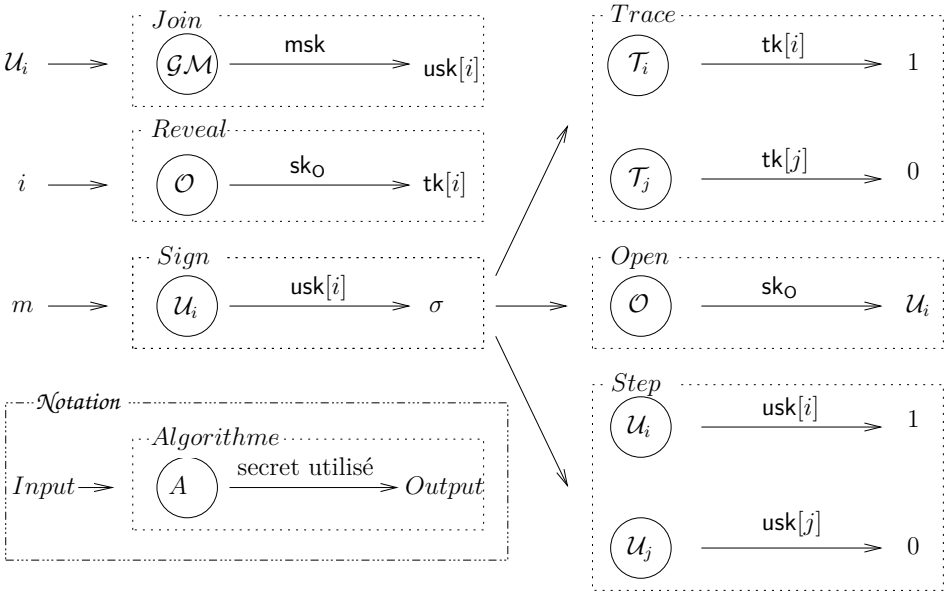


FIGURE 4.1 – Algorithms in Traceable Signature Schemes.

— $\mathsf{Setup}(1^\lambda)$: It generates the global parameters, the group public key pk and the private keys: the master secret key msk for the group manager, and the opening key sk_O for the Opener.

— $\mathsf{Join}(\mathcal{U}_i)$: This is an interactive protocol between a user $\mathcal{U}_i$ and the group manager (using his private key msk). At the end of the protocol, the user obtains a signing key $\mathsf{usk}[i]$,

and the group manager adds the user to the registration list, storing some information in the registered information table *Reg*. We note $\mathcal{S}$ the set of indices of registered users.

— $\mathsf{Sign}(\mathsf{pk}, i, \mu, \mathsf{usk}[i])$: This is a protocol expected to be made by a registered user $i \in \mathcal{S}$, using his own signing key. It produces a signature σ on the message μ.

— $\mathsf{Verif}(\mathsf{pk}, \mu, \sigma)$: Anybody should be able to verify the validity of the signature, with respect to the public key pk. This algorithm thus outputs 1 iff the signature is valid.

— $\mathsf{Open}(\mathsf{pk}, \mu, \sigma, \mathsf{sk_O})$: If σ is valid, the Opener, using $\mathsf{sk_O}$, outputs a user i assumed to be the author of the signature with a publicly verifiable proof Π_O of this accusation.

— $\mathsf{Reveal}(\mathsf{pk}, i, \mathsf{sk_O})$: This algorithm, with input $\mathsf{sk_O}$ and a target user i, outputs a tracing key $\mathsf{tk}[i]$ specific to the user i, together with a proof $\Pi_{\mathcal{E}}$ confirming this $\mathsf{tk}[i]$ is indeed a tracing key of the user i.

— $\mathsf{Trace}(\mathsf{pk}, \mu, \sigma, \mathsf{tk}[i])$: Using the sub-opener key $\mathsf{tk}[i]$ for user i, this algorithm outputs 1 iff σ is a valid signature produced by i, together with a proof Π_{sO} confirming the decision.

— $\mathsf{Claim}(\mathsf{pk}, \mu, \sigma, \mathsf{usk}[i])$: Using the user's secret key $\mathsf{usk}[i]$, this algorithm outputs 1 iff σ if a valid signature produced by i, together with a proof Π_c confirming the claim.

Correctness. This notion guarantees that honest users should be able to generate valid signatures.

In the following experiments that formalize the security notions, the adversary can run the Join protocol, either passively (receives only public values, as seen by an eavesdropper) or actively (he chooses and receives all the values, as the legitimate user):

— either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the secret keys: the index i is added to the HU (Honest Users) list ;

— or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user it will control: the index i is added to the CU (Corrupted Users) list.

The adversary is given the master secret key (the group manager is corrupted), and so does not need access to the joinA oracle since it can simulate it by itself, to create corrupted users (that are not necessarily in CU). After a user is created, the adversary plays the role of corrupted users, and can interact with honest users, granted the previous oracles, and also new ones:

— $\mathsf{open}(\mu, \sigma)$, if (μ, σ) is valid, returns the identity i of the signer. Then (i, μ, σ) is appended to the list $\mathcal{O}$ of opened signatures ;

— $\mathsf{reveal}(i)$, if $i \in \mathsf{HU}$, returns the tracing key $\mathsf{tk}[i]$ for the user i. Then i is appended to the list $\mathcal{R}$ of the revealed users ;

— $\mathsf{tr}(i, \mu, \sigma)$, if $i \in \mathsf{HU}$ and (μ, σ) is valid, returns 1 iff i is the signer who made σ on μ is i. Then (i, μ, σ) is appended to the list $\mathcal{T}$ of traced signatures ;

— $\mathsf{step}(i, \mu, \sigma)$, if $i \in \mathsf{HU}$, plays as the honest user i would do to step in/out of the signature σ on message μ.

For a corrupted user i, with the secret key $\mathsf{sk}i$, the adversary can run itself the Claim and Trace procedures, and or course sign too. We thus have the following sets:

— $\mathcal{S}$, the set of registered users, which is the disjunction of HU and CU the honest and dishonest users respectively ;

$\mathsf{Experiment}\ \mathsf{Exp}^{\mathsf{Misl}}_{\mathcal{TS},\mathcal{A}}(\lambda)$
1. $(\mathsf{pk}, \mathsf{msk}, \mathsf{sk_O}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mu, \sigma) \leftarrow \mathcal{A}(\mathsf{pk} : \mathsf{joinP}, \mathsf{joinA}, \mathsf{corrupt}, \mathsf{sign}, \mathsf{reveal})$
3. `IF` $\mathsf{Verif}(\mathsf{pk}, \mu, \sigma) = 0$, `RETURN` 0
4. `IF` $\mathsf{Open}(\mathsf{pk}, \mu, \sigma, \mathsf{sk_O}) = \perp$, `RETURN` 1
5. `IF` $\exists j \notin \mathsf{CU} \cup \mathfrak{S}[\mu]$, $\mathsf{Open}(\mathsf{pk}, \mu, \sigma, \mathsf{sk_O}) = (j, \Pi)$ `RETURN` 1
6. `ELSE RETURN` 0

$$\mathsf{Adv}^{\mathsf{Misl}}_{\mathcal{TS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{Misl}}_{\mathcal{TS},\mathcal{A}}(\lambda) = 1]$$

FIGURE 4.2 – Misidentification for Soundness

$\mathsf{Experiment}\ \mathsf{Exp}^{\mathsf{nf}}_{\mathcal{TS},\mathcal{A}}(\lambda)$
1. $(\mathsf{pk}, \mathsf{msk}, \mathsf{sk_O}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mu, \sigma) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{msk}, \mathsf{sk_O} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sign})$
3. `IF` $\mathsf{Verif}(\mathsf{pk}, \mu, \sigma) = 0$ `RETURN` 0
4. `IF` $\exists i \in \mathsf{HU} \setminus \mathfrak{S}[\mu]$, $\mathsf{Open}(\mathsf{pk}, \mu, \sigma, \mathsf{sk_O}) = (i, \Pi)$ `RETURN` 1
5. `ELSE RETURN` 0

$$\mathsf{Adv}^{\mathsf{nf}}_{\mathcal{TS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{nf}}_{\mathsf{nf},\mathcal{A}}(\lambda) = 1]$$

FIGURE 4.3 – Non-Frameability for Soundness

— $\mathfrak{S}$, the list of generated signatures (i, μ, σ), and $\mathfrak{S}[\mu] = \{i | (i, \mu, \sigma) \in \mathfrak{S}\}$;
— $\mathcal{O}$, the list of opened signatures (i, μ, σ), and $\mathcal{O}[\mu] = \{i | (i, \mu, \sigma) \in \mathcal{O}\}$;
— $\mathcal{R}$, the list of revoked users i ;
— $\mathcal{T}$, the list of traced signatures (i, μ, σ), and $\mathcal{T}[\mu] = \{i | (i, \mu, \sigma) \in \mathcal{T}\}$.

A signature is identified by a user-message pair and not σ itself in those sets because we do not focus on strong unforgeability. The subsequent relaxation on the security has already been used in [LY09] for traceable signatures.

Soundness. This is the main security notion that defines two unforgeability properties. The security games are shortened thanks to the correctness which implies that the opening, the tracing and the stepping processes are consistent:

— Misidentification, which means that the adversary should not be able to produce a non-trivial valid signature that could not be opened to a user under its control. The adversary wins if Open either accuses an unknown user or has an invalid proof, so returning $\perp$ (see Figure 4.2) ;
— Non-Frameability, which means that the adversary should not be able to produce a non-trivial valid signature corresponding (that opens) to an honest user even if the authorities are corrupted (see Figure 4.3) ;

$\mathcal{TS}$ is *Sound* if, for any polynomial adversary $\mathcal{A}$, both advantages $\mathsf{Adv}^{\mathsf{Misl}}_{\mathcal{TS},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{nf}}_{\mathcal{TS},\mathcal{A}}(\lambda)$ are negligible.

The first notion (Misidentification) guarantees traceability (the Open algorithm always succeeds on valid signatures) but also honest users cannot be framed when the group manager is honest. The second one (non-frameability) is somewhat stronger since it allows the group manager to be corrupted, but would not guarantee by itself traceability.

Anonymity. We now address the privacy concerns. For two distinct signers i_0, i_1, chosen by the adversary, the latter should not have any significant advantage in guessing if the issued signature comes from i_0 or i_1. We can consider either a quite strong anonymity notion (usually named full-anonymity) where the adversary is allowed to query the opening oracle (resp. tracing, stepping) on any signatures, except signatures that are equivalent to the challenge signature with respect to the signers i_0 or i_1 ; or the classical anonymity notion,

Security level λ (bits)	100	112	128	160	192
Number of rounds t_λ	64	71	81	101	122

TABLE 4.1 – Number of rounds as a function of the security parameter.

where open, tr and of course reveal are not available to the adversary. $\mathcal{TS}$ is *anonymous* if, for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{TS},\mathcal{A}}(\lambda)$ is negligible (see Figure 4.4).

Experiment $\mathsf{Exp}^{\mathsf{anon}_b}_{\mathcal{TS},\mathcal{A}}(\lambda)$
1. $(\mathsf{pk}, \mathsf{msk}, \mathsf{sk}_\mathsf{O}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mu, i_0, i_1) \leftarrow \mathcal{A}(\texttt{FIND}, \mathsf{pk}, \mathsf{msk} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sign}, \mathsf{open}^*, \mathsf{tr}^*, \mathsf{reveal}^*, \mathsf{step}^*)$
3. $\sigma \leftarrow \mathsf{Sign}(\mathsf{pk}, i, \mu, \mathsf{usk}[i_b])$
4. $b' \leftarrow \mathcal{A}(\texttt{GUESS}, \sigma : \mathsf{joinA}, \mathsf{joinP}, \mathsf{corrupt}, \mathsf{sign}, \mathsf{open}^*, \mathsf{tr}^*, \mathsf{reveal}^*, \mathsf{step}^*)$
5. `IF` $i_0 \notin \mathsf{HU} \setminus (\mathcal{R} \cup \mathcal{T}[\mu] \cup \mathcal{O}[\mu] \cup \mathfrak{S}[\mu])$ `OR` $i_1 \notin \mathsf{HU} \setminus (\mathcal{R} \cup \mathcal{T}[\mu] \cup \mathcal{O}[\mu] \cup \mathfrak{S}[\mu])$ `RETURN` 0
6. `ELSE RETURN` b'

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{TS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{anon}-1}_{\mathcal{TS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{anon}-0}_{\mathcal{TS},\mathcal{A}}(\lambda) = 1]$$

FIGURE 4.4 – Anonymity ($\mathsf{open}^* = \mathsf{tr}^* = \mathsf{reveal}^* = \mathsf{step}^* = \emptyset$) vs Full-Anonymity ($\mathsf{open}^* = \mathsf{open}, \mathsf{tr}^* = \mathsf{tr}, \mathsf{reveal}^* = \mathsf{reveal}, \mathsf{step}^* = \mathsf{step}$)

In the following however, our scheme will only fulfill the anonymity requirement, as the Claim algorithm disables stronger anonymity notions.

As explained in [KTY04], the user tracing (combined with the GM publishing the user's *tracing trapdoor*) can be used to implement a type of *CRL-based revocation* that nullifies all signatures by a private key. This corresponds to the Verifier-Local Revocation strategy, which seems appropriate in the PKI context, in this case, the verification algorithm is extended by also checking, that the signature does not trace back to a revocated key.

4.4.2 Presentation of the scheme

As mentioned earlier, the main building block of our construction is the lattice analog to Stern's protocol, first introduced in [LNSW13] and which we modify into the one depicted in Figure 4.2, using 3 parallel instances of original Stern protocol. These parallel instances *do not* change the cheating probability 1/3, thereby involving t_λ times the protocol has to be repeated to ensure soundness with overwhelming probability (namely $1 - 2^{-\lambda}$), see Table 4.1.

In the algorithms depicted below, mSternExt will refer to the Fiat-Shamir transformation of our modified Stern Extension protocol introduced in Figure 4.2.

Algorithms. We now describe the core algorithms of the new lattice-based traceable signature scheme we propose. The Setup Algorithm 10 is intended to produce public parameters compatible with both mNTRUSign from [ABDG14] and the reductions from [LPSS14] and [NZZ15]. Contrarily to all the previous frameable lattice-based schemes, the KeyGen Algorithm 11 only outputs the group keys gpk and gsk. Users' secret signing key generation is delegated to the interactive Join Algorithm 12. The Sign Algorithm 13 takes a user's secret

signing key $\mathtt{sk}_i = (\mathbf{x}_i, \mathtt{tk}_i)$ as input, and processes the $\mathbf{x}_i$ part (using Decompose and Extend) before using the FS version of the mSternExt Fig. 4.2. The Verif Algorithm 14 ensures the good execution of the resulting non-interactive previous signing algorithm. Finally, the Open Algorithm 15 uses the tracing key $\mathtt{tk}_i$ part to verify the TwZK property described Sec. 4.3 to recover the signature's issuer with overwhelming probability.

The group public key is the couple $(\mathbf{B}, \mathbf{P}) \in \mathbb{Z}_q^{N\times 2N} \times \mathbb{Z}_q^{N\times 2N}$ and a secret signing key is of the form $(\mathbf{x}_i, \mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i)) \in \mathbb{Z}_q^{2N} \times \mathbb{Z}_q^{2N}$. Throughout the following algorithms, we assume that public matrix $\mathbf{B}$ and syndrome $\mathbf{s}$ are obtained using $\mathcal{M}_{N\times 2N}(\rho_{\mathbf{B}}, \mathbb{Z}_q)$ and $\mathcal{M}_{N\times 1}(\rho_{\mathbf{s}}, \mathbb{Z}_q)$ respectively.

Algorithme 10 : $\mathcal{GS}$.Setup(λ)

Input : λ the bit security level
Output : $\mathtt{params} = (N, q, d, \mathcal{N}, \nu)$, the global parameters of the scheme (see [ABDG14, Sec. 5.2])

1 Let $\rho_{\mathbf{B}}, \rho_{\mathbf{s}} \xleftarrow{\$} \{0,1\}^\lambda$ be the seeds for matrix $\mathbf{B}$ and vector $\mathbf{s}$
2 **return** $\mathtt{params} = (N, q, d, \mathcal{N}, \nu, \rho_{\mathbf{B}}, \rho_{\mathbf{s}})$

Algorithme 11 : $\mathcal{GS}$.KeyGen(params) (to be run by $\mathcal{GM}$)

Input : params, the global parameters of the scheme
Output : (msk, mpk), $\mathcal{GM}$'s master secret and public keys

1 $(\mathbf{P}, \mathbf{S}) \leftarrow$ mNTRUSign.KeyGen(params) /* see [ABDG14, Alg. 7] */
2 **return** $(\mathtt{msk} = \mathbf{S}, \mathtt{mpk} = (\mathbf{P}, \rho_{\mathbf{B}}, \rho_{\mathbf{s}}))$

Algorithme 12 : $\mathcal{GS}$.Join($\mathcal{U}_i, \mathcal{GM}$) (Interactive protocol between $\mathcal{U}_i$ and $\mathcal{GM}$)

Input : msk, mpk, and params
Output : $\mathtt{tk}_i$, $\mathcal{U}_i$'s tracing key for $\mathcal{GM}$

1 $\mathcal{U}_i$ generates $\mathbf{x} \in \{-d, \ldots, 0, \ldots, d\}$ and sends $\mathbf{Bx}$ to $\mathcal{GM}$
2 $\mathcal{GM}$ computes $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i}, \mathbf{e}_i) =$ mNTRUSign.Sign($\mathbf{s} - \mathbf{Bx}_i$) and sends $\mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i)$ to $\mathcal{U}_i$ /* see [ABDG14, Alg. 8] */
3 $\mathcal{U}_i$ appends $\mathtt{tk}_i$ to his secret key: $\mathtt{sk}_i = (\mathbf{x}_i, \mathtt{tk}_i)$
4 $\mathcal{GM}$ appends $(\mathbf{Bx}_i, \mathtt{tk}_i)$ to the tracing keys list $\mathcal{TK}$
5 **return** $\mathtt{tk}_i$

Remark 1. *As mentioned earlier, it is important to notice that each tracing key is a mNTRUSign signature that has passed the rejection sampling hence, whose norm is controlled and designed for the Decompose routine of the (m)SternExt protocol.*

Algorithme 13 : $\mathcal{GS}$.Sign($\mathtt{sk}_i, \mathtt{mpk}$) (to be run by $\mathcal{U}_i$)

Input : $\mu \in \{0,1\}^*$, the message to be signed
Output : $\pi(\mathbf{x}_i, \mathtt{tk}_i)$, a proof of knowledge on small $\mathbf{x}_i, \mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i)$ such that $\mathbf{Bx}_i + \mathbf{P}(\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i) = \mathbf{s}$

1 Let $\mathbf{B}'$ be the matrix $\mathbf{B}$ extended with $2m$ zero columns, and $\mathbf{X}'_i =$ Extend(Decompose($\mathbf{x}_i$))
2 $\pi \leftarrow$ mSternExt($\mathbf{B}', \mathbf{P}, \mathtt{tk}_i = (\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i), \mathbf{X}'_i, \mu$) /* see Fig. 4.2 */
3 **return** π

Algorithme 14 : $\mathcal{GS}.\mathsf{Verify}(\mathtt{mpk}, \pi, \mu)$

Input : π a candidate signature on message μ
Output : `Accept` if π is a valid signature on message μ, `Reject` otherwise

for $\mathtt{tk}_i \in \mathcal{RVK}$
  if $\mathtt{tk}_i = ((\mathbf{x}_{1,i}, \mathbf{x}_{2,i}), \mathbf{e}_i)$ was used for π
    return `Reject`
Let $\{\mathsf{rsp}_i\}$ denote the sequence of outputs of the verification step in Figure 4.2
forall $\mathsf{rsp} \in \{\mathsf{rsp}_i\}$
  if $\mathsf{rsp} = \mathtt{Reject}$
    return `Reject`
return `Accept`

Algorithme 15 : $\mathsf{Open}((\pi, \mu), \mathcal{TK})$

Input : Couple message/signature, and the list $\mathcal{TK}$ of tracing keys
Output : Index j of the issuer, or $\perp$

$\textsc{Ch} = \mathcal{H}(\textsc{Cmt} \mid \mu)$;
for i **from** 0 **to** $t_\lambda - 1$ **do**
  switch $\textsc{Ch}_i$ **do**
    case 0
      return $j \in \mathcal{S}$ *such that* $\mathbf{a}_2 = COM(\{\phi_i(\mathtt{tk}_i^{(j)} + \mathbf{u}_i)\} \mid \mu)$, *for* $\mathtt{tk}^{(j)} = (\mathbf{x}_{1,j}, \mathbf{x}_{2,j} - \mathbf{e}_j) \in \mathcal{TK}$
    case 1
      return $j \in \mathcal{S}$ *such that* $\mathbf{a}_1 = COM(\{\phi_i((\mathbf{x}_{1,i}, \mathbf{x}_{2,i} - \mathbf{e}_i) + \mathbf{u}_i - \mathtt{tk}_i^{(j)})\} \mid \mu)$, *for* $\mathtt{tk}^{(j)} = (\mathbf{x}_{1,j}, \mathbf{x}_{2,j} - \mathbf{e}_j) \in \mathcal{TK}$
    case 2
      continue; `/* No tracing capabilities */`
return $\perp$ `/* Fail to trace (` $\textsc{Ch} = (2 \ldots 2)$ `) */`

Algorithme 16 : $\mathsf{Claim}((\pi, \mu), \mathtt{tk}_i), \mathsf{Trace}((\pi, \mu), \mathtt{tk}_i)$

Input : Couple message/signature, index i, and tracing key $\mathbf{Z} \in \mathtt{tk}[i]$.
Output : `True` if the signature belongs to user i, `False` otherwise.

if $i = \mathsf{Open}((\pi, \mu), \{\mathtt{tk}_i\})$
  return `True`
else
  return `False`

4.4.3 Security Analysis

We will rely on various assumptions, k-split-SIS to handle the dynamic joins in *Non-Frameability*, ISIS for *Unforgeability*, and the Testable weak Zero-Knowledge property of proofs for *Anonymity*. Due to space restrictions, the proofs for anonymity and soundness are deferred to App. 4.B and 4.C respectively.

Completeness. Correctness of our scheme is guaranteed by the completeness of the various Stern proofs used through our scheme. Due to the nature of the Trace, Claim and Open algorithms, it is clear that as soon as Claim returns 1 on an input $((\sigma, \mu), \mathbf{y}_i)$, Trace will do the

same, and Open will point to the user i. Thanks to the Random Oracle, we can ensure that no $\textsc{Ch}_{=}\,\mathcal{H}\,(\textsc{Cmt} \mid \mu)$ is the $\mathbf{0}$ string, hence every tracing procedure is deemed to succeed.

Anonymity. We now turn to proving that issued signatures cannot link to a specific user inside the group.

Theorem 4.4.1. *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme with an advantage ϵ in a polynomial time τ, then there exists an adversary $\mathcal{B}$ that can break the Testable weak Zero-Knowledge of the Stern proofs with an advantage $\epsilon' \geq \epsilon/n$ and a time $\tau \approx \tau'$ where n is the number of join query.*

Soundness. Within the soundness analysis, we prove traceability and non-frameability. As in the correctness analysis, we merge the Open, Claim and Trace algorithms, and we make sure they do not fail, by controlling the Random Oracle so that he does not output a $\mathbf{0}$ string.

Theorem 4.4.2. *Assume $k = O(n^{1/4})$, if there exists an adversary $\mathcal{A}$ against the soundness of the scheme, then we can build an adversary $\mathcal{B}$ that can either break a computational problem (k-split-SIS, ISIS) or the security of the underlying proof (Zero-Knowledge or Non-Malleability) where Q is maximal number of users (sum of the number of active and passive join queries).*

4.4.4 Choosing the parameters

In this Section, we briefly explain how to set the parameters to obtain a working instance of our new traceable signature scheme. The first step is to obtain parameters for the mNTRUSign signature scheme, those are given in [ABDG14, Tab. 3], then ensure that the output distribution of mNTRUSign.Sign algorithm satisfies the rejection sampling test (*i.e.* the tracing keys have the expected distribution). This step is fundamental so users can correctly use the Decompose routine from [LNSW13].

For the security reduction to work, it is important to restrict the number of users to $\mathcal{O}(N^{\frac{1}{4}})$. As mentioned earlier, this limitation mainly comes from the $k - \mathsf{SIS}$ problem from [BF11]. It might be possible to tackle this issue using recent improvements by Ling *et al.* [LPSS14], this is left as a perspective.

Finally, notice that it is possible to increase the number of group user by taking larger parameters than those recommended in [ABDG14], but still following the guidelines there.

4.5 Conclusion and Perspectives

In this paper, we introduced the first post-quantum traceable signature scheme that supports dynamic enrolment of new users, assures non-frameability to its users, and allow the group manager to hire sub-openers to trace individual misbehaviours. The scheme satisfies both the traceability and anonymity notions of the BMW security model, and is secure under worst-case hardness of lattice problems assumptions.

Our scheme satisfies the traceability property according to a technique (implicit in [ABCG15]) of Alamélou *et al.*, that we formally introduced as *Testable weak Zero-Knowledge*. We showed this notion – and its associated security model – is perfectly suited for traceable signatures, and actually corresponds to the deterministic version of SternExt protocol.

Among other interesting features, we were unfortunately unable to figure out a way of providing *backward-unlinkability*. Indeed, as soon as a user gets revoked, anyone can trace

# of users	4	10	20	50
N	431	10,007	160,001	6,250,009
d	34	834	13,334	520,835
$\log_2 q$	16	18	22	28
η	1.296	0.941	0.985	0.997
$\mathcal{N}$	109	2617	41825	1,633,698
α	6	6.376	6.091	6.018
σ	848	14,774	247,434	9,772,779
mpk size $(N \log_2 q + 2\lambda)$	900 B	22 kB	429 kB	21 MB
msk size	175 B	3.9 kB	61.9 kB	2.4 MB
User key size	1.3 kB	40.2 kB	800kB	38.5 MB
Signature size	156 kB	4.87 MB	97.6 MB	4.7 GB

TABLE 4.2 – Parameters for our traceable signature scheme. All the parameters are given for 100 bits of security.

back the signatures this specific user issued before being revoked. On a different aspect, trace agents hired by the group manager could become malicious and try to frame the user(s) they are responsible for tracing. It would be even more privacy-enhancing to have mechanisms to revoke misbehaving sub-openers, but this is currently an open question.

The fundamental lattice problem behind our protocol is a "one-more SIS" problem. Unfortunately there does not exist, up to our knowledge, any security proof for this problem. The k-SIS problem introduced by Boneh and Freeman can be seen as such a problem but with a linearity constraint which limits k to the dimension of the lattice. In order to have a security proof for our scheme we related our security to this problem which intrinsically limits the number of users in any case to the dimension of the lattice (and more precisely to $O(n^{1/4})$ in our proof. Meanwhile if a more general "one-more SIS" problem could be proven secure our scheme may admit a number of users polynomial in n. Therefore, it is an interesting question to solve.

Finally, it would be nice having such a traceable signature scheme to be secure in the standard model such as the construction of [LY09], but this is still an open question for post-quantum cryptography.

Appendix

4.A Binary Stern is Testable weak Zero Knowledge

In this section, we recall the unrandomized Stern Protocol, and show it satisfies the TwZK properties.

Public data: $(\mathbf{H}, \mathbf{s}, \omega) \in \mathbb{F}_2^{k \times n} \times {\mathbb{F}_2}^{n-k} \times \mathbb{N}^*$, $h, \mathcal{H}$
$\mathcal{P}$ proves the knowledge of $\mathbf{z} \in \mathbb{F}_2^n$ such as $\mathbf{H}\mathbf{z}^\top = \mathbf{s}$ and $wt(\mathbf{z}) = \omega$

$\mathsf{Prove}(\mathbf{z}, \mathcal{L}; \pi, \mathbf{u})$:
- $\pi \xleftarrow{\$} \Sigma_n$
- $\mathbf{u} \xleftarrow{\$} \mathbb{F}_2^n$
- $\mathbf{c}_1 = h(\pi|\mathbf{H}\mathbf{u}^T), \mathbf{c}_2 = h(\pi(\mathbf{u})), \mathbf{c}_3 = h(\pi(\mathbf{z}+\mathbf{u}))$
- $c = \mathcal{H}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$
- 3 cases:
  - $c = 0$: $\mathcal{P}$ sends $\pi, \mathbf{u}, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$
  - $c = 1$: $\mathcal{P}$ sends $\pi, \mathbf{z}+\mathbf{u}, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$
  - $c = 2$: $\mathcal{P}$ sends $\pi(\mathbf{z}), \pi(\mathbf{u}), (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$

$\mathsf{Verif}(*, *, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3), \mathcal{L})$: computes c:
- 3 cases:
  - $c = 0$: checks $\mathbf{c}_1, \mathbf{c}_2$
  - $c = 1$: checks $\mathbf{c}_1, \mathbf{c}_3$
  - $c = 2$: checks $\mathbf{c}_2, \mathbf{c}_3$
  - and $wt(\pi(\mathbf{z})) = \omega$

$\mathsf{TestWit}((*, *, (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)), z')$: computes c:
- 3 cases:
  - $c = 0$: computes $\pi(\mathbf{z}' + \mathbf{u})$ and checks $\mathbf{c}_3$
  - $c = 1$: computes $\mathbf{z}' + (\mathbf{z} + \mathbf{u})$ and checks $\mathbf{c}_2$
  - $c = 2$: can not test

FIGURE 4.A.1 – Stern's protocol, ran through the Fiat Shamir framework

Proof. Correctness and Soundness are proven directly as in the initial Stern Protocol. As for the initial protocol, there are 3 different cheating strategies depending on the value computed for c. We are going to focus on $c = 0$, the others are left to the sagacity of the reader. (It should be noted that for $c = 2$ no test are possible using $\mathsf{TestWit}$, so the indistinguishability proof is even easier.)

We use the Random Oracle programmability on $\mathcal{H}$ to force the Hash Value to be 0. Following Stern's framework, in this case the SProve algorithm picks an honest π and $\mathbf{u}$, and a random small weight vector $\mathbf{z}$ (not satisfying the equation), and proceeds as expected.

The difference in distributions between honestly generated proofs and simulated one comes from $h(\pi(\mathbf{z}+\mathbf{u}))$, which in the first case is correctly generated while in the other one is not. Assuming h is also a random oracle the value $h(\pi(\mathbf{z}+\mathbf{u}))$ is random compared to the rest of the view of the adversary.

Now the only way the adversary could possibly distinguish a value $h(\pi(\mathbf{z}+\mathbf{u}))$ computed for a $\mathbf{z}$ in the language from a random value, would be by querying $\pi(\mathbf{z}+\mathbf{u})$ to the ROM. (Until now, the ROM programmability allows us to argue that the two visions are indistinguishable).

Using Random Oracle Observability, one can then monitor the calls to the random oracles made by the adversary, and for the $\pi, \mathbf{u}$ used in the challenge, parse his calls to define values z_is. In the eventuality the adversary queries the ROM on a $\mathbf{z}_i$ in the language (*i.e.* $\mathbf{H}\mathbf{z}_i^\top = \mathbf{s}$ and $wt(\mathbf{z}_i) = \omega$), a simulator generating a random proof without knowing a word in the

language can use the adversary's query to solve the Syndrome Decoding challenge (or a Closest Vector Problem in lattice terms). □

4.B Proof of Theorem 4.4.1

Intuitively, we will show that one can supersede all the signing queries on the challenge identities by random answers with simulated proofs.

Proof. Let us assume that an adversary is able to break the anonymity property of our scheme. It means that in the anonymity game, he has a non-negligible advantage $\epsilon > 0$ to distinguish $G^{(0)}$ where $b = 0$ from $G^{(1)}$ where $b = 1$. We start our sequence of games from $G^{(0)}$, denoted $\mathcal{G}_0$.

$\mathcal{G}_1$ The simulator $\mathcal{B}$ is allowed access to a weak Zero-Knowledge Oracle (he can ask several proofs for the same witness on different tag, and then receive a challenge π^* for a fresh tag) first runs the normal Initialization process. With everything else, he now knows the secret master key msk.

The future challenge user i_0 will have a private key $\mathsf{usk}[i_0] = (\mathbf{x}_0, \mathtt{tk}_0 = (\mathbf{x}_{1,0}, \mathbf{x}_{2,0} - \mathbf{e}_0))$.

This Setup is indistinguishable from the real one since all the keys are generated as in the real game.

Because of the knowledge of the master secret key, $\mathcal{B}$ easily answers any join queries, both active and passive. However he still has to guess i_0, which is correct with probability $1/n$, where n is the total number of passive join queries. It can also answer any corruption ; that should not happen for the challenge user, even if we could in this game.

As our simulator is able to know all $\mathsf{usk}[i]$ for passive registered users (except the challenge one), he will be able to answer signing queries as users would do. As this game is identical to the real game, $\mathcal{A}$ still has an advantage

$\mathcal{G}_2$ We then modify the game, and for the challenge user the simulator $\mathcal{B}$ now interacts with the weak Zero-Knowledge Oracle for a proof using the tag μ_j, when the challenge query arise, we use the challenge proof π on the tag μ^* as the answer. Under the Testable weak Zero-Knowledge property of the Stern proof, this game is indistinguishable from the previous one.

To complete the proof, we should make the same sequence again, starting from $\mathcal{G}_0$' that is $G^{(1)}$, up to $\mathcal{G}'_5$, that is perfectly indistinguishable from $\mathcal{G}_3$, hence the computational indistinguishability between $\mathcal{G}'_0 = G^{(1)}$ and $\mathcal{G}_0 = G^{(0)}$. □

4.C Proof of Theorem 4.4.2

Note that following [LPSS14], k-SIS can be polynomially reduced to SIS. The proof relies on the fact that the SternExt protocol (or its modified versions - see Section 4.3) works for vectors which are short for the infinite norm. A result by Boneh and Freeman [BF11, Property 4.8] shows that if $k < n^{1/4}$, then the only combinations in the $\mathbb{Q}$-span of k valid $(\mathbf{x}_i, \mathtt{tk}_i)$ vectors which norm is below a small constant times their norm, are the k vectors in themselves or their negation. Hence, since parameters are chosen so that we meet this case, it means that if a forger successes in the SternExt authentication, either he found a short vector which is not in the $\mathbb{Q}$-span of the k valid vectors and hence he breaks the k-SIS problem and hence the SIS problem, or he breaks the security of the TwZK proof of knowledge.

Proof. Non-frameability and misidentification are very closely related, we will treat both simultaneously, there are two ways to cheat the soundness of our scheme: either by creating a new user ($\mathcal{G}_1$) which induces a misidentification attack, or by using an existing valid signature but on a new message ($\mathcal{G}_2$) which breaks the non-frameability.

We will construct two different games, in the first one ($\mathcal{G}_1$), we assume the adversary is able to forge a signature by generating a new user secret key (a new tuple $(\mathbf{x}_i, \mathtt{tk}_i)$), in the second one ($\mathcal{G}_2$) the adversary is either able to use some weakness in either the join proof, or the signature proofs ; or to directly attack the syndrome $\mathbf{Bx}_i$.

$\mathcal{G}_1$ Let us be given a k-split-SIS challenge $(\mathbf{S} = [\mathbf{B} \mid \mathbf{P}])$, $\{(\mathbf{x}_i, \mathtt{tk}_i)\}_{i\in[\![1,k]\!]}$. We build an adversary $\mathcal{B}$ able to solve this challenge, from $\mathcal{A}$ that breaks the soundness of our scheme by generating a new signature with an honest proof. $\mathcal{B}$ behaves honestly, but controls the Random Oracle so that he can observe the queries made by the adversary. To answer the i-th join queries, if this is an active join, $\mathcal{B}$ extracts $\mathbf{x}_i$ from the proof (using the observability) and adapts it to $\tilde{\mathbf{x}}_i$ so that there exists a unused $\tilde{\mathtt{tk}}_i$ such that $(\tilde{\mathbf{x}}_i, \tilde{\mathtt{tk}}_i)$ is a valid secret key ; if it is a passive join, $\mathcal{B}$ directly chooses a fresh $(\mathbf{x}_i, \mathtt{tk}_i)$.[3] $\mathcal{B}$ is able to give the tracing key $\mathtt{tk}_i$ to the adversary in both cases.

After at most q join queries, $\mathcal{A}$ is able to output a new signature with a new honestly generated proof with non-negligible probability. As $\mathcal{B}$ can use the extractability of the proof scheme, he can obtain $(\mathbf{x}_i^*, \mathtt{tk}_i^*)$ and so he is able to answer the challenge k-SIS instance.

$\mathcal{G}_2$ In this game, we assume $\mathcal{A}$ breaks the non-frameability by reusing the same $\mathbf{x}_i$, this happens either with $\mathcal{A}$ extracting $\mathbf{x}_i$ from the initial syndrome during the join, or by mauling signatures.

$\mathcal{B}$ now has access to a simulation soundness oracle, where he asks proofs for q tags, and then has to generate one for a fresh one.

He starts by initializing the protocol normally, and gives msk to $\mathcal{A}$. He then behaves normally, and joins as the challenge user by picking a small $\mathbf{x}$ and running the normal join procedures (as for any other passive join queries).

$\mathcal{A}$ can corrupt any user, if he tries to corrupt the challenge user, the simulation fails. As all uncorrupted users look the same, with non-negligible probably the simulation continues. As $\mathcal{B}$ knows $\mathbf{x}$, he can answer any signing query.

After at most Q signing queries, $\mathcal{A}$ succeeds in breaking the non-frameability with non-negligible probability by generating a signature on an uncorrupted user. As uncorrupted users are indistinguishable, with non negligible probability this user is the challenge one. In the following, we are going to detail a step of changes such that, $\mathcal{B}$ can exploit this attack.

— $\mathcal{B}$ now sends a syndrome $\mathbf{t}$ during the join, together with a simulated proof of knowledge of the associated $\mathbf{x}$ (while still knowing the true value $\mathbf{x}$). He then proceed honestly. Under the Zero-Knowledge property of a simulated (randomized) Stern Proof the advantage of the adversary remains the same.

— $\mathcal{B}$ now forgets $\mathbf{x}$, and starts answering signing queries via Testable weak Zero-Knowledge Oracle for the word $\mathbf{s}$ using the message to sign as a tag. Under the Non-Malleability of the non randomized Stern proof, the adversary advantage remains the same.

3. This is possible with overwhelming probability using a trick similar to the one in [Lyu12]

— At this step $\mathcal{A}$ is able to produce an honestly computed proof of knowledge of $(\mathbf{x}_i, \mathtt{tk}_i)$ such that $[\mathbf{B} \mid \mathbf{P}] \begin{bmatrix} \mathbf{x}_i \\ \mathtt{tk}_i \end{bmatrix} = \mathbf{s}$ with a non-trivial small weight vector, during the join procedure $\mathcal{B}$ sent a random $\mathbf{t}$ and received $\mathtt{tk}'$ such that $\mathbf{P}\mathtt{tk}' = \mathbf{s} - \mathbf{t}$. As this is a non-frameability attack, we have $\mathtt{tk}' = \mathtt{tk}$, so $[\mathbf{B}|\mathbf{P}] \begin{bmatrix} (\mathbf{x} - \mathbf{x}_i) \\ (\mathtt{tk} - \mathtt{tk}') \end{bmatrix} = \mathbf{t}$. $\mathcal{B}$ now possesses a small weight (non-trivial) solution to an ISIS problem on $(\mathbf{B} \mid \mathbf{P}), \mathbf{t}$.

Now let $\mathcal{A}$ be an adversary against the soundness of our scheme with an advantage ϵ. If with probability greater than $\epsilon/2$, $\mathcal{A}$ breaks the misidentification property of the scheme, then we can run the game $\mathcal{G}_1$ and break a k-split-SIS, else if with probability greater than $\epsilon/2$, $\mathcal{A}$ breaks the non-frameability property the sequence of game $\mathcal{G}_2$ to either break the Zero-Knowledge property of Stern Proof, the Non-Malleability of non-randomized Stern proofs, or a SIS problem.

So if there exists an adversary against the soundness of our scheme, we can break with non-negligible probability one of the previous problems. □

Chapitre 5

Délégation de Signature dans le Cloud

Sommaire

Contexte

Le chiffrement complètement homomorphe (Fully Homomorphic Encryption (FHE) en anglais) a connu de profonds changements au cours des dix dernières années. Durant cette période, les coûts calculatoires ont chuté, des librairies ont été développées, et de nouvelles applications deviennent possibles. Des prototypes permettant de réaliser des diagnostics médicaux, de faire du traitement du signal, des statistiques sur le génome ou encore des requêtes sur des bases de données, le tout de manière privée laissent entrevoir le déploiement pratique du FHE dans un futur proche. Cependant, dans la plupart de ces applications, augmenter la vie privée, la sécurité ou encore débloquer de nouvelles fonctionnalités vient avec surcoût notoire en terme de calculs ou de communications.

Dans ce chapitre, nous partons de ce dernier paradigme et montrons qu'il est possible d'utiliser le FHE pour améliorer les performances en pratique ainsi que la flexibilité des dispositifs optimisés matériellement pour réaliser des signatures ECDSA, tout en maintenant leur sécurité et leur fonctionnalité. Plus précisément, nous montrons comment diminuer la charge de calcul d'un Hardware Security Module (HSM) en délégant une partie de ces calculs à un tiers non sûr disposant d'une grande puissance calculatoire, en utilisant du FHE. Déléguer ces calculs constitue une solution bon marché et à la demande au problème de flexibilité de ce type de systèmes, où les performances résultantes augmentent proportionnellement aux ressources ajoutées.

Enfin, nous démontrons la faisabilité de nos protocoles par une preuve de concept (implémentation) utilisant la librairie HElib (adaptée pour gérer des modules pour l'espace des clairs en multiprécision).

Ce travail est issu d'une collaboration avec Carlos Aguilar Melchor, Philippe Gaborit, Tancrède Lepoint, et Thomas Ricosset.

5.1 Introduction

HSMs play an essential role in public key infrastructures. They act as security safeguards by securely generating, storing and manipulating cryptographic secrets on trusted hardware. An HSM has a number of different uses, ranging from key generation/storage to secure hybrid encryption. In this chapter, we focus on its use as accelerator for SSL connections (offloading of the computationally intensive cryptographic calculations). In particular, an HSM will be used as a fast decryption/signing tool that can be used by servers to handle hundreds or thousands of TLS handshakes per second. In this setting, using an HSM ensures that nobody (rogue server administrator, hacker having gained administrative privileges, person stealing the HSM itself) can easily obtain the decryption/signing keys.

Even though HSMs are themselves expensive, the biggest cost of an HSM is in the procedures it entails: procedures for installing, configuring, testing, auditing, operating, restoring and retiring. [1] In particular, if a user has increasing throughput needs, the whole process must be restarted for each HSM, and thus requires growing IT budgets. In other words, using HSMs yields costly systems with limited *scalability* (*i.e.* the ability to be enlarged to accommodate a growing amount of work), and no *flexibility* (*i.e.* the ability to respond to a temporary work growth in a timely and cost-effective manner).

1. It takes almost a year and a half to get FIPS 140-2 certified, with base fees ranging from 6,250 USD to 16,000 USD depending on the targeted security level according to the NIST — cf. Tab. 4-2 of http://csrc.nist.gov/groups/STM/cmvp/documents/CMVPMM.pdf.

Motivation. Using external resources to store secret data is usual when using HSMs. Now, FIPS 140-2 guidelines allow exportation of a secret key when the key is encrypted and the system prevents unauthorized disclosure, *i.e.* complies with the roles and constraints defined in FIPS 140-2 (in particular the keys used to output the encrypted secret key must provide enough security). Usually, key export is used to move keys between two equally FIPS 140-2 compliant HSMs.

Recent advances in *homomorphic cryptography* completely reshaped the possibilities to secure a system on untrusted environments. Regarded as cryptography's "Holy Grail", fully homomorphic encryption (FHE) enables computation of arbitrary functions on encrypted data [Gen09b, BGV12]. In this work, we study how homomorphic encryption can be used to outsource signature generations to untrusted sources of computational power, so as to obtain on-demand, scalable and flexible systems which maintain the security protection brought by hardware modules.

5.1.1 This Work

5.1.1.1 Our Contributions

In this work, we present a protocol outsourcing the generation of ECDSA signatures to some untrusted source of computational power by an HSM. [2] Note that it is also possible to outsource RSA signatures with a different protocol, but even with optimizations such as RNS representation *and* Montgomery Reductions (such as [BI04]), our tests demonstrate that the computational cost is too high to consider practical benefits in a foreseeable future. Given this fact, and space constraints, we only provide an ECDSA protocol.

The practicality of our technique is validated through proof-of-concept implementations of (i) the outsourcing of part of the ECDSA signature process, and (ii) the operations performed by the HSM. Our implementations are based on an adaptation of HElib [HS14, HS15] so as to handle large plaintext spaces (HElib is restricted to single precision prime powers).

The implementation (i) has been deployed on an Amazon EC2 instance (`c4.8xlarge`) to allow reproducibility and verification of our results. Our implementations will be made fully available under GNU General Public License (GPL). Performance results show that using fully-homomorphic encryption allows to obtain a flexible and on-demand system with increased performances compared to a classical HSM. It is worth noting that our practical setting using homomorphic cryptography *speeds the system up*, rather than slowing it down.

Exporting some information about the signing keys or intermediate random values encrypted with an HE scheme allows to use external resources not only for storage but also for computational purposes. Besides increasing the signature throughput and lowering the costs, outsourcing will bring additional benefits such as flexibility and product longevity. Flexibility will come from the small computational and communication costs of the outsourcing protocols: a unique HSM for a manufacturer could adapt the amount of external resources (sold or rented by the manufacturer) used depending on the needs of each client. The impact on product longevity should also be considered. Traditional public-key cryptography does not scale well with increasing security parameters. A linear increase in the security requirements of ECDSA results in a cubic overhead in computational costs. In practice this means that, as time goes by and security has to be increased, HSMs must be replaced. Outsourcing signatures guarantees that the costly ECDSA operations are done outside of the HSM and

2. The external resource can be for example a PCI card, a blade server with dedicated hardware, or a cloud service. The main motivation is to overcome the usual limitations of HSMs (decryption/signature throughput, high cost).

thus we only need to replace the resources used by the HSM and not the HSM itself. [3] Of course, to increase security, the parameters of the lattice-based FHE schemes used by the HSM must be adapted, but lattice-based cryptography scales up very well with security (quasi-linearly [BV14]), which will likely improve the HSMs longevity.

5.1.1.2 Motivate Standardization

One of the objectives of this chapter is to promote the eventual emergence of a standard adopting common cryptographic techniques for homomorphic encryption. Indeed, the practicality of our techniques was validated using a generic and non-optimized HE library HElib [HS14, HS15]. Thus our solution could have an important impact on the HSM (and smartcard) market as soon as lattice-based cryptography meets FIPS 140-2 and CC standards, and optimized implementations thereof become available.

Current efforts in standardization include the IEEE P1363 and X9.98 standardizations of the NTRU cryptosystem [HPS98] and the upcoming standard ISO/IEC 18033-6 on partially homomorphic encryption.

5.1.2 Related Work

Homomorphic Signatures and Proxy Signatures. Note that signature outsourcing using homomorphic encryption should not be confused with homomorphic signatures [JMSW02] or proxy signatures [MUO96]. Homomorphic signatures allow to produce a short signature on the output of some computation on signed data. Proxy signature schemes permit an entity to delegate its signing rights to another. Many works have been proposed, with many different flavors, but seldom based on standard signatures. In particular, to our knowledge, all proxy signatures output signatures having a different form than classical signatures. Also, proxy signatures schemes allow the proxy to sign as many messages as she wants.

In this work, we use a fully homomorphic *encryption* scheme to homomorphically compute standard, classical, ECDSA signatures using external resources. In particular, the final signatures are signatures that have exactly the same structure as if they have been computed directly. Only the most expensive part of the computation is securely delegated, while the computationally light operations are still being performed by the original entity.

Precomputations and Outsourced Computations. Many works in the early 90's proposed to speed up protocols either based on factoring [Sch91, BGMW93, LL94, de 95, BPV98] or discrete logs [BPV98] by using precomputations. While we also use precomputations in our protocol (see Fig. 5.1 for an overview and 5.3 for more details), most of speed-up is achieved through the outsourcing of some computations to an external, untrusted source of computational power, using additional techniques such as windowing [Knu97].

Additionally, several protocols to outsource computation have been proposed in the literature, especially regarding to the delegation of exponentiations [Fei86, MKI90, NS01, HL05, VDCG+06]. In our context, the work outsourced to the untrusted source of computational power will essentially be a scalar multiplication on an elliptic curve.

Multi-Party Computation. Multiparty computation can be used to emulate HSMs — as proposed by Dyadic Security (`https://www.dyadicsec.com`), by splitting the secret keys

3. Even better, if an HSM uses a rented resource (such as the Cloud), the computational power of the resource is likely to increase on its own.

and credentials between different machines which are strongly segregated. In our context, this would either require the HSM to perform the full computation (that is, what we wanted to avoid) or to split the computation between different untrusted sources of computational power. Besides a greater communication and overall computational complexity, it also follows that in case the outputs of the HSMs are monitored by an adversary, she can recover the secret key.

Verifiable Computation on Encrypted Data. At ACM CCS 2014, Fiore, Gennaro and Pastro [FGP14] proposed a notion of verifiable computation over encrypted data.

Prototypes using Homomorphic Encryption. Open-source libraries for homomorphic encryption are available online [LN14, HS14, HS15, ABG+16], and were used as building blocks in several prototypes in the recent years, such as statistics computations [NLV11], machine learning [GLN13], signal processing [AFF+13], database queries [BGH+13, CKK15], private health diagnosis [BLN14], genome statistics [LLN15], and edit distance [CKL15]. To the best of our knowledge, no prototype tackled the delegation of standard signatures.

5.1.3 Outline

The chapter is organized as follows. Sec. 5.2 introduces the notation used throughout the chapter, Sec. 5.3 introduces the general approach to outsource signatures in presence of an untrusted external source of computational power, and Sec. 5.4 focuses on the delegation of ECDSA signatures. A security analysis is provided in Sec. 5.5. Experimental results for our protocol are presented in Sec. 5.6, before concluding the chapter in the last section.

5.2 Preliminaries

Most of the notations used throughout this chapter were already introduced in chapter 2. Let $\mathbb{Z}_q$ be the set of integers modulo q, $\mathbb{F}_q$ be the finite field of q elements, and denote $E(\mathbb{F}_q)$ the group of points $(x, y) \in \mathbb{F}_q^2$ belonging to an elliptic curve E. Sampling uniformly an element x from a set S is denoted $x \xleftarrow{\$} S$. If q is an integer, denote $\ell_{w,q}$ the length of its representation using a basis of w elements (*e.g.* $\ell_{2,q}$ is its bit length). Finally $k_{\mathrm{ECDSA},\lambda}$ represents the key bit length for ECDSA for a security parameter λ. Following NIST recommendations [NIS15], we have $k_{\mathrm{ECDSA},128} = 256$.

5.2.1 ECDSA

In this section, we briefly recall the ECDSA signature scheme. Introduced by Vanstone in 1992 (see [HMV03]), ECDSA is the elliptic-curve analogue of DSA and features short keys and signature sizes. Its signature and verification procedures are provided in Alg. 17 (we do not recall the key generation procedure, which is of no relevance to this work). The domain parameters provided by the NIST include three types of curves: over prime fields (P-xxx), over binary fields (B-xxx) or Koblitz curves (K-xxx). Note that, when compared to prime curves, curves over binary fields and Koblitz curves would reduce considerably the outsourcing costs given their small characteristic. However, such curves are not usually proposed in the HSM setting (*e.g.* in the HSMs we compare to). For this reason, in this work, we only consider elliptic curves over prime fields. For such curves, q denotes the field size,

(a, b) are two field elements defining the short Weierstraß equation

$$y^2 = x^3 + ax - b, \text{ with } 4a^3 + 27b^2 \neq 0 \mod q,$$

and G is a base point of prime order n on the curve. As usual for such curves, we consider that $|E(\mathbb{F}_q)| = n$ (*i.e.* a cofactor $h = 1$).

Algorithme 17 : ECDSA

ECDSA.Sign
Input : m, $pk = (q, E, G, Q = sG, n)$ and $sk = s$, m being a message, q an integer, E an elliptic curve over $\mathbb{F}_q$, G, Q two points of E, s an element of $\mathbb{F}_q$, and n the order of G on E.
Output : ECDSA signature $(r, t) \in \mathbb{F}_q^2$ of m
begin
1 **repeat**
2 Sample $k \xleftarrow{\$} \{1, \ldots, n-1\}$ and compute $(x, y) = kG$;
3 Define $r = x \bmod n$ and $t = k^{-1}(h(m) + s \cdot x) \bmod n$;
4 **until** $r \neq 0$ **and** $t \neq 0$;
5 **return** (r, t);

ECDSA.Verif
Input : q, E, G, Q, (r, t), m, as defined in **ECDSA.Sign**
Output : **true** if and only if (r, t) is a valid signature on m
begin
1 **if** $Q = \infty$ **or** $Q \notin E(\mathbb{F}_q)$ **then return false** ;
2 $(x, y) = (h(m)t^{-1} \bmod n)G + (rt^{-1} \bmod n)Q$;
3 **if** $r == x \bmod n$ **then return true**;
4 **else return false** ;

5.2.2 Homomorphic Encryption and HElib

An homomorphic encryption (HE) scheme [Gen09b] allows to publicly process encrypted data without knowing the secret key. In this section we recall the Brakerski-Gentry-Vaikuntanathan (BGV) homomorphic encryption scheme [BGV12] — implemented in the software library HElib [HS14, HS15] — that we use in our prototypes. This description is mostly taken from [GHS12].

5.2.2.1 BGV

BGV is defined over polynomial rings of the form $R = \mathbb{Z}[x]/(\Phi_m(x))$ where m is a parameter and Φ_m the m-th cyclotomic polynomial. The plaintext space is usually the ring $R_p = R/pR$ for an integer p.

A plaintext polynomial $a(x) \in R_p$ is encrypted as a vector over $R_q = R/qR$, where q is an odd public modulus. More specifically, BGV contains a chain of moduli of decreasing size $q_0 > q_1 > \cdots > q_L$ and freshly encrypted ciphertexts are defined modulo q_0. During homomorphic evaluation, we keep switching to smaller moduli after each multiplication until we get ciphertexts modulo q_L, which cannot be multiplied anymore — L is therefore an upper bound on the multiplicative depth of the circuit we can compute.

5.2.2.2 Homomorphic Operations

The plaintext space of BGV are elements of R_p, and homomorphic additions (resp. multiplications) correspond to additions (resp. multiplications) over the ring R_p.[4]

5.2.2.3 Batching

Rather than encrypting elements of $R_p = \mathbb{Z}_p[x]/(\Phi_m(x))$, the BGV cryptosystem can be slightly modified to encrypt vectors of elements of $\mathbb{Z}_p$ [SV10, SV14, CCK+13] in an SIMD fashion: homomorphic operations implicitly perform the componentwise operations over the plaintext vectors (and rotations are easy to perform via the Frobenius endomorphism). Such a feature is called *batching* and essentially allows, for the cost of an homomorphic evaluation, to evaluate the function independently on several inputs [BGV12]. Let us recall briefly the batching technique for BGV from [SV10, SV14]. If the cyclotomic polynomial $\Phi_m(x)$ factors modulo the plaintext space p into a product of irreducible factors $\Phi_m(x) = \prod_{j=0}^{\ell-1} F_j(x) \pmod p$, then a plaintext polynomial $a(x) \in R_p$ can be viewed as encoding ℓ different small polynomials, $a_j = a \bmod F_j$, and each constant coefficient of the a_j can be set to an element of $\mathbb{Z}_p$. Unfortunately, not every tuple (p, m) yield an efficient batching — we will discuss how we selected (p, m) in Sec. 5.6.1.3.

5.2.2.4 HElib

HElib [HS14, HS15] is, as of today, a standard library for HE prototypes. HElib is a C++ library that implements the BGV scheme (with batching) for arbitrary plaintext space modulus p^r, with p and r in simple-precision. This software library uses NTL [Sho15] and includes numerous optimizations described in [GHS12, HS14, HS15]. HElib supports multi-threading and is distributed under the terms of the GNU GPL version 2.

5.3 Outsourcing Signatures to an External Source of Computational Power: General Context

Assume an HSM is connected to an external source of computational power (*e.g.* the Cloud, a LAN server or a Cryptographic Coprocessor in a PCI-e card) that we denote UC (for Untrusted Cloud). Fig. 5.1 presents the generic interaction between an HSM and UC. Such an interaction consists of two phases:

— An (initial) **offline phase**, during which the HSM precomputes some data based on its secret key and on the parameters of the signature algorithm. Part of this data may be encrypted with an HE scheme, and both the encrypted part and the cleartext part are sent to the UC. Upon reception, the UC may do some extra precomputation.

— An **online phase**, during which the HSM does some precomputation given the input documents and sends data to the UC ; upon reception, the UC does some (homomorphic) computation and sends a reply encrypted under the HE scheme. Finally, the HSM decrypts the reply, does some post-processing to obtain potential signatures to output.

We will describe our protocols and give performances under the latter generic model for sake of comparison and clarity.

4. One can easily obtain a scheme with plaintext space $\mathbb{Z}_p$ by embedding $\mathbb{Z}_p$ into R_p via $a \in \mathbb{Z}_p \mapsto a \in R_p$. Hence homomorphic operations correspond to arithmetic operations over the ring $\mathbb{Z}_p$.

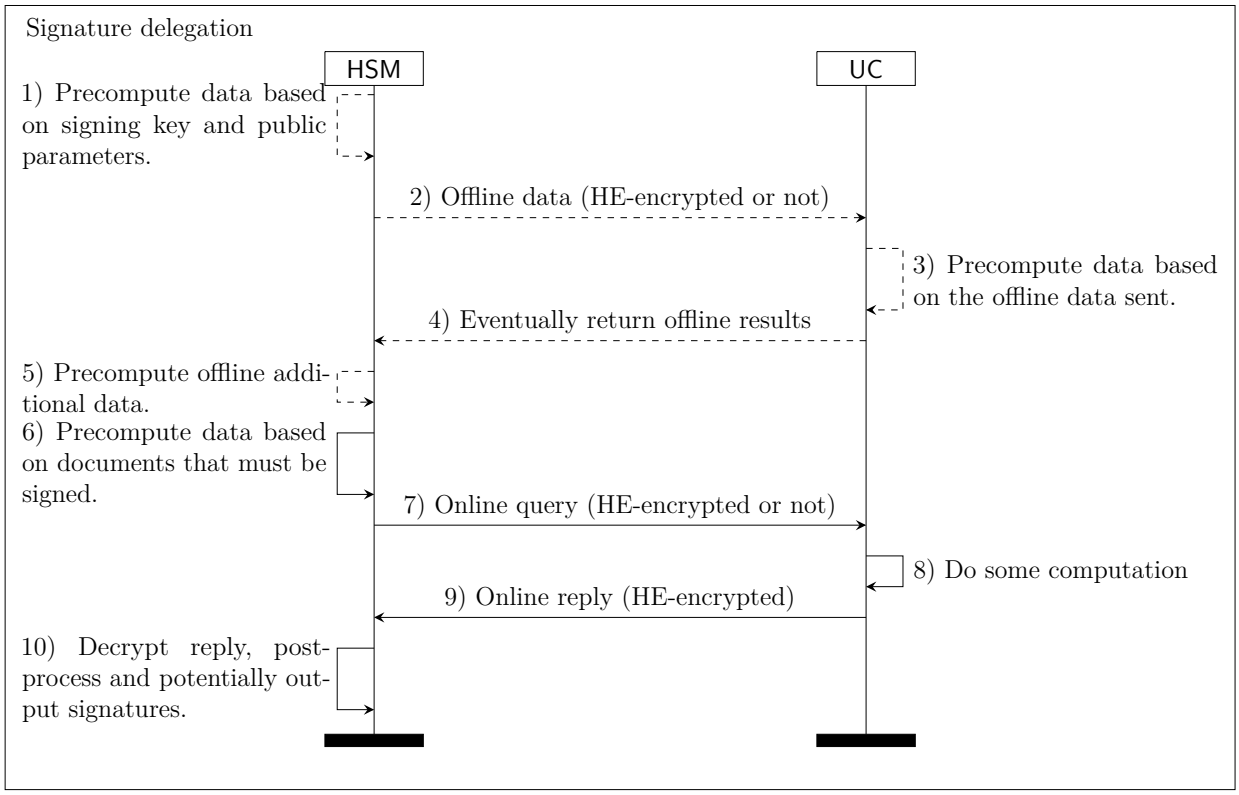


FIGURE 5.1 – Generic description of the HSM outsourcing. Dashed (resp. filled) lines represent offline (resp. online) operations.

5.3.1 Quantitative Goal

Tab. 5.1 presents the processing throughput of various high-end HSMs, and a cost-effective HSM (Ultimaco Cryptoserver se10). The performance values give us a quantitative goal for our prototype: we will estimate the cost of outsourcing batches of about 1000 signatures. If this cost is reasonable in CPU power, it will demonstrate that our protocols are suitable to obtain on-demand and flexible systems competitive with current systems using high-end HSMs.

The careful reader will note that most applications requiring a high throughput of signatures, also require low delay (*e.g.* on SSL/TLS transactions). Even if good results are obtained in the throughput at which signatures are outsourced this does not mean (specially if signatures are batched) that the delay will be low enough for most applications. It is important to note that this does not apply to ECDSA as the outsourced computation (random elliptic curve points) do not depend on the input data and thus can be obtained (in the "offline phase") at a high throughput by the HSM and stored on a buffer to be used immediately (in the "online phase") when a signature must be issued.

5.3.2 Throughput Limitation

Potentially, the computational power of the UC could be as large as needed. In such a setting the bottleneck can be two-fold: (1) the HSM communication bandwidth and (2) the HE scheme encryption/decryption throughput and the HSM post-processing. In practice, both potential bottlenecks have a similar impact, and give an upper bound on the amount of encrypted data that can enter/exit the HSM.

HSM Model	IBM CEX5S	AEP Keyper+	Thalès 6000+	Ultimaco se10
Approx. Price (USD)	$\gg$ 40k ?	40k	40k	4k
FIPS140-2 Cert.	Level 4	Level 4	Level 3	Level 3
RSA 2048	3800	2000	3000	16
ECDSA P256	8100	950	2400	120

TABLE 5.1 – Number of signatures issued per second for several HSMs, according to the constructor specifications, where λ denotes the security level. The results are close to the ones obtained through experimentation in [IN10, p. 14] (RSA only) in 2010. Prices are indicative and were collected from various web sources. We were not able to obtain pricing information for the ultra high-end model CEX5S, besides that it is sold as an option for the IBM z13 mainframe.

If the UC is in a LAN, or even better in a PCI-e card, available bandwidths can be very high. In the PCI-e setting, bandwidth can theoretically reach 252.064 Gbit/s (PCI-e x32 v4.0 with 16-lane slot). However, the HSM will not be able to do homomorphic encryptions or decryptions at such speeds. In practice a throughput limitation around some Gigabits will always exist, and will be the main performance bottleneck.

5.4 Outsourcing ECDSA Signatures

Recall that an ECDSA signature is a tuple $(r, t) \in \mathbb{Z}_n^2$ where $t = k^{-1}(h(m) + s \cdot r) \bmod n$ for a nonce $k \in \mathbb{Z}_n$ such that r is the x-coordinate of the point kG. The most consuming step in the signature generation is the computation of kG (Step 2 in Alg. 17): this is therefore what we want to delegate to the UC. However, the knowledge of the nonce k used in a signature (r, t) allows to compute the secret key as $s = (kt - h(m))/r \bmod n$: keeping k secret is thus of paramount importance. The high-level idea of the protocol is that we will send k (actually a representation thereof) encrypted under an HE scheme $\mathcal{E}$ so that the Cloud can homomorphically compute kG without knowing k. Then the HSM will be able to decrypt, obtain kG, and carry on the signature generation. In Sec. 5.4.1, we give a high-level overview of our protocol ; in the rest of the section, we detail each step of the protocol, and in Sec. 5.4.5 we give the full protocol for completeness.

5.4.1 High-Level Description

Offline Phase. The nonce k used in an ECDSA signature is completely independent of the message to be signed. Thus the computation of kG can be done in an offline phase (the online phase will thus consist of Steps 3-5 of Alg. 17). The high-level description of this phase is provided in Fig. 5.1.

In a first step, some precomputation will be performed on the nonce k (see Sec. 5.4.2 — essentially it will compute a windowed version of k), then in Step 2), the ECDSA parameters and an encryption of k under the HE scheme $\mathcal{E}$ will be sent to the UC. The UC will homomorphically compute kG in Step 3), and will send back the computation to the HSM (Step 4)). Finally, the HSM will decrypt and verify the result.

In the rest of this section, we will specify in detail every step of this protocol, and namely: (i) what precomputations need to be performed on k (Sec. 5.4.2), (ii) which elliptic curve representation and point addition should be used (Sec. 5.4.4).

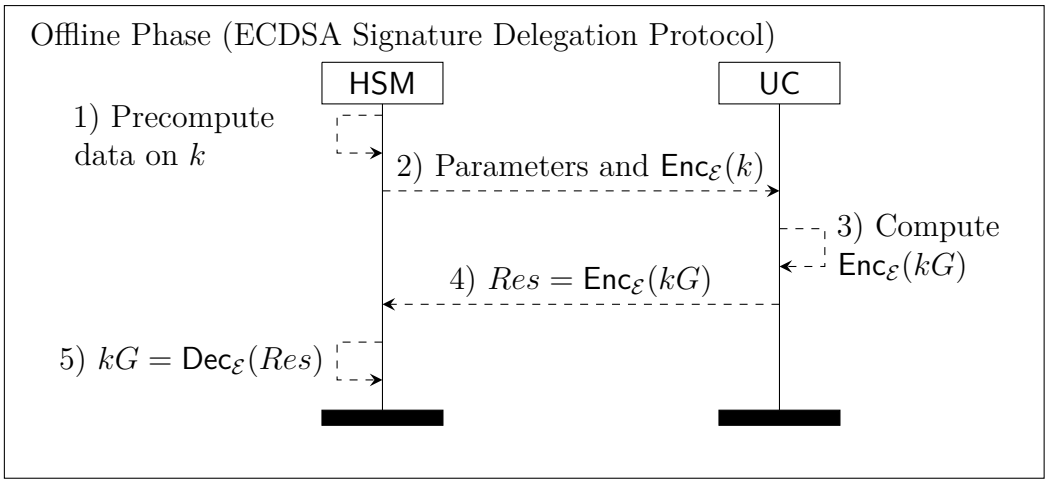


FIGURE 5.1 – High-level description of the offline phase of the ECDSA protocol: delegation of nonce computation. Dashed (resp. filled) lines represent offline (resp. online) operations.

Online Phase. During the offline phase, the HSM will receive and store many kG's to be used as one-time[5] nonces in the online phase. The high-level description of this phase is provided in Fig. 5.2. The UC is no longer needed in that phase, therefore Steps 6) to 9) from the generic interaction described in Section 5.3 are null. In Step 10), the HSM carries on the signature of the messages (Steps 3-4 of Alg. 17).

5.4.2 Step 1): Initialization and Windowing

Each doubling or point addition to compute kG from k will increase the multiplicative depth of the UC computation. Therefore, we use a classical window technique to trade computation complexity for memory.[6] In Sec. 5.4.4, we will detail how to represent the elliptic curve points to obtain an addition formula of multiplicative depth 2.

More precisely during Step 1), for a ω bits window size, the HSM precomputes all $2^{\omega} \cdot \ell_{2,n}/\omega$ points of the form $P_{i,j} = (i \cdot 2^{\omega(j-1)})G$ for i from 0 to $2^{\omega} - 1$, and for j from 1 to $\ell_{2,n}/\omega$ (we discuss the memory implications of this windowing technique in Sec. 5.6.2). Thus, for every scalar $k = \sum_{j=1}^{\ell_{2,n}/\omega} k^{(j)} \cdot 2^{\omega j}$, we have that

$$kG = \left(\sum_{j=1}^{\ell_{2,n}/\omega} k^{(j)} 2^{\omega j} \right) G = \sum_{j=1}^{\ell_{2,n}/\omega} P_{k^{(j)},j} \,,$$

and computing the scalar multiplication only costs $\ell_{2,n}/\omega$ point additions.

The whole process is described in Fig. 5.3 for a single outsourced signature. Note that the dashed lines correspond to operations that can be done offline, so the signature itself can be done online quite fast (Step 10)). The main (offline) cost for the HSM is to send the ordered set $(\mathsf{Enc}_{\mathcal{E}}(x_j), \mathsf{Enc}_{\mathcal{E}}(y_j))_j$ for $P_{k_j,j} = (x_j, y_j)$ and $j \in [0..\ell_{2,n}/\omega)$.

5. In ECDSA, reusing a nonce to sign two different messages leaks the secret key.

6. We leave as an interesting open problem a fine-grained analysis of the homomorphic evaluations of the best time-memory trade-offs for regular implementation of scalar multiplication over prime-field elliptic curves [Riv11].

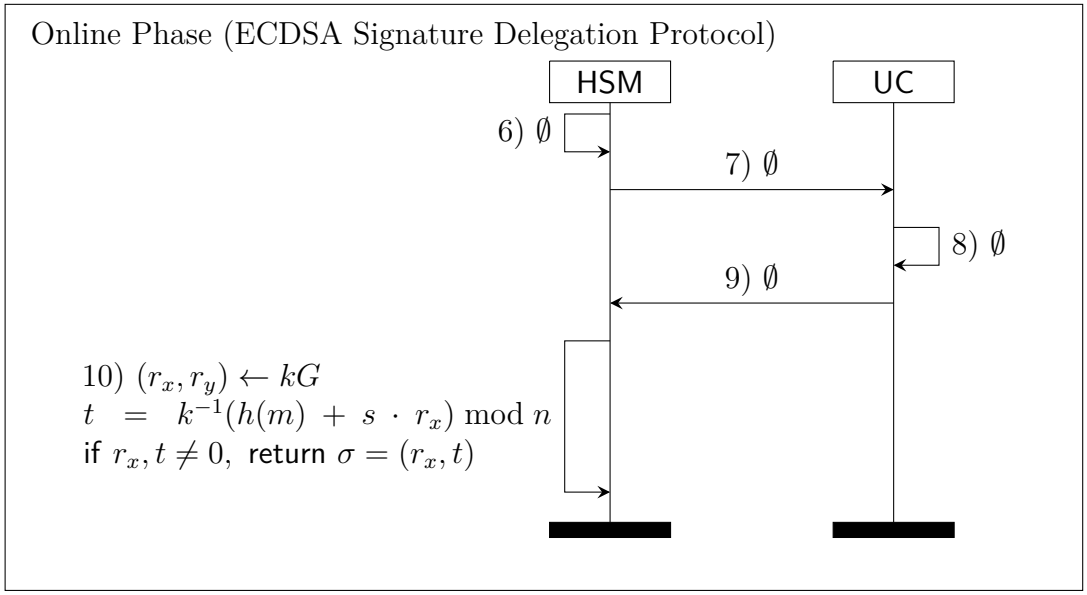


FIGURE 5.2 – High-level description of the online phase of the ECDSA protocol for one signature. Dashed (resp. filled) lines represent offline (resp. online) operations.

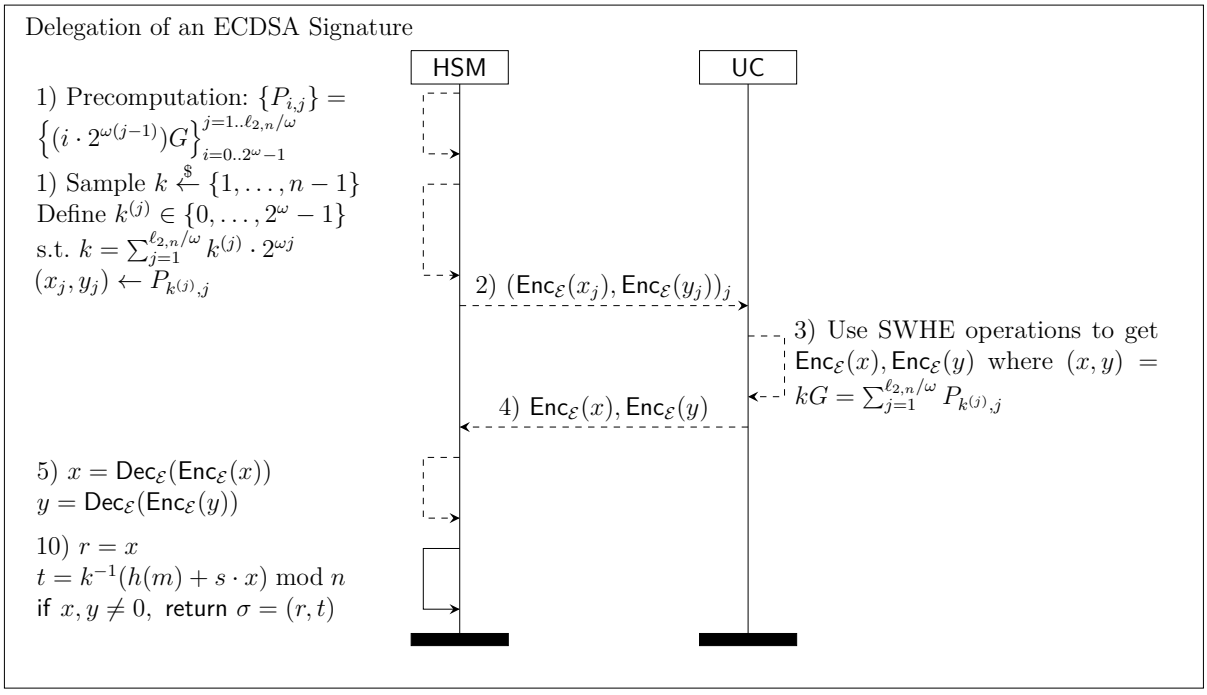


FIGURE 5.3 – ECDSA protocol for one signature. Dashed (resp. filled) lines represent offline (resp. online) operations.

5.4.3 Using HE Batching

Instead of encrypting only one x_j (resp. y_j) per ciphertext, we will use HE batching to encrypt several (say m of them) x_j's in parallel for *different nonces* k's (and the same j). Thus, for the same communication and UC computation complexities, we will be able to precompute several kG's in parallel during the offline phase.

5.4.4 Steps 1) to 3): Revisiting Elliptic Curve Point Additions

In elliptic curve cryptography there are dozens of ways to perform additions and point doubling (see [HMV03] for instance). Most algorithms were designed to minimize the number of multiplications on $\mathbb{F}_q$ one has to perform.[7] In practice the existing algorithms are not a priori optimized in terms of multiplicative circuit depth and we must revisit the existing work to get the best possible algorithm given this new optimization target.

In the rest of the chapter, we focus on elliptic curves with short Weierstraß equation of the form $E(\mathbb{F}_q) : y^2 = x^3 - 3x + b$, for $q \geq 5$ (the ones used in NIST standards). Recently Renes *et al.* proposed an efficient complete addition law that is valid not only for curves of composite order, but also for all prime order NIST curves [RCB15], using standard projective coordinates. In such a representation, points $P(x, y) \in E(\mathbb{F}_q)$ can be written with triple (X, Y, Z) where $x = X/Z$ and $y = Y/Z$ [HMV03]. Such coordinates were designed to speed up the point addition and doubling computation by avoiding field inversions for the benefit of multiplications.

Renes *et al.* presented an addition law [RCB15, Algorithm 4] with maximum multiplicative degree 4, yielding only depth 2 addition circuits ; we briefly review these formulae. Let $P_1(X_1, Y_1, Z_1)$ and $P_2(X_2, Y_2, Z_2)$ be points in the projective embedding of $E(\mathbb{F}_q)$, and denote by $P_3(X_3, Y_3, Z_3)$ their sum. Notice that there are no requirements on P_1 and P_2 being different, nor on being distinct from $\mathcal{O}(0, 1, 0)$.

For sake of clarity, we introduce the following notations:

$$
\begin{aligned}
T_0 &= (X_1Y_2 + X_2Y_1) \\
T_1 &= Y_1Y_2, \\
T_2 &= 3(X_1Z_2 + X_2Z_1 - bZ_1Z_2), \\
T_3 &= Y_1Z_2 + Y_2Z_1, \\
T_4 &= b(X_1Z_2 + X_2Z_1) - X_1X_2 - 3Z_1Z_2, \\
T_5 &= 3(X_1X_2 - Z_1Z_2).
\end{aligned}
$$

Then the complete addition law on the projective embedding is given by the formula:

$$
\begin{aligned}
X_3 &= T_0 \cdot (T_1 + T_2) - 3T_3 \cdot T_4, \\
Y_3 &= (T_1 - T_2)(T_1 + T_2) + 3T_5T_4, \\
Z_3 &= T_3(T_1 - T_2) + T_0T_5.
\end{aligned}
$$

As mentioned in [RCB15], the "plaintext" cost of this formula is $12\mathbf{M} + 2\mathbf{m}_b + 29\mathbf{a}$, where $\mathbf{M}$ (resp. $\mathbf{m}_b$ resp. $\mathbf{a}$) is the cost of a single multiplication (resp. multiplication by b, resp. addition). Therefore, the cost of evaluating this formula in the encrypted domain is 12

7. Also they were design to resist side-channel attacks by using the same "unified" formula for addition and point doubling, but we do not have to worry about such attacks because in homomorphic encryption, the operation flow is independent of the input scalar.

homomorphic multiplications, plus 2 homomorphic sums of b terms, plus 29 homomorphic additions. The main advantage of this formula is that it can be evaluated as a depth 2 circuit, as shown in Fig. 5.A.1.

On a different but non negligible side, this addition law is also optimal in terms of communication complexity. Indeed, as it requires at least three coordinates to avoid field inversions — which are problematic for homomorphic computations — the formula of Renes *et al.* reaches the optimal lower bound. Moreover, the points we start our additions with are on standard coordinates which means that when putting them into projective coordinates, the third coordinate is always 1. We can therefore send just two coordinates per point.

For our proof-of-concept implementation of the protocol using a modified HElib BGV, [RCB15, Algorithm 4] turned out to be the addition law yielding best performances. This can be explained by the relatively low number of multiplications required by this formula, additionally to the optimal depth and representation.

A reasonable question is whether one can reduce significantly the amount of operations by using more coordinates or increasing depth. The short answer proved to be no given the literature on elliptic curve point addition formulas [8]. Roughly, a deeper circuit can sometimes save one coordinate in the point representation, but the FHE parameters become too large. Vice-versa an additional coordinate either imply additional multiplications or communications that are not worth it. Nevertheless, tradeoffs for constrained devices are possible and finding the best tuple (elliptic curve, set of coordinates, addition algorithm, homomorphic encryption scheme) for computing over encrypted elliptic curve points remains an interesting open question.

5.4.5 Full Protocol

For completeness, we provide in Fig. 5.4 the full description of the protocol. The offline stage can be run in advance and different batches of points to be computed can be sent to different servers on the cloud to ensure enough precomputed points are available to do the signatures in the online phase. The delegated work can thus be run completely in parallel on different batches and the results can be kept on the cloud servers to be retrieved by the HSM whenever the buffer of precomputed points gets low.

5.5 Security

In this section, we provide the basic definitions of security of the underlying schemes, and show that the global protocol is secure against a semi-honest adversary. We then discuss the malicious adversary setting.

5.5.1 Basic definitions

The usual definition of security for an homomorphic encryption scheme is indistinguishability against chosen plaintext attacks. The encryption scheme we use ensures this property under standard assumptions [BGV12].

Definition 5.42 (IND-CPA)**.** *Let $\mathcal{E} = (KeyGen, Enc, Dec)$ be a encryption scheme, and let $\mathcal{A}$ be a probabilistic polynomial-time (PPT) adversary. $\mathcal{E}$ is indistinguishable under chosen-*

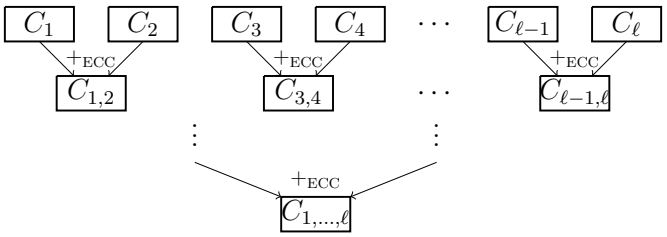
8. See https://hyperelliptic.org/EFD/ for instance.

ECDSA outsourcing protocol

Input: ECDSA parameters, a value $\eta \in \mathbb{N}^*$, η hashes $h(m_i)$ to be signed, and a FHE scheme $\mathcal{E}$, a batching size b for the FHE scheme, a window size ω, and an ECC point representation.
Output: A set of η signatures σ_i respectively on $h(m_i)$.

Offline stage

1. **Only once:** HSM computes all the $2^\omega \ell_{2,n}/\omega$ points $\{P_{i,j}\} = \{\left(i \cdot 2^{\omega(j-1)}\right) G\}_{i=0..2^\omega-1}^{j=1..\ell_{2,n}/\omega}$. Noting $\ell = \ell_{2,n}/\omega$, for $k \in \{1, \ldots, n-1\}$ and $(k^{(1)}, \ldots k^{(\ell)})$ the decomposition of k in base 2^ω we have $kG = P_{k^{(1)},1} + \ldots + P_{k^{(\ell)},\ell}$.
2. HSM draws $k_1, \ldots, k_b \xleftarrow{\$} \{1, \ldots, n-1\}$.
3. For each k_i define $(k_i^{(1)}, \ldots, k_i^{(\ell)})$ as its decomposition in basis 2^ω.
4. For each $j \in \{1, \ldots, \ell\}$, HSM defines $MX_j = (\text{proj}_X(P_{k_1^{(j)},j}), \ldots, \text{proj}_X(P_{k_b^{(j)},j}))$.
5. For each $j \in \{1, \ldots, \ell\}$, HSM defines $MY_j = (\text{proj}_Y(P_{k_1^{(j)},j}), \ldots, \text{proj}_Y(P_{k_b^{(j)},j}))$.
6. HSM computes for each $j \in \{1, \ldots, \ell\}$, $CX_j = \text{Enc}_\mathcal{E}(MX_j)$ and $CY_j = \text{Enc}_\mathcal{E}(MY_j)$.
7. HSM sends $(CX_1, CY_1, \ldots, CX_\ell, CY_\ell)$ to UC.
8. UC generates $CZ_1 = \ldots = CZ_\ell = \text{Enc}_\mathcal{E}((1, \ldots, 1))$, note $C_j = (CX_j, CY_j, CZ_j)$.
9. UC computes homomorphically $C = \textbf{HE.ECC.Add}(C_{j_1}, C_{j_2})$ taking two by two the vectors of encrypted coordinate sets and iterating recursively until there is only one result.



10. UC sends the encrypted result $C_{1,\ldots,l}$ to HSM.
11. HSM decrypts the result and for each $i \in \{1, \ldots b\}$ the coordinates X_i, Y_i, Z_i are used to define a point (x_i, y_i) with $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$.

Online stage

12. For each $h(m)$ to be signed, HSM uses a random k for which he has kG stored.
13. HSM finishes **ECDSA.Sign** by computing (on plaintext data) $y = k^{-1}(h(m) + s \cdot x) \bmod n$.
14. HSM outputs the ECDSA signature (x, y).

FIGURE 5.4 – Full ECDSA outsourced protocol with batching

plaintext attack (IND-CPA) if for all pair of plaintexts (m_0, m_1)*, we have*

$$Adv^{\mathcal{A}} = |Pr\left[\mathcal{A}(Enc(m_0)) = 1\right] - Pr\left[\mathcal{A}(Enc(m_1)) = 1\right]|$$

is negligible.

For signature schemes, the standard definition of security is (existential) unforgeability against chosen message attacks. ECDSA ensures this property in the random oracle model [Vau03].

Definition 5.43 (UF-CMA)**.** *Let* $\mathcal{S} = (KeyGen, Sign, Verif)$ *be a signature scheme, and let* $\mathcal{A}$ *be a probabilistic polynomial-time (PPT) adversary which can sign messages of his choice to produce any message properly signed, with the exception of this message itself.* $\mathcal{S}$ *is unforgeable under chosen messages attack (UF-CMA) if we have*

$$Adv^{\mathcal{A}} = Pr\begin{bmatrix}\mathcal{A}^{Sign(sk,\cdot)}(pk) = (m, \sigma);\\ Verif(pk, m, \sigma) = 1\end{bmatrix}$$

is negligible.

5.5.2 Proof of Security for the Outsourced Protocol

The main goal of this section is to show that the outsourcing protocol does not result in loosing the UF-CMA property when considering the new inputs an attacker may have. In other words we want to show that if the ECDSA is UF-CMA and the encryption scheme is IND-CPA, then the resulting outsourced protocol is also UF-CMA.

The basic idea in this proof is that an adversary able to use the encrypted output with the IND-CPA scheme to break the UF-CMA of the global scheme is also able to break the security of the IND-CPA scheme or able to break the UF-CMA of the ECDSA protocol. Indeed if he is not able to break the UF-CMA of the ECDSA protocol, then he can build a distinguisher for an IND-CPA challenge.

For clarity, the notation related to coordinates has been removed, and a simplified description of the protocol is considered.

Algorithme 18 : $\mathrm{OP}_{\mathrm{ECDSA}}$

Input : $(h(m_i))_{1\leq i\leq b}$
Output : $(\sigma_i)_{1\leq i\leq b} = (r_i, t_i)_{1\leq i\leq b}$

1 HSM generates $k_i \xleftarrow{\$} \{1, \ldots, n-1\}$ in basis 2^ω;

2 HSM sends $\left(\mathrm{Enc}_{\mathcal{E}}\left(\left(P_{k_i^{(j)},j}\right)_{1\leq i\leq b}\right)\right)_{1\leq j\leq \ell}$ to UC ;

3 UC sends $\sum_{j=1}^{\ell} \mathrm{Enc}_{\mathcal{E}}\left(\left(P_{k_i^{(j)},j}\right)_{1\leq i\leq b}\right)$ to HSM;

4 HSM returns $(r_i = x_i \bmod n, t_i = k_i^{-1}(h(m_i) + s \cdot x_i) \bmod n)_{1\leq i\leq b}$ where $(x_i, y_i) = \sum_j P_{k_i^{(j)},j} = k_i G$;

Theorem 5.5.1. *If* $\mathcal{E}$ *is IND-CPA and ECDSA is UF-CMA, then* OP_{ECDSA} *is UF-CMA.*

Proof. To prove security we use the following games.

Game 0: This is $\mathrm{OP}_{\mathrm{ECDSA}}$ itself;

Game 1: This game is the same as **Game 0**, except that the HSM computes k_iG by itself in Step 4.

1. HSM generates $k_i \xleftarrow{\$} \{1,\dots,n-1\}$ in basis 2^ω
2. HSM sends $\left(\mathrm{Enc}_{\mathcal{E}}\left(\left(P_{k_i^{(j)},j}\right)_{1\le i\le b}\right)\right)_{1\le j\le \ell}$ to UC.
3. UC sends $\sum_{j=1}^{\ell}\mathrm{Enc}_{\mathcal{E}}\left(\left(P_{k_i^{(j)},j}\right)_{1\le i\le b}\right)$ to HSM.
4. HSM computes $(x_i,y_i)_{1\le i\le b} = (k_iG)_{1\le i\le b}$
5. HSM returns $(r_i = x_i \bmod n, t_i = k_i^{-1}(h(m_i)+s\cdot x_i) \bmod n)_{1\le i\le b}$

Game 2: This game is the same as **Game 1**, except that all the $P_{k_i^{(j)},j}$ are replaced by 0.

1. HSM generates $k_i \xleftarrow{\$} \{1,\dots,n-1\}$ in basis 2^ω
2. HSM sends $\left(\mathrm{Enc}_{\mathcal{E}}\left((0)_{1\le i\le b}\right)\right)_{1\le j\le \ell}$ to UC.
3. UC sends $\sum_{j=1}^{\ell}\mathrm{Enc}_{\mathcal{E}}\left((0)_{1\le i\le b}\right)$ to HSM.
4. HSM computes $(x_i,y_i)_{1\le i\le b} = (k_iG)_{1\le i\le b}$
5. HSM returns $(r_i = x_i \bmod n, t_i = k_i^{-1}(h(m_i)+s\cdot x_i) \bmod n)_{1\le i\le b}$

Consider a PPT adversary $\mathcal{A}$ having an advantage $Adv_{\mathcal{A}}^{\mathbf{G}_i}$ to produce a forgery in **Game i**, knowing the input, output, transmissions and internal UC processing, such that $Adv_{\mathcal{A}}^{\mathbf{G}_0} = \varepsilon$ and for all games $\mathbf{G}_i$

$$Adv_{\mathcal{A}}^{\mathbf{G}_i} = Pr\left[\begin{array}{l}\mathcal{A}^{\mathbf{G}_i(sk,\cdot)}(pk) = (m,\sigma);\\ \mathrm{Verif}(pk,m,\sigma) = 1\end{array}\right].$$

Clearly $Adv_{\mathcal{A}}^{\mathbf{G}_1} = Adv_{\mathcal{A}}^{\mathbf{G}_0} = \varepsilon$ since the view of the adversary is exactly the same in the two games. Next, by Lemma 5.1, we have that $Adv_{\mathcal{A}}^{\mathbf{G}_1} = Adv_{\mathcal{A}}^{\mathbf{G}_2} + \varepsilon(\lambda)$ where $\varepsilon(\lambda)$ is negligible in the security parameter λ. Finally **Game 2** corresponds to a local ECDSA signature generation, therefore $Adv_{\mathcal{A}}^{\mathbf{G}_2} = Adv_{\mathcal{A}}^{\mathrm{ECDSA}}$ and $Adv_{\mathcal{A}}^{\mathrm{ECDSA}}$ is negligible by the UF-CMA hypothesis of ECDSA. □

Lemma 5.1. *If $\mathcal{E}$ is IND-CPA, then $Adv_{\mathcal{A}}^{\mathbf{G}_1} = Adv_{\mathcal{A}}^{\mathbf{G}_2} + \varepsilon(\lambda)$ where $\varepsilon(\lambda)$ is negligible.*

Proof. Assume that $\varepsilon(\lambda)$ is not negligible. We can then construct a distinguisher $\mathcal{B}$ having success probability $\ge 1/2 + \varepsilon(\lambda)/\ell$ against the semantic security of Enc.

The proof follows from a simple hybrid argument. We define $\ell+1$ different games $\mathbf{G}_{1,j'}$ for $j' = 0,\dots,\ell$ as:

Game (1, j′): This game is the same as **Game 1**, except that all the $P_{k_i^{(j)},j}$'s are replaced by

$$P'_{k_i^{(j)},j} = \begin{cases} 0 & \text{for } j \le j' \\ P_{k_i^{(j)},j} & \text{for } j > j'. \end{cases}$$

1. HSM generates $k_i \xleftarrow{\$} \{1,\dots,n-1\}$ in basis 2^ω

2. HSM sends $\left(\mathrm{Enc}_{\mathcal{E}}\left(\left(P'_{k_i^{(j)},j}\right)_{1\le i\le b}\right)\right)_{1\le j\le \ell}$ to UC.
3. UC sends $\sum_{j=1}^{\ell}\mathrm{Enc}_{\mathcal{E}}\left(\left(P'_{k_i^{(j)},j}\right)_{1\le i\le b}\right)$ to HSM.
4. HSM computes $(x_i, y_i)_{1\le i\le b} = (k_i G)_{1\le i\le b}$
5. HSM returns $(r_i = x_i \bmod n, t_i = k_i^{-1}(h(m_i) + s\cdot x_i) \bmod n)_{1\le i\le b}$

In particular, we have that **Game (1, 0)** is exactly **Game 1** and that **Game $(\mathbf{1}, \ell)$** is exactly **Game 2**. Therefore there exists an index j_0 such that $\mathcal{A}$ has advantage at least $\varepsilon(\lambda)/\ell$ to distinguish between **Game $(\mathbf{1}, \mathbf{j_0})$** and **Game $(\mathbf{1}, \mathbf{j_0+1})$**.

In the following, we construct a distinguisher $\mathcal{B}$ that will include its challenge between these two games.

— $\mathcal{B}$ generates fresh ECDSA keys $(sk'_{\mathrm{ECDSA}}, pk'_{\mathrm{ECDSA}})$

— $\mathcal{B}$ generates $k_i \xleftarrow{\$} \{1, \ldots, n-1\}$ in basis 2^ω

— $\mathcal{B}$ generates all the $P'_{k_i^{(j)},j}$'s as

$$P'_{k_i^{(j)},j} = \begin{cases} 0 & \text{for } j \le j_0 \\ P_{k_i^{(j)},j} & \text{for } j > j_0. \end{cases}$$

— $\mathcal{B}$ asks for a challenge on messages $P_0 = 0$ and $P_1 = P'_{k_i^{(j)},j_0+1}$, and receives a challenge ciphertext $\mathrm{Enc}\,(P_\beta)$ for an unknown $\beta \in \{0, 1\}$

— Then $\mathcal{B}$ encrypts the $P'_{k_i^{(j)},j}$, but replaces the encryption of $P_{k_i^{(j)},j_0+1}$ by the challenge. We denote by E the resulting set of ciphertexts. Note that if the challenge is encrypting $P_0 = 0$, then the set E corresponds to the set of ciphertexts of **Game $(\mathbf{1}, \mathbf{j_0+1})$**, and otherwise it corresponds to the set of ciphertexts of **Game $(\mathbf{1}, \mathbf{j_0})$**.

— Next $\mathcal{B}$ runs $\mathcal{A}$ with E in the ECDSA protocol, and gets its answer (m, σ)

— if $\mathrm{Verif}(pk'_{\mathrm{ECDSA}}, m, \sigma) = 1$

then $\mathcal{B}$ returns 0

else $\mathcal{B}$ returns 1.

If $b = 0$, this means that the setting was that of **Game $(\mathbf{1}, \mathbf{j_0})$**, and therefore P_1 was encrypted in the challenge; however it was $P_0 = 0$. This contradicts the IND-CPA security of the $\mathcal{E}$ encryption scheme (because ℓ is logarithmic in λ) and concludes the proof. □

5.5.3 Malicious adversaries

The proofs presented above consider semi-honest (also known as honest-but-curious) adversaries. In particular, it supposes that the UC behaves as expected during the protocol execution. A malicious, active, UC might divert from the protocol so that the elliptic curve points used for the ECDSA signature are not correct. With such an attack, the signatures that are published do not follow the ECDSA protocol, and therefore we cannot say they do not reveal enough information for an attacker to break the system.

For example, if the cloud sends back the k_i's as the x coordinates of the obtained points, the HSM will reveal the k_i's and the attacker will immediately be able to retrieve the secret key used for the signature.

It is possible to modify the protocol so as to ensure resistance against malicious adversaries at a very high cost, but in practice a simpler business-like solution is possible for a lower cost (in practice, most companies will opt for the latter setting). Namely, one can set up traps by including replication in the k_i's, *i.e.* instead of sampling b different k_i's randomly and batching them all together, to sample b/κ different k_i's and to repeat each of the k_i's in κ different components. (In practice, a few replicas will be enough.) In order to cheat without getting caught in this setting, the malicious UC will have to be consistent with the unknown geometry and his probability of success will therefore be small (similar to the *covert model* of [Lin13]). In order to make it worthless for UC to be malicious, the businesses can set up a contract with the UC in which the cost of getting caught when cheating is more important than the possible gain.

5.6 Implementation Details

We implemented prototypes of our outsourcing protocol. The code was written in C++ using the open-source HElib library [9] [HS14, HS15], which in turn uses NTL [Sho15]. Our implementation will be made available under the GNU GPL license.

5.6.1 Parameters Constraints and Extension of HElib

5.6.1.1 The P-256 Curve

The commercially available HSMs we considered (cf. Tab. 5.1) usually implement NIST's P-256 Elliptic Curve. Since our goal is to demonstrate the feasibility of our outsourcing protocol, we chose to focus on the same curve. However, we would like to emphasize that this does not help us: the P-256 curve is quite bad for our setting! On the contrary, curves over binary fields, Koblitz curves or Edwards curves might yield much faster protocols given their small characteristic and their point addition formulae. The study of the fastest elliptic curve to be used in combination with homomorphic encryption is certainly an interesting theoretical open problem orthogonal to this work.

Recall that the P-256 curve is the elliptic curve

$$(E_{\text{P-256}})\colon y^2 = x^3 - 3x + b \in \mathbb{F}_{p_{\text{P-256}}}$$

where

$$\begin{cases} p_{\text{P-256}} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\ b &= 41058363725152142129326129780047268409114441015993725554835256314039467401291 \end{cases}$$

5.6.1.2 Extension of HElib to Large p's

The HElib library only handles plaintext space modulus p^r with p and r in single precision. For our protocol however, since we considered the elliptic curve P-256, we have to work with plaintext space modulus $p = p_{\text{P-256}}$; we therefore modified HElib to handle that case. [10]

9. https://github.com/shaih/HElib.

10. A similar modification had been performed in https://github.com/dwu4/fhe-si, but was based on an earlier version of HElib without many of the recent improvements (special primes, etc.).

5.6.1.3 Selection of BGV Parameters

HElib automatically selects all the parameters for the BGV scheme from the tuple (m, p, L) where m is the index of the cyclotomic polynomial $\Phi_m(x)$, the plaintext space modulus p and L the multiplicative depth of the circuit to be homomorphically evaluated. As described in Sec. 5.2.2, our ECDSA protocol needs to perform batching, *i.e.* needs to embed several plaintext elements per ciphertext. Now, for every (m, p), the cyclotomic polynomial $\Phi_m(x)$ factors modulo p into a product of irreducible factors $\Phi_m(x) = \prod_{j=1}^{b} F_j(x) \pmod p$ for a $b = b(m, p) \in \{1, \ldots, \deg(\Phi_m(x)) = \phi(m)\}$ where ϕ is Euler's totient function. Unfortunately, not every tuple (p, m) yield an efficient batching ; for example $p = p_{\text{P-256}}$ and $m = 1031$ do not allow batching at all — cf. Tab. 5.1.

m	1031	1531	2048
$\phi(m)$	1030	1024	1024
b	1	1024	512

TABLE 5.1 – Degree $\phi(m)$ of the cyclotomic polynomial $\Phi_m(x)$, and number b of factors thereof modulo $p = p_{\text{P-256}}$.

In particular, full batching will be possible when $b = \phi(m)$, *i.e.* when all the roots of $\Phi_m(x)$ are in $\mathbb{Z}_p$. Since $\Phi_m(x)$ divides $x^m - 1$, all the roots of $\Phi_m(x)$ will be in $\mathbb{Z}_p$ when $m \mid p - 1$. Now we have that

$$p_{\text{P-256}} - 1 = 2 \cdot 3 \cdot 5^2 \cdot 17 \cdot 257 \cdot 641 \cdot 1531 \cdot 65537 \cdot 490463 \cdot 6700417 \cdot p'$$

for $p' = 83594504224[...]3916927241$ a large prime. Finally, since our goal was to batch about 1000 signatures (cf. Sec. 5.3.1), we select $m = 1531$ and by Tab. 5.1, this yields a scheme that works with polynomials of degree $1024 - 1 = 1023$, and that can batch 1024 elements.

5.6.2 Windowing

In our outsourcing protocol, one can select different window sizes ω to encrypt the scalars k_i's. A larger window size will increase the memory used but will decrease the communication cost — cf. Tab. 5.2. Also with larger windows, the UC will have to perform less homomorphic computations. The limiting factor is therefore the memory used by the HSM. However, this memory is used to store encrypted values and thus does not have to be tamper resistant (it can even be outsourced on the HSM's host). [11]

More precisely, the k_i's are decomposed in basis 2^ω, and give $2\lambda/\omega$ elliptic curve points represented by the two coordinates (x, y) in $\mathbb{F}_{p_{\text{P-256}}}$. Note that the decomposition is easy to compute: it simply consists of the $2\lambda/\omega$ successive sequences of ω bits of the k_i's binary decompositions. The precomputation costs are as follow:

— $(2\lambda)^2/\omega \cdot 2^\omega \cdot 2$ bits in (unprotected) memory storage for the HSM, and

— $2\lambda/\omega \cdot 2 \cdot \zeta_{\mathcal{E}, p_{\text{P-256}}}$ bits in communication between the HSM and the UC, where $\zeta_{\mathcal{E}, p_{\text{P-256}}}$ denotes the bits needed to encrypt a coordinate in $\mathbb{F}_{p_{\text{P-256}}}$ with the BGV scheme $\mathcal{E}$.

Moreover, the addition of the encrypted coordinates (which we denote $\text{Enc}_{\mathcal{E}}(x)$ and $\text{Enc}_{\mathcal{E}}(y)$) of the elliptic points received by the UC has a multiplicative depth linear in d_{add}

11. To verify that the stored values have not been modified, a MAC can be stored alongside these values.

Parameters	$\lambda = 128$		
	$\omega = 8$	$\omega = 16$	$\omega = 32$
Enc. points sent	32	16	8
Com. cost (bits)	$64 \cdot \zeta_{\mathcal{E},q}$	$32 \cdot \zeta_{\mathcal{E},q}$	$16 \cdot \zeta_{\mathcal{E},q}$
Outsourced Mem. (HSM)	512KB	64MB	2TB
Mult. depth	$5d_{add}$	$4d_{add}$	$3d_{add}$

TABLE 5.2 – Impact of Windowing over storage and communication costs. λ is the bit-security level, ω the bit-size of the window, and $\zeta_{\mathcal{E},q}$ is the number of bits required to encrypt an element of $\mathbb{F}_q$.

Window Size	Mult. depth	# Elliptic Curve Additions	UC Time (s)	HSM → UC	UC → HSM	HSM Outsourced Memory
				Com. Cost	Com. Cost	
8	2	16	24.8s	156.3 Mb	82.4 Mb	512 KB
8	4	16 + 8	61.2s	236.2 Mb	36.1 Mb	512 KB
8	6	16 + 8 + 4	102s	314.9 Mb	14.8 Mb	512 KB
8	8	16 + 8 + 4 + 2	148s	391.4 Mb	5.82 Mb	512 KB
8	10	16 + 8 + 4 + 2 + 1	208s	471.9 Mb	2.25 Mb	512 KB
16	2	8	12.4s	78.2 Mb	41.2 Mb	64 MB
16	4	8 + 4	30.1s	118.1 Mb	18.0 Mb	64 MB
16	6	8 + 4 + 2	51.6s	157.4 Mb	7.4 Mb	64 MB
16	8	8 + 4 + 2 + 1	74.4s	195.7 Mb	2.9 Mb	64 MB
32	2	4	6.2s	39.1 Mb	20.6 Mb	2 TB
32	4	4 + 2	15.4s	59.1 Mb	9.0 Mb	2 TB
32	6	4 + 2 + 1	25.9s	78.7 Mb	3.7 Mb	2 TB

TABLE 5.3 – Benchmarkings of the ECDSA outsourcing protocol on a `c4.8xlarge` AWS instance for different window sizes and evaluations of different multiplicative depths — "UC Time (s)" denotes the benchmarking of the homomorphic evaluation consisting of "# Elliptic Curve Additions" additions of points over the curve P-256, we also include the communication and storage costs.

and a number of multiplications that is closely related to the representation of those points (in our case, we have $d_{add} = 2$, cf. Sec. 5.4.4). It is precisely this depth that determines the size of the BGV parameters $params_{\mathcal{E}}$ and hence, the size of data that is sent. Tab. 5.2 shows the impacts of different window sizes over communications and memory usage.

As the UC receives $\ell_{2,n}/\omega$ points to add, his addition circuit has depth $d = \log_2(\ell_{2,n}/\omega)$, which implies that our homomorphic encryption scheme $\mathcal{E}$ must be able to handle d times the depth d_{add} of an homomorphic elliptic curve point-addition.

Finally, we can reduce UC's computational cost of the signature generation at the expense of an additional work factor for the HSM by allowing UC to abort the signature prematurely and let the HSM finish the job. This optimization is discussed in the next Section.

5.6.3 Early aborts

In our outsourcing protocol, the UC compute homomorphically the coordinates of

$$kG = \left(\sum_{j=1}^{\ell_{2,n}/\omega} k^{(j)} 2^{\omega j}\right) G = \sum_{j=1}^{\ell_{2,n}/\omega} P_{k^{(j)},j}\,,$$

given the encryptions of the coordinates of the $P_{k^{(j)},j}$. In order to reduce the total depth of the point additions computation (namely $\log_2(\ell_{2,n}/\omega) \cdot d_{add}$ using a binary tree), the UC can perform the addition up to a specific multiplicative depth and send back the results (the coordinates of the partial sums) to the HSM. The latter will have to finish the point addition computation over the plaintexts. This in terms allows to lower the multiplicative depth capability of the BGV scheme at the expense of increasing the number of encrypted coordinates sent to the HSM. The resulting trade-off depends mainly on the ciphertext expansion. Notice that the more computation does the UC, the larger is the multiplicative depth and the ciphertext expansion, but the less ciphertexts are sent.

In our final evaluation, we will consider different early abortions trade-off — cf. Sec. 5.6.4.

5.6.4 Full Evaluation

Our prototype was deployed on a commercially available cloud computing service — namely a `c4.8xlarge` AWS instance, with turboboost turned off and running on a single thread, while the HSM was laptop-emulated. We benchmarked on different window sizes $\omega = 8, 16, 32$ and different early aborts (corresponding to resp. multiplicative depths $1 \cdot d_{add}$, $2 \cdot d_{add}$, $\ldots$, $5 \cdot d_{add}$).

We consider communications costs from the HSM to the UC (encrypted coordinates of the points to be added), and from the UC to the HSM (encrypted results). Of course we also consider computational costs for the HSM and UC. All the results are given in Tab. 5.3.

One of the most interesting compromises is a windowing size of 16 bits and full additions (no early abort) which can be executed in 74 seconds on a single thread and results in 200Mbits of uploaded data and 3Mbits of downloaded data. In order to use this window size the HSM also needs to access an internal or locally outsourced memory cache of 32MBytes.

Note that all the operations done by the UC are coordinate wise over vectors of 1024 coordinates given the encryption scheme parameters chosen. Therefore we can suppose that the computation can benefit from a linear gain in a multi-threaded environment with a large amount of threads. It would be possible to send the computation to two `c4.8xlarge` servers which can handle each 36 threads and thus get a reply in roughly a second.

For the HSM the main computational limit is the throughput at which it can produce the encrypted flow to be sent to the UC and decrypt the flow that comes from the UC. Our client could produce an encryption flow of 800Mbits/s and decrypt an incoming flow of 1.1Gbits/s. Such processing speeds would allow using 8 `c4.8xlarge` instances and retrieve thus up to $4 \cdot 1024$ encrypted points per second.

Note that if the UC is in the cloud and the HSM requires using it at its maximum throughput, having a constant upload bandwidth usage of 800Mbits/s can be quite costly. The cloud would be more adapted for a sporadic usage. If the processing power is required often, installing computing blades locally (through a LAN or PCI connection) would be a much more interesting strategy.

5.7 Conclusion and Perspectives

In this chapter, we described a protocol that allows a constrained device processing ECDSA signatures to benefit from an outside source of computational power, in a secure way. The attainable performance goes beyond most high-end existing HSMs, except for the HSM available as an option on IBM's z13 supercomputer.

Meanwhile, all the real life use-cases aforementioned require a FIPS 140-2 certification, which completely vanishes from the moment we encrypt data using a non FIPS approved homomorphic scheme. Nevertheless, we proved that such schemes could actually improve the signing throughput, introducing flexibility and increasing longevity. Such an improvement provides an interesting alternative to running parallel HSMs, and we therefore believe that our contribution may be a motivating example toward standardizing lattice-based homomorphic cryptography.

Future work will mostly consist in design a similar protocol for RSA and implementing the smartcard use-case to gauge its concrete feasibility.

Also, besides the optimizations proposed above, we believe it possible to further speed-up the constrained device using dedicated hardware, for elliptic curve point addition but our experiments do not cover that range.

Appendix

5.A Complete Addition Formula in Multiplicative Depth 2

Fig. 5.A.1 below shows that the complete addition formula of [RCB15, Algorithm 4] has multiplicative depth 2.

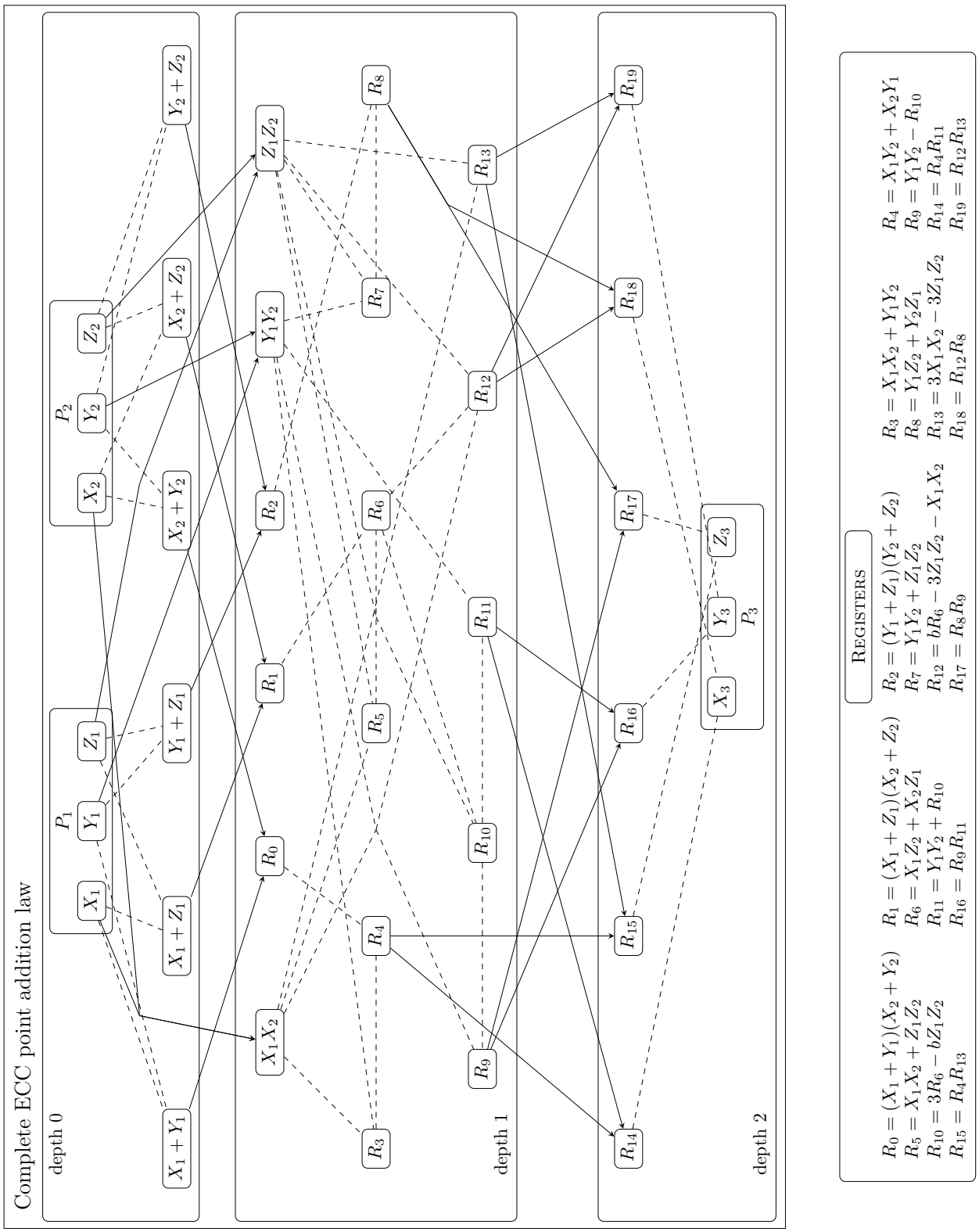


FIGURE 5.A.1 – Complete addition using [RCB15, Algorithm 4]: only 3 coordinates, multiplicative depth 2, and 12 multiplications (see original algorithm for this). Dashed lines correspond to additions, filled ones to multiplications. We note this algorithm **ECC.Add** and its homomorphic encryption version **HE.ECC.Add**.

Troisième partie

Cryptographie basée sur les Codes Correcteurs d'Erreurs

Chapitre 6

Cryptosystème Efficace Sans Masquage

Sommaire

Contexte

Dans ce chapitre, nous proposons une nouvelle méthode pour construire des cryptosystèmes efficaces basés sur les codes correcteurs d'erreurs, et qui ne nécessite pas de masquer le code utilisé. Cette méthode est inspirée d'un schéma présenté par Alekhnovich en 2003, basé sur l'utilisation de matrices aléatoires. Cependant, notre approche est légèrement différente et optimisée afin de ramener la sécurité de nos constructions au problème du décodage de codes quasi-cycliques aléatoires.

Nous proposons deux cryptosystèmes construits avec cette méthode : Hamming Quasi-Cyclic (HQC), basé sur la métrique de Hamming, et Rank Quasi-Cyclic (RQC), basé sur la métrique rang. En plus de la preuve de sécurité, qui réduit l'indistinguabilité contre les attaques à clairs choisis au problème de décodage des familles de codes quasi-cycliques, nous

fournissons également une analyse détaillée de la probabilité d'échec de déchiffrement de notre schéma en métrique de Hamming.

Notre schéma bénéficie d'un algorithme de déchiffrement très rapide ainsi que de tailles de clés de seulement quelques milliers de bits. Les cryptosystèmes proposés sont efficaces pour de faibles taux de chiffrement, et de fait particulièrement appropriés pour de l'échange de clés et de l'authentification. Asymptotiquement, pour un paramètre de sécurité λ, les clés publiques ont pour tailles respectives en $\mathcal{O}(\lambda^2)$ pour HQC et en $\mathcal{O}(\lambda^{\frac{4}{3}})$ pour RQC.

Ce travail est issu d'une collaboration avec Carlos Aguilar Melchor, Olivier Blazy, Philippe Gaborit, et Gilles Zémor.

6.1 Introduction

6.1.1 Background and Motivations

The first code-based cryptosystem was proposed by McEliece in 1978. This system which can be seen as a general encryption setting for coding theory is based on a hidden trapdoor associated to a decodable family of codes, hence a strongly structured family of codes. The inherent construction of the system makes it difficult to obtain a security reduction to a general problem of decoding random codes with a proven reduction. Even if the original Goppa family is still today considered as secure, many other families of codes (Reed-Solomon codes, Reed-Muller codes or some alternant codes [MB09, BCGO09]) were broken in polynomial time by recovering the hidden structure [FOPT10]. The fact that an attacker has potentially the possibility to recover the hidden structure of the code (even possibly for Goppa codes) lies like a sword of Damocles over the system. Finding a cryptosystem based on general instances of random codes has always been a major issue in code-based cryptography. Recently the MDPC cryptosystem [MTSB13] (in the spirit of the NTRU cryptosystem [HPS98]) proposed an approach in which the hidden code has only a weak structure compared to strongly structured families of decodable codes. A similar cryptosystem with the same type of approach based was also proposed in [GMRZ13]. Beside this weak hidden structure, the MDPC system has very nice features and in particular relatively small key sizes, because of the cyclic structure of the public matrix. Even if this system is a strong step forward for code-based cryptography, the weak hidden structure problem remains.

In 2003, Alekhnovich proposed an innovative approach based on the LPN problem in which the public matrix had no hidden trapdoor [Ale03], even if the system was not efficient, the approach in itself was a breakthrough. And since, other systems based on the LPN problem have been proposed [DV13, KMP14], also a ring version (ring-LPN) is also introduced in [HKL+12] for authentication and for encryption in [DP12]. The Alekhnovich's approach was later the basis for the so called Learning With Errors (LWE) lattice-based cryptosystem by Regev [Reg03]. Again the first version of the LWE problem was not very efficient but introducing more structure in the public key (as for NTRU) leads to the very efficient Ring-LWE cryptosystem [LPR10b], one strong feature of this latter paper is that it gives a reduction from the decisional version of the ring-LWE problem to a search version of the problem. Such a reduction is not known for the case of the ring-LPN problem.

In this paper, starting from the Alekhnovich approach we propose the first efficient cryptosystem based on random quasi-cyclic matrices. Our construction benefits from very nice features: a reduction to the general problem of decoding random quasi-cyclic codes, hence with no hidden structure, and also very good parameters and efficiency. Since our approach is relatively general it can also be used with different metrics such as the rank

metric. At last another strong feature of our system is that its inherent structure permits to give a precise analysis of the decryption failure probability, which is also a hard point for the MDPC cryptosystem and is not done in detail for other approaches based on the LPN problem. The only relative weakness of our system is the relatively low encryption rate, but this is not a major issue since our system remains very efficient for classical applications of encryption schemes such as authentication or key exchange.

6.1.2 Our Contributions

We propose the first efficient code-based cryptosystem whose security relies on decoding small weight vectors of random quasi-cyclic codes. We provide a reduction of our cryptosystem to this problem together with a detailed analysis of the decryption failure probability. Our analysis allows us to give small parameters for code-based encryption in Hamming and Rank metrics. When compared to the MDPC [MTSB13] or LRPC [GMRZ13] cryptosystems, our proposal offers greater semantic security and better decryption guarantees for similar parameters, but with a lower encryption rate. Overall we propose concrete parameters for different level of security, for classical and quantum security, these parameters show the great potential of rank metric for cryptography especially for high level of security. When compare to the ring-LPN based cryptosystem [DP12] our system has better parameters with a factor 10 and 100 respectively for the size of the ciphertext and the size of the public key. We also give a general table comparing the different asymptotic sizes for different code-based cryptosystems.

6.1.3 Overview of Our Techniques

Our cryptosystem is based on two codes. A first code $\mathcal{C}[n', k']$, for which an efficient decoding algorithm $\mathcal{C}.\mathsf{Decode}(\cdot)$ is known, the code $\mathcal{C}$ together with its generator matrix $\mathbf{G}$ are publicly known. The second code used is a $[2n, n]$ (with $n \geq n'$) random double-circulant code in systematic form, with generator matrix $\mathbf{D} = (\mathbf{I}_n \mid \mathbf{rot}(\mathbf{d}))$ (see Eq. (2.33) for the definition of $\mathbf{rot}(\cdot)$). The general idea of the system is that the double-circulant code is used to generate some noise, which can be handled and decoded by the code $\mathcal{C}$. Notice that in practice, we have $n' \sim n$. In some sense the system can be seen as a noisy adaptation of the ElGamal cryptosystem.

The secret key for our cryptosystem is a *short* vector $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$ (for some metric), whose syndrome $\mathbf{s}^\top = \mathbf{D} \cdot (\mathbf{x}, \mathbf{y})^\top$ is appended to the public key $\mathsf{pk} = (\mathbf{G}, \mathbf{D}, \mathbf{s}^\top)$. To encrypt a message $\boldsymbol{\mu}$ belonging to some plaintext space, it is first encoded through the generator matrix $\mathbf{G}$, then hidden using the syndrome $\mathbf{s}$ and an additional short vector $\boldsymbol{\epsilon}$ to prevent information leakage. In other words, encrypting a message simply consists in providing a noisy encoding of it with a particular shape. Formally, we have $\boldsymbol{\rho} = \boldsymbol{\mu}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \boldsymbol{\epsilon}$ for a short random vector $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2)$, and for some natural operator $\cdot$ defined in chapter 2 (see Def. 3.30). The ciphertext is $(\mathbf{v} = \mathbf{r}\mathbf{D}^\top, \boldsymbol{\rho})$. The legitimate recipient owning sk can recover the plaintext using the efficient decoding algorithm $\mathcal{C}.\mathsf{Decode}$ on input $\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y}$.

For correctness, all previous constructions rely on the fact that the error term added to the encoding of the message is less than or equal to the decoding capability of the code being used. In our construction, this assumption is no longer required and the correctness of our cryptosystem is guaranteed assuming the legitimate recipient can remove sufficiently many errors from the noisy encoding $\boldsymbol{\rho}$ of the message using sk.

The above discussion leads to study the probability that a decoding error occurs, which would yield a decryption failure. In order to provide strong guarantees on the correctness of

the cryptosystem we propose, for the Hamming metric, a detailed analysis of the error vector distribution is provided in Sec. 6.5. In a nutshell, the coordinates of an n-dimensional vector $\mathbf{v}$ of weight w (for some weight function ω) are assumed to follow independent Bernoulli distributions of parameter $p = \frac{w}{n}$. With such an assumption, the Hamming weight $\omega(\mathbf{v})$ follows a binomial distribution of parameters n and p. While this might not be perfectly accurate since the weight hence have a standard deviation of $n \times p = w$ instead of 0, we manage to lower this standard deviation sufficiently so that this model is accurate enough for our purposes (see Sec. 6.5 for details).

Comparison to McEliece framework. In the McEliece encryption framework, a hidden code is considered. This leads to two important consequences: first, the security depends on hiding the structure of the code, and second the decryption algorithm corresponds to the hidden code and cannot be changed. This yields different instantiations depending on the code being hidden, for which as we know many of them are attacked and a few resist.

In our framework there is not one unique hidden code, but two independent codes, the random double-circulant structure assures the security of the scheme, and the public code $\mathcal{C}$ assures the decryption. It permits in particular to consider public families of codes which are difficult to hide but very efficient for decoding, also it demands to find a tradeoff for the code $\mathcal{C}$, between efficiency for decoding and practical decoding complexity, but at the difference of the McEliece scheme, where the decoding code is fixed, it can be changed for our scheme depending of the application.

The global decryption failure for our scheme depends on the articulation between the error-vector distribution induced by the double-circulant code and the decoding algorithm $\mathcal{C}.\mathsf{Decode}(\cdot)$. After having studied the error-vector distribution for Hamming metric we link it with a particular code adapted for low dimensional rates and error rate of order 1/3. Notice that the system could possibly be used for greater encryption rate at the cost of higher parameters. This led us considering tensor product codes: the composition of two linear codes. Tensor product codes are defined (Def. 6.44) in Sec. 6.6, and a detailed analysis of the decryption failure probability for such codes is provided there. For the case of rank metric, we consider Gabidulin codes and the case where the error-vector is always decodable, with null decryption failure.

6.1.4 Road Map

The rest of the chapter is organized as follows: Sec. 6.3 describes the cryptosystem we propose and its security is discussed in Sec. 6.4. Sec. 6.5 and 6.6 study the decryption failure probability and the family of tensor product codes we consider to perform the decoding for small rate codes. Finally, Sec. 6.7 give parameters.

6.2 Preliminaries

Most of the necessary background to understand this chapter was introduce in chapter 2.

6.3 A New Encryption Scheme

We begin this Section by describing a generic version of the proposed encryption scheme, this description does not depend on the particular metric used. The particular case of Hamming metric is denoted by HQC (for Hamming Quasi-Cyclic) and RQC (for Rank

Quasi-Cyclic) in the case of rank metric. Parameter sets for binary Hamming Codes and Rank Metric Codes can be respectively found in Sec. 6.7.1 and 6.7.2.

Presentation of the scheme. Recall from the introduction that the scheme uses two types of codes, a decodable $[n', k']$ code which can correct δ errors and a random double-circulant $[2n, n]$ code. In practice n' is slightly smaller or equal to n. In the following, we assume $\mathcal{V}$ is a vector space on some field $\mathbb{F}$, ω is a norm on $\mathcal{V}$ and for any $\mathbf{x}$ and $\mathbf{y} \in \mathcal{V}$, their distance is defined as $\omega(\mathbf{x} - \mathbf{y}) \in \mathbb{R}^+$. Now consider a linear code $\mathcal{C}$ over $\mathbb{F}$ of dimension k and length n (generated by $\mathbf{G} \in \mathbb{F}^{k \times n}$), that can correct up to δ errors via an efficient algorithm $\mathcal{C}.\mathsf{Decode}(\cdot)$. The scheme consists of the following four polynomial-time algorithms:

— $\mathsf{Setup}(1^\lambda)$: generates the global parameters $n = n(\lambda), n' = n'(\lambda), k' = k'(\lambda)$, $\delta = \delta(\lambda)$, and $w = w(\lambda)$. The plaintext space is $\mathbb{F}^{k'}$. Outputs $\mathsf{param} = (n, n', k', \delta, w)$.

— $\mathsf{KeyGen}(\mathsf{param})$: generates $\mathbf{d} \xleftarrow{\$} \mathcal{V}$, matrix $\mathbf{D} = (\mathbf{I}_n \mid \mathbf{rot}(\mathbf{d}))$, the generator matrix $\mathbf{G} \in \mathbb{F}^{k' \times n'}$ of $\mathcal{C}$, $\mathsf{sk} = (\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\mathbf{x}), \omega(\mathbf{y}) = w$, sets $\mathsf{pk} = \left(\mathbf{G}, \mathbf{D}, \mathbf{s} = \mathsf{sk} \cdot \mathbf{D}^\top\right)$, and returns $(\mathsf{pk}, \mathsf{sk})$.

— $\mathsf{Encrypt}(\mathsf{pk} = (\mathbf{G}, \mathbf{D}, \mathbf{s}), \boldsymbol{\mu}, \theta)$: uses randomness θ to generate $\boldsymbol{\epsilon} \xleftarrow{\$} \mathcal{V}$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ such that $\omega(\boldsymbol{\epsilon}), \omega(\mathbf{r}_1), \omega(\mathbf{r}_2) \leq w$, sets $\mathbf{v}^\top = \mathbf{D}\mathbf{r}^\top$ and $\boldsymbol{\rho} = \boldsymbol{\mu}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \boldsymbol{\epsilon}$. It finally returns $\mathbf{c} = (\mathbf{v}, \boldsymbol{\rho})$, an encryption of $\boldsymbol{\mu}$ under pk.

— $\mathsf{Decrypt}(\mathsf{sk} = (\mathbf{x}, \mathbf{y}), \mathbf{c} = (\mathbf{v}, \boldsymbol{\rho}))$: returns $\mathcal{C}.\mathsf{Decode}(\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y})$.

Notice that the generator matrix $\mathbf{G}$ of the code $\mathcal{C}$ is publicly known, so the security of the scheme and the ability of decrypting do not rely on the knowledge of the error correcting code $\mathcal{C}$ being used.

Correctness. The correctness of our new encryption scheme clearly relies on the decoding capability of the code $\mathcal{C}$. Specifically, assuming $\mathcal{C}.\mathsf{Decode}$ correctly decodes $\boldsymbol{\rho} - \mathbf{v} \cdot \mathbf{y}$, we have:

$$\mathsf{Decrypt}\left(\mathrm{sk}, \mathsf{Encrypt}\left(\mathrm{pk}, \boldsymbol{\mu}, \theta\right)\right) = \boldsymbol{\mu}. \tag{6.1}$$

And $\mathcal{C}.\mathsf{Decode}$ correctly decodes $\boldsymbol{\rho} - \mathbf{x} \cdot \mathbf{y}$ whenever

$$\omega(\mathbf{s} \cdot \mathbf{r}_2 - \mathbf{v} \cdot \mathbf{y} + \boldsymbol{\epsilon}) \leq \delta \tag{6.2}$$
$$\omega((\mathbf{x} + \mathbf{h} \cdot \mathbf{y}) \cdot \mathbf{r}_2 - (\mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2) \cdot \mathbf{y} + \boldsymbol{\epsilon}) \leq \delta \tag{6.3}$$
$$\omega(\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}) \leq \delta \tag{6.4}$$

In order to upper bound the decryption failure probability, a complete analysis of the distribution of the error vector $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$ is provided in Sec. 6.5.

6.4 Security of the Scheme

In this section we prove the security of our scheme, the proof is generic for any metric, and the security is reduced to the respective quasi-cyclic problems defined for Hamming and rank metric in Section 2.

Theorem 6.4.1. *The scheme presented above is* IND-CPA *under the* 2-*DQCSD and* 3-*DQCSD assumptions.*

Démonstration. To prove the security of the scheme, we are going to build a sequence of games transitioning from an adversary receiving an encryption of message $\boldsymbol{\mu}_0$ to an adversary receiving an encryption of a message $\boldsymbol{\mu}_1$ and show that if the adversary manages to distinguish one from the other, then we can build a simulator breaking the DQCSD assumption, for QC codes of order 2 and 3 (codes with parameters $[2n, n]$ or $[3n, 2n]$), and running in approximately the same time.

Game $\boldsymbol{G}_0$: This is the real game, we run an honest KeyGen algorithm, and after receiving $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1)$ from the adversary we produce a valid encryption of $\boldsymbol{\mu}_0$.

Game $\boldsymbol{G}_1$: We now start by forgetting the decryption key sk, and taking $\mathbf{s}$ at random, and then proceed honestly. Under the DQCSD assumption, this game is indistinguishable from the previous one.

Game $\boldsymbol{G}_2$: Now that we no longer know the decryption key, we can start generating random ciphertexts. So instead of picking correctly weighted $\mathbf{r_1}, \mathbf{r_2}, \boldsymbol{\epsilon}$, the simulator now picks random vectors in the full space.

The adversary view is:

$$\begin{pmatrix} \mathbf{v} \\ \boldsymbol{\rho} - \boldsymbol{\mu}_0\mathbf{G} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathbf{rot(d)} \\ \mathbf{0} & \mathbf{I}_n & \mathbf{rot(s)} \end{pmatrix} \cdot (\mathbf{r_1}, \boldsymbol{\epsilon}, \mathbf{r_2})^\top$$

When $\mathbf{r_1}, \mathbf{r_2}, \boldsymbol{\epsilon}$ are sampled uniformly at random, $\mathbf{v}$ and $\boldsymbol{\rho}$ are also random. Hence honestly generated ciphertexts are indistinguishable from invalid ones under the DQCSD assumption (applied to a $2n \times 3n$ QC matrix of order 3).

Game $\boldsymbol{G}_3$: We now proceed in the other way. Given random $\mathbf{r_1}$, $\mathbf{r_2}$, and $\boldsymbol{\epsilon}$, messages $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$, and $\mathbf{v}, \boldsymbol{\rho}$ computed as before, one can find values $\mathbf{r'_1}, \mathbf{r'_2}, \boldsymbol{\epsilon'}$ such that $\mathbf{v}^\top = \mathbf{D}\mathbf{r'}^\top$ and $\boldsymbol{\rho} = \boldsymbol{\mu}_1\mathbf{G} + \mathbf{s} \cdot \mathbf{r'}_2 + \boldsymbol{\epsilon'}$.

Hence, the previous invalid encryption of $\boldsymbol{\mu}_0$ is perfectly indistinguishable from a fake encryption of $\boldsymbol{\mu}_1$.

Game $\boldsymbol{G}_4$: In this game, we now pick $\mathbf{r'_1}, \mathbf{r'_2}, \boldsymbol{\epsilon'}$ with the correct weight, as explained in Game $\boldsymbol{G}_2$, under DQCSD this is indistinguishable from the previous game.

Game $\boldsymbol{G}_5$: We now conclude the proof, by switching the public key to an honestly generated one, and like in Game $\boldsymbol{G}_1$, this game is indistinguishable from the previous one, under DQCSD.

We managed to build a sequence of games allowing a simulator to transform a ciphertext of a message $\boldsymbol{\mu}_0$ to a ciphertext of a message $\boldsymbol{\mu}_1$. Hence the advantage of an adversary against the IND-CPA experiment is bounded:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) \leq 2 \cdot \left(\mathsf{Adv}^{\text{2-DQCSD}}(\lambda) + \mathsf{Adv}^{\text{3-DQCSD}}(\lambda)\right). \tag{6.5}$$

□ □

6.5 Analysis of the Error Vector Distribution for HQC

The aim of this Section is to determine the probability that the condition in Eq. (6.4) holds. In order to do so, we study the error distribution of the error vector $\mathbf{e} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$. In the following, we denote by $\mathbb{E}_\omega[n, w, \epsilon]$, the expected weight of the vector $\mathbf{e}$ defined before. Using the linearity of Expectation we have that the expected weight of the error vector is

simply n times the expected weight E of a coordinate. From that we have $\mathbb{E}_w[n, w, \epsilon] = n \cdot E$, and this allows to focus on a single coordinate in our analysis.

Now, as mentioned in Sec. 6.1.3, the first assumption made in the following analysis is that each coordinate of an n-dimensional vector $\mathbf{v}$ of weight $\omega(\mathbf{v}) = w$ follows a the Bernoulli distribution of parameter $p = \frac{w}{n}$, which we denote $v_i \sim \mathcal{B}(p)$, with:

$$\Pr[v_i = c] = \begin{cases} p \text{ if } c = 1, \\ 1 - p \text{ if } c = 0, \\ 0 \text{ otherwise.} \end{cases} \tag{6.6}$$

We split the analysis into three parts: the first one consists in establishing the distribution of a coordinate of the product of two vectors, then we study the distribution of the sum of two such product vectors, and end with the distribution of the error vector $\mathbf{e}$ by adding vector $\boldsymbol{\epsilon}$.

Step 1. The distribution of the product of two random variables following a Bernoulli distribution is given by the following proposition. We denote by $\mathcal{B}(n, p)$ the binomial distribution which is the repetition of n successive independent samples of $\mathcal{B}(p)$.

Proposition 5.1. *Let* $\mathbf{x}, \mathbf{y} \sim \mathcal{B}\left(n, \frac{w}{n}\right)$, *and let* $\mathbf{z} = \mathbf{x} \cdot \mathbf{y}$ *as defined in Eq. (2.32). Then*

$$\Pr[z_k = c] = \begin{cases} \sum_{0 \le i \le n, i \text{ odd}} \binom{n}{i} \cdot \left(\left(\frac{w}{n}\right)^2\right)^i \cdot \left(1 - \left(\frac{w}{n}\right)^2\right)^{n-i} & \text{if } c = 1, \\ \sum_{0 \le i \le n, i \text{ even}} \binom{n}{i} \cdot \left(\left(\frac{w}{n}\right)^2\right)^i \cdot \left(1 - \left(\frac{w}{n}\right)^2\right)^{n-i} & \text{if } c = 0, \\ 0 \text{ otherwise.} \end{cases} \tag{6.7}$$

Démonstration. Before going through the proof of this proposition, we need the following two lemmata:

Lemme 5.2. *Consider two independent random variables* $X, Y \sim \mathcal{B}(\frac{w}{n})$. *Then*

$$\Pr[X \cdot Y = c] = \begin{cases} \left(\frac{w}{n}\right)^2 \text{ if } c = 1, \\ (1 - \frac{w}{n})^2 + 2\frac{w}{n}(1 - \frac{w}{n}) = 1 - \left(\frac{w}{n}\right)^2 \text{ if } c = 0, \\ 0 \text{ otherwise.} \end{cases} \tag{6.8}$$

Proof of Lemma 5.2. This Lemma is pretty straightforward and uses basic probability notions, but we include the proof here for completeness. We have that $\Pr[X = 1] = \Pr[Y = 1] = \frac{w}{n}$, and as we consider the product of X and Y, $\Pr[X \cdot Y = 1] = \Pr[\{X = 1\} \cap \{Y = 1\}]$. And because X and Y are independent, $\Pr[X \cdot Y = 1] = \Pr[X = 1] \cdot \Pr[Y = 1] = \left(\frac{w}{n}\right)^2$, and hence $\Pr[X \cdot Y = 0] = 1 - \Pr[X \cdot Y = 1] = 1 - \left(\frac{w}{n}\right)^2$.

(Proof of Lemma 5.2) □

Lemme 5.3. *Let* $S_k = \{(i, j) \in \{0, \dots, n-1\}^2 \text{ such that } i + j \equiv k \mod n\}$ *for* $k \in \{0, \dots, n-1\}$. *Then* $\#S_k = n$.

Proof of Lemma 5.3. Let $\tilde{S}_k = \{(i, j) \in \{0, \dots, n-1\}^2 \text{ such that } i + j = k\}$. As $0 \le i + j \le 2(n-1)$ and $k \in \{0, \dots, n-1\}$, we have that $S_k = \tilde{S}_k \cup \tilde{S}_{n+k}$. Now given $i \in \{0, \dots, n-1\}$, let $\tilde{S}_{k,i} = \{j \in \{0, \dots, n-1\} \text{ such that } i + j = k\}$. Then is it clear that

$$\tilde{S}_k = \dot{\bigcup}_{i=0}^{k} \tilde{S}_{k,i}, \text{ and } \tilde{S}_{n+k} = \dot{\bigcup}_{i=k+1}^{n-1} \tilde{S}_{n+k,j}. \tag{6.9}$$

To conclude, it suffices to notice that both $\tilde{S}_{k,i}$ and $\tilde{S}_{n+k,i}$ contain a single element. Hence,

$$\begin{aligned} \#S_k &= \#\left(\tilde{S}_k \dot{\bigcup} \tilde{S}_{n+k}\right) && (6.10)\\ &= \#\left(\dot{\bigcup}_{i=0}^{k} \tilde{S}_{k,i}\right) + \#\left(\dot{\bigcup}_{i=k}^{n-1} \tilde{S}_{n+k,i}\right) && (6.11)\\ &= \sum_{i=0}^{k} \#\tilde{S}_{k,i} + \sum_{i=k+1}^{n-1} \#\tilde{S}_{n+k,i} && (6.12)\\ &= k+1+n-1-(k+1)+1 = n. && (6.13) \end{aligned}$$

(Proof of Lemma 5.3) □

Now equipped with these two results, and by considering $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ as the results of n successive samples of $X, Y \sim \mathcal{B}(\frac{w}{n})$ and by defining $\mathbf{z} = \mathbf{x} \cdot \mathbf{y} \in \mathbb{F}_2[X]/(X^n - 1)$ as in Eq. (2.32), we have that $\mathbf{z}$ is the result of n successive samples of a new random variable Z. By Lemma 5.3, we have that z_i is the sum of n product of two random variables following $\mathcal{B}(\frac{w}{n})$.

Now by Lemma 5.2, each of the n members of this addition has a probability $\left(\frac{w}{n}\right)^2$ to be equal to one. Therefore $z_i \sim \mathcal{B}(n, \left(\frac{w}{n}\right)^2)$. Finally, as we are working in a finite field of even characteristic, the sum will be equal to 1 only if it has an odd number of members equal to 1 in it, which leads to the claimed probability.

(Proof of Proposition 5.1) □

□

Step 2. We now turn to determining the distribution of the sum of two product vectors. For the sake of readability, let us denote by $\tilde{p} = \tilde{p}(n, w) = \Pr[z_k = 1]$ from Eq. (6.7).

Proposition 5.4. *Let* $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{B}\left(n, \frac{w}{n}\right)$*, and let* $\mathbf{t} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y}$*. Then*

$$\Pr[t_k = c] = \begin{cases} 2\tilde{p}(1-\tilde{p}) \text{ if } c = 1, \\ (1-\tilde{p})^2 + \tilde{p}^2 \text{ if } c = 0, \\ 0 \text{ otherwise.} \end{cases} \quad (6.14)$$

Démonstration. The proof once again involves basic probability facts. Let $\mathbf{a} = \mathbf{x} \cdot \mathbf{r}_2$ and $\mathbf{b} = \mathbf{r}_1 \cdot \mathbf{y}$ so that $\mathbf{t} = \mathbf{a} - \mathbf{b}$. As $t_k \in \mathbb{F}_2$, we have that $t_k = 1$ if and only if $(a_k, b_k) \in \{(0,1),(1,0)\}$. Therefore $\Pr[t_k = 1] = \Pr[\{a_k = 1\} \cap \{b_k = 0\}] + \Pr[\{a_k = 0\} \cap \{b_k = 1\}]$. By Proposition 5.1, $\Pr[a_k = 1] = \Pr[b_k = 1] = \tilde{p}$, *i.e.* a_k and b_k are independent random variables following $\mathcal{B}(\tilde{p})$ (since $\mathbf{x}$, $\mathbf{y}$, $\mathbf{r}_1$, and $\mathbf{r}_2$ are independent). Therefore, $\Pr[t_k = 1] = \Pr[a_k = 1] \cdot \Pr[b_k = 0] + \Pr[a_k = 0] \cdot \Pr[b_k = 1] = 2\tilde{p}(1-\tilde{p})$ as claimed. □ □

Step 3. Finally, by adding the final term $\boldsymbol{\epsilon}$ to $\mathbf{t}$, we obtain the distribution of the error vector $\mathbf{e} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$.

Théorème 6.5.1. *Let* $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{B}\left(n, \frac{w}{n}\right)$*,* $\boldsymbol{\epsilon} \sim \mathcal{B}\left(n, \epsilon\right)$*, and let* $\mathbf{e} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$*. Then*

$$\Pr[e_k = c] = \begin{cases} 2\tilde{p}(1-\tilde{p})(1-\frac{\epsilon}{n}) + ((1-\tilde{p})^2 + \tilde{p}^2)\frac{\epsilon}{n} \text{ if } c = 1, \\ ((1-\tilde{p})^2 + \tilde{p}^2)(1-\frac{\epsilon}{n}) + 2\tilde{p}(1-\tilde{p})\frac{\epsilon}{n} \text{ if } c = 0, \\ 0 \text{ otherwise.} \end{cases} \quad (6.15)$$

Démonstration. We use the notation of Proposition 5.4 and denote $\mathbf{t} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y}$. We have that $t_k \sim \mathcal{B}(2\tilde{p}(1-\tilde{p}))$ and $\epsilon_k \sim \mathcal{B}(\frac{\epsilon}{n})$. Again, as $e_k \in \mathbb{F}_2$, we have that $e_k = 1$ if and only if $(t_k, \epsilon_k) \in \{(0,1),(1,0)\}$. Therefore $\Pr[e_k = 1] = \Pr[\{t_k = 1\} \cap \{\epsilon_k = 0\}] + \Pr[\{t_k = 0\} \cap \{\epsilon_k = 1\}]$. Moreover, as t_k and ϵ_k are independent, $\Pr[e_k = 1] = \Pr[t_k = 1] \cdot \Pr[\epsilon_k = 0] + \Pr[t_k = 0] \cdot \Pr[\epsilon_k = 1] = 2\tilde{p}(1-\tilde{p})(1-\frac{\epsilon}{n}) + ((1-\tilde{p})^2 + \tilde{p}^2)\frac{\epsilon}{n}$ as claimed. □ □

Summary. Theorem 6.5.1 gives us the probability that a coordinate of the error vector $\mathbf{e}$ is 1. We now make the simplifying assumption that for our considered parameters ($w = \mathcal{O}(\sqrt{n})$), the error vector $\mathbf{e} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$ is the result of n successive independent samples of a random variable following a Bernoulli distribution of parameter $p^\star$, where $p^\star$ is defined as in Eq. (6.15): $p^\star = p^\star(n,w) = 2\tilde{p}(1-\tilde{p})(1-\frac{\epsilon}{n}) + ((1-\tilde{p})^2 + \tilde{p}^2)\frac{\epsilon}{n}$ (notice that in practice the results obtained by simulation on the decryption failure are very coherent with this assumption). Under the assumption made above, the Hamming weight of the error vector follows a binomial distribution $\mathcal{B}(n, p^\star)$. Formally, for $0 \leq d \leq \min(2 \cdot w^2 + \epsilon, n)$, we have that:

$$\Pr[\omega(\mathbf{e}) = d] = \binom{n}{d} \cdot (p^\star)^d \cdot (1-p^\star)^{(n-d)}. \tag{6.16}$$

6.6 Decoding Codes with Low Rates and Good Decoding Properties

The previous Section allowed us to determine precisely the distribution of the error vector $\mathbf{e}$ in the configuration where a simple linear code is used. Now the decryption part corresponds to decoding the error described in the previous section. Any decodable code can be used at this point, depending of the considered application, clearly small dimension codes will permit a better decoding, but at the cost of a low encryption rate. This particular case that we consider, corresponds typically to the case of key exchange or authentication, where only a small amount of data needs to be encrypted (typically 80, 128 or 256 bits, a symmetric secret key size). We, hence search for codes with low dimensional rates which are able to correct many errors. Again a tradeoff is necessary between efficient decodable codes but with a high decoding cost and less efficient decodable codes but with a smaller decoding cost.

An example of such family of codes with good decoding properties, which can be analyzed, together with a simple decoding algorithm, is given by Tensor Product Codes, which are used for biometry [BCC+07], where the same type of problematic appears. More specifically we will consider a special simple case of Tensor Product Codes (BCH codes and repetition codes), for which a precise analysis of the decryption can be obtained for Hamming distance.

6.6.1 Tensor Product Codes

Définition 6.44 (Tensor Product Code)**.** *Let $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) be a $[n_1, k_1, d_1]$ (resp. $[n_2, k_2, d_2]$) linear code over $\mathbb{F}$. The* Tensor Product Code *of $\mathcal{C}_1$ and $\mathcal{C}_2$ denoted $\mathcal{C}_1 \otimes \mathcal{C}_2$ is defined as the set of all $n_2 \times n_1$ matrices whose rows are codewords of $\mathcal{C}_1$ and whose columns are codewords of $\mathcal{C}_2$.*

More formally, if $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) is generated by $\mathbf{G}_1$ (resp. $\mathbf{G}_2$), then

$$\mathcal{C}_1 \otimes \mathcal{C}_2 = \left\{ \mathbf{G}_2^\top \mathbf{X} \mathbf{G}_1 \text{ for } \mathbf{X} \in \mathbb{F}^{k_2 \times k_1} \right\} \tag{6.17}$$

Remark 2. *Following the notations of the above Definition, the tensor product of two linear codes is a* $[n_1n_2, k_1k_2, d_1d_2]$ *linear code.*

6.6.2 Specifying the Tensor Product Code

Even if tensor product codes seem well-suited for our purpose, an analysis similar to the one in Sec. 6.5 becomes much more complicated. Therefore, in order to provide strong guarantees on the decryption failure probability for our cryptosystem, we chose to restrict ourselves to a tensor product code $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$, where $\mathcal{C}_1$ is a BCH(n_1, k, δ_1) code of length n_1, dimension k, and correcting capability δ_1 (*i.e.* it can correct up to δ_1 errors), and $\mathcal{C}_2$ is the repetition code of length n_2 and dimension 1, denoted $\mathbb{1}_{n_2}$. (Notice that $\mathbb{1}_{n_2}$ can decode up to $\delta_2 = \lfloor \frac{n_2-1}{2} \rfloor$.) Subsequently, the analysis becomes possible and remains accurate but the negative counterpart is that there probably are some other tensor product codes achieving better efficiency (or smaller key sizes).

In the Hamming metric version of the cryptosystem we propose, a message $\boldsymbol{\mu} \in \mathbb{F}^k$ is first encoded into $\boldsymbol{\mu}_1 \in \mathbb{F}^{n_1}$ with a BCH$(n_1, k_1 = k, \delta_1)$ code, then each coordinate $\mu_{1,i}$ of $\boldsymbol{\mu}_1$ is re-encoded into $\tilde{\boldsymbol{\mu}}_{1,i} \in \mathbb{F}^{n_2}$ with a repetition code $\mathbb{1}_{n_2}$. We denote $n = n_1 n_2$ the length of the tensor product code (its dimension is $k = k_1 \times 1$), and by $\tilde{\boldsymbol{\mu}}$ the resulting encoded vector, *i.e.* $\tilde{\boldsymbol{\mu}} = (\tilde{\boldsymbol{\mu}}_{1,1}, \ldots, \tilde{\boldsymbol{\mu}}_{1,n_1}) \in \mathbb{F}^{n_1 n_2}$.

The efficient algorithm used for the repetition code is the majority decoding, *i.e.* more formally:

$$\mathbb{1}_{n_2}.\mathsf{Decode}(\tilde{\boldsymbol{\mu}}_{1,j}) = \begin{cases} 1 \text{ if } \sum_{i=0}^{n_2-1} \tilde{\mu}_{1,j,i} \geq \lceil \frac{n_2+1}{2} \rceil, \\ 0 \text{ otherwise.} \end{cases} \tag{6.18}$$

Decryption Failure Probability. With a tensor product code $\mathcal{C} = \text{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$ as defined above, a decryption failure occurs whenever the decoding algorithm of the BCH code does not succeed in correcting errors that would have arisen after wrong decodings by the repetition code. Therefore, the analysis of the decryption failure probability is again split into three steps: evaluating the probability that the repetition code does not decode correctly, the the conditional probability of a wrong decoding for the BCH code given an error weight and finally, the decryption failure probability using the law of total probability.

Step 1. We now focus on the probability that an error occurs while decoding the repetition code. As shown in Sec. 6.5, the probability for a coordinate of $\mathbf{e} = \mathbf{x}\mathbf{r}_2 - \mathbf{r}_1\mathbf{y} + \boldsymbol{\epsilon}$ to be 1 is $p^\star = p^\star(n_1n_2, w, \epsilon)$ (see Eq. (6.15)). As mentioned above, $\mathbb{1}_{n_2}$ can decode up to $\delta_2 = \lfloor \frac{n_2-1}{2} \rfloor$ errors. Therefore, assuming that the error vector $\mathbf{e}$ has weight γ (which occurs with the probability given in Eq. (6.16)), the probability of getting a decoding error on a single block of the repetition code $\mathbb{1}_{n_2}$ is hence given by:

$$\bar{p}_\gamma = \bar{p}_\gamma(n_1, n_2) = \sum_{i=\lfloor \frac{n_2-1}{2} \rfloor + 1}^{n_2} \binom{n_2}{i} \left(\frac{\gamma}{n_1 n_2}\right)^i \left(1 - \frac{\gamma}{n_1 n_2}\right)^{n_2 - i}. \tag{6.19}$$

Step 2. We now focus on the BCH(n_1, k, δ_1) code, and recall that it can correct up to δ_1 errors. Now the probability $\mathcal{P}$ that the BCH(n_1, k, δ_1) code fails to decode correctly the encoded message $\boldsymbol{\mu}_1$ back to $\boldsymbol{\mu}$ is given by the probability that an error occurred on at least $\delta_1 + 1$ blocks of the repetition code. Therefore, we have

$$\mathcal{P} = \mathcal{P}(\delta_1, n_1, n_2, \gamma) = \sum_{i=\delta_1+1}^{n_1} \binom{n_1}{i} (\bar{p}_\gamma)^i (1 - \bar{p}_\gamma)^{n_1 - i}. \tag{6.20}$$

Step 3. Finally, using the law of total probability, we have that the decryption failure probability is given by the sum over all the possible weights of the probability that the error has this specific weight times the probability of a decoding error for this weight. This is captured in the following theorem, whose proof is a straightforward consequence of the formulae of Sec. 6.5 and 6.6.1.

Theorem 6.6.1. *Let* $\mathcal{C} = \mathrm{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$, $(pk, sk) \leftarrow \mathsf{KeyGen}$, $\boldsymbol{\mu} \xleftarrow{\$} \mathbb{F}_2^k$, *and some randomness* $\theta \in \{0,1\}^*$, *then with the notations above, the decryption failure probability is*

$$p_{\text{fail}} = \Pr[\mathsf{Decrypt}(\mathrm{sk}, \mathsf{Encrypt}(\mathrm{pk}, \boldsymbol{\mu}, \theta)) \neq \boldsymbol{\mu}.] \tag{6.21}$$

$$= \sum_{\gamma=0}^{\min\left(2w^2+\epsilon, n_1 n_2\right)} \Pr[\omega(\mathbf{e}) = \gamma] \cdot \mathcal{P}(\delta_1, n_1, n_2, \gamma) \tag{6.22}$$

6.7 Parameters

6.7.1 HQC Instantiation for Hamming metric

In this Section, we describe our new cryptosystem in the Hamming metric setting. As mentioned in the previous Section, we use a tensor product code (Def. 6.44) $\mathcal{C} = \mathrm{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$. A message $\boldsymbol{\mu} \in \mathbb{F}^k$ is encoded into $\boldsymbol{\mu}_1 \in \mathbb{F}^{n_1}$ with the BCH code, then each coordinate $\mu_{1,i}$ of $\boldsymbol{\mu}_1$ is encoded into $\tilde{\boldsymbol{\mu}}_{1,i} \in \mathbb{F}^{n_2}$ with $\mathbb{1}_{n_2}$. To match the description of our cryptosystem in Sec. 6.3, we have $\boldsymbol{\mu}\mathbf{G} = \tilde{\boldsymbol{\mu}} = (\tilde{\boldsymbol{\mu}}_{1,1}, \ldots, \tilde{\boldsymbol{\mu}}_{1,n_1}) \in \mathbb{F}^{n_1 n_2}$. To obtain the ciphertext, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} \mathcal{V}^2$ and $\boldsymbol{\epsilon} \xleftarrow{\$} \mathcal{V}$ are generated and the encryption of $\boldsymbol{\mu}$ is $\mathbf{c} = (\mathbf{r}\mathbf{H}^\top, \boldsymbol{\rho} = \boldsymbol{\mu}\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \boldsymbol{\epsilon})$.

Parameters for Our Scheme. We provide two kinds of parameter set depending on the decryption failure probability one is ready to deal with ; the higher this probability, the more efficient the cryptosystem. For each parameter set, the parameters are chosen so that the minimal workfactor of the best known attack exceeds the security parameter. For classical attacks, best known attacks include the works from [CC98, BLP08, FS09, BJMM12] and for quantum attacks, the work of [Ber10]. We consider $w \approx \sqrt{n}$ and follow the complexity described in [TS16].

Finally our cryptosystem is quite efficient since the decryption simply involves a decoding of a repetition code and a small length BCH code.

In Tab. 6.1, n_1 denotes the length of the BCH code, n_2 the length of the repetition code $\mathbb{1}$ so that the length of the tensor product code $\mathcal{C}$ is $n = n_1 n_2$ (actually the smallest primitive prime greater than $n_1 n_2$). k is the dimension of the BCH code and hence also the dimension of $\mathcal{C}$. δ is the decoding capability of the BCH code, *i.e.* the maximum number of errors that the BCH can decode. w is the weight of the n-dimensional vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{r}_1$, and $\mathbf{r}_2$ and similarly $\epsilon = \omega(\boldsymbol{\epsilon}) = 3 \times w$ for our cryptosystem.

Assuming one is willing to trade the decryption failure probability for efficiency, we provide in Tab. 6.2 parameter sets for different security levels in the classical and quantum model of computing. A thoughtful warning should be given at this point regarding the usecase for which we are willing to increase the decryption failure probability: if this does not constitute any flaw in applications such as key exchange, it could be a dramatical flaw for encryption, as shown recently [GJS16].

		Cryptosystem Parameters								
	Instance	n_1	n_2	$n' \approx n_1 n_2 = n$	k'	δ	w	$\epsilon = 3w$	security	p_{fail}
Classical	Toy	255	25	6,379	63	30	36	108	64	$< 2^{-64}$
	Low	255	37	9,437	79	27	45	135	80	$< 2^{-80}$
	Medium	255	53	13,523	99	23	56	168	100	$< 2^{-100}$
	Strong	511	41	20,959	121	58	72	216	128	$< 2^{-128}$
Quantum	Toy	255	65	16,603	63	87	72	216	64	$< 2^{-64}$
	Low	511	47	24,019	76	85	89	267	80	$< 2^{-80}$
	Medium	255	141	35,963	99	23	112	336	100	$< 2^{-100}$
	Strong	511	109	55,711	121	58	143	429	128	$< 2^{-128}$

TABLE 6.1 – Parameter sets for our cryptosystem in Hamming metric. The tensor product code used is $\mathcal{C} = \text{BCH}(n_1, k, \delta) \otimes \mathbb{1}_{n_2}$. The parameters for the BCH codes were taken from [PW72]. Security in the first four instances is given in bits, in the classical model of computing. In the last four instances, the security level is the equivalent of the classical security level but in the quantum computing model, following the work of [Ber10]. The public key size, consisting of $(\mathbf{h}, \mathbf{x} + \mathbf{h} \cdot \mathbf{y})$, has size $2n$ (in bits) (meanwhile considering a seed for h the size can be reduced to n plus the size of the seed), and the secret key (consisting of $\mathbf{x}$ and $\mathbf{y}$ both of weight w) has size $2w \cdot \lceil \log_2(n) \rceil$ (bits) - which again can be reduced to the size of a seed, at last the size of the encrypted message is $n + n' \approx 2n$.

		Cryptosystem Parameters								
	Instance	n_1	n_2	$n' \approx n_1 n_2 = n$	k	δ	w	$\epsilon = 3w$	security	p_{fail}
Classical	Toy	127	43	5,471	64	10	36	108	64	$< 2^{-30}$
	Low	255	31	7,907	79	27	45	135	80	$< 2^{-30}$
	Medium	255	43	10,973	99	23	56	168	100	$< 2^{-30}$
	Strong	255	63	16,603	123	19	72	216	128	$< 2^{-30}$
Quantum	Toy	127	115	14,621	64	10	72	216	64	$< 2^{-30}$
	Low	255	83	21,169	79	27	89	267	80	$< 2^{-30}$
	Medium	255	121	30,859	99	23	112	336	100	$< 2^{-30}$
	Strong	255	181	46,171	123	19	143	429	128	$< 2^{-30}$

TABLE 6.2 – Parameter sets for our cryptosystem in Hamming metric for a decryption failure probability of at most 2^{-30}. All the parameters are similar to Tab. 6.1, including the key sizes.

Auditability of the Results. Parameter sets for the Hamming metric version of the cryptosystem we propose and especially the decryption failure probability in Tab. 6.1 and 6.2 were determined using the formula of Eq. (6.22). In order for the reader to check the veracity of the statements claimed above and below in those tables, we provide in the auxiliary supporting material a Sage [Dev16] implementation to compute the aforementioned probabilities. As a disclaimer, the authors would like to emphasize that the implementation is not intended to performance purposes, and we would like to warn the reader that the computations can take a while for some parameters (several minutes).

Computational Cost. The most costly part of the encryption and decryption is the matrix vector product, in practice the complexity is hence $\mathcal{O}(n^{\frac{3}{2}})$ (for $w \approx \sqrt{n}$). Asymptotically the cost becomes linear in n.

Notice that it would be possible to consider other type of decodable code in order to increase the encryption rate to 1/4 (say), but at the cost of an increasing of the length of the code, for instance using LDPC (3,6) codes would increase the rate, but multiply the length by a factor roughly three.

6.7.2 RQC Instantiation for rank metric

The decryption algorithm of our cryptosystem asks to decode an error $\mathbf{e} = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \boldsymbol{\epsilon}$ where the words $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{r}_1, \mathbf{r}_2)$ have rank weight w. At the difference of Hamming metric the rank weight of the vector $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y}$ is almost always w^2 and is in any case bounded above by w^2. In particular with a strong probability the rank weight of $\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y}$ is the same than the rank weight of $\mathbf{x} \cdot \mathbf{r}_2$ since $\mathbf{x}$ and $\mathbf{y}$ share the same rank support, so as $\mathbf{r}_1$ and $\mathbf{r}_2$.

Parameters for Our Scheme. In Tab. 6.3, n denotes the length of the Rank metric code, k' its dimension, q is the number of elements in the base field $\mathbb{F}_q$, and m is the degree of the extension. Similarly to the Hamming instantiation, w is the rank weight of vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{r}_1$, and $\mathbf{r}_2$, and ϵ the rank weight of $\boldsymbol{\epsilon}$. For the case of rank metric, we always consider $n' = n = m$.

Specific structural attacks. Specific attacks were described in [HT15, GRSZ14] for LRPC cyclic codes. These attack use the fact that the targeted code has a generator matrix formed from shifted low weight codewords and in the case of [HT15], also uses multi-factor factorization of $x^n - 1$. These attack corresponds to searching for low weight codewords of a given code of rate 1/2. In the present case the attacker has to search for a low weight word associated to a non null syndrom, such that previous attacks imply considering a code with a larger dimension so that in practice these attacks do no improve on direct attacks on the syndrome. Meanwhile in practice by default, we choose n a primitive prime number, such that the polynomial $x^n - 1$ has no factor of degree less than $\frac{n-1}{2}$ except $x - 1$. The best attacks consists in decoding a random double-circulant $[2n, n]$ over $\mathbb{F}_{q^m}$ for rank weight ω. For decoding, we consider Gabidulin $[n', k']$ codes over $\mathbb{F}_{q^{n'}}$, which can decode $\frac{n'-k'}{2}$ rank errors and choose our parameters such that $w^2 + \epsilon \leq \frac{n'-k'}{2}$, so that at the difference of Hamming metric there is no decryption failure. Examples of parameters are given in Tab. 6.3 according to best known attacks (combinatorial attacks in practice) described in Sec. 2. The table also gives the security for quantum attacks. For the case of rank metric, we always consider $n' = n = m$.

	Cryptosystem Parameters									
Instance	n	k'	m	q	w	ϵ	plaintext	key size	security	EQS*
RQC-I	37	13	37	4	3	3	962	2,738	90	45
RQC-II	53	13	53	2	4	4	689	2,809	95	47
RQC-III	61	3	61	2	5	4	183	3,721	140	70
RQC-IV	83	3	83	2	6	4	249	6,889	230	115
RQC-V	61	3	61	4	5	4	366	7,442	264	132

TABLE 6.3 – Parameter sets for RQC: our cryptosystem in Rank metric. * EQS stands for Equivalent Quantum Security, and refers to the results of [GHT16]. The plaintexts, key sizes, and security are expressed in bits.

Remark. The system is based on cyclic codes, which means considering polynomials modulo $x^n - 1$, interestingly enough, and only in the case of the rank metric, the construction remains valid when considering not only polynomials modulo $x^n - 1$ but also modulo a polynomial with coefficient in the base field $GF(q)$. Indeed in that case the modulo does not change the rank weight of a codeword. Such a variation on the scheme may have an interest to avoid potential structural attack which may use the factorization of the quotient polynomial for the considerd polynomial ring.

Computational Cost. The encryption cost corresponds to a matrix-vector product over $\mathbb{F}_{q^m}$, for a multiplication cost of elements of $\mathbb{F}_{q^m}$ in $m\log(m)\log(\log(m))$, we obtain an encryption complexity in $\mathcal{O}\left(n^2 m \log\left(m\right)\log\left(\log\left(m\right)\right)\right)$. The decryption cost is also a matrix-vector multiplication plus the decoding cost of the Gabidulin codes, both have the complexities in $\mathcal{O}\left(n^2 m \log\left(m\right)\log\left(\log\left(m\right)\right)\right)$.

6.7.3 Comparison with Other Cryptosystems

In the following we consider the different type of code-based cryptosystems and express different parameters of the different systems in terms of the security parameters λ, considering best known attacks of complexity $2^{\mathcal{O}(w)}$ for decoding a word of weight w for Hamming distance and complexity in $2^{\mathcal{O}(wn)}$ for decoding a word of rank weight w for a code of double-circulant code of length $2n$ for rank metric. McEliece-Goppa corresponds to the original scheme proposed by McEliece [McE78] of dimension rate $\frac{1}{2}$.

Table 6.4 shows that even if the recent cryptosystem MDPC has a smaller public key and a weaker hidden structure than the McEliece cryptosystem, the size of the ciphertext remains non negligible. The HQC benefits from the same type of parameters than the MDPC systems but with no hidden structure at the cost of a smaller encryption rate. At last the table shows the very strong potential of rank metric based cryptosystems, whose parameters remain rather low compared to MDPC and HQC cryptosystems.

6.8 Conclusion and Future Work

In this work, we presented a new efficient approach for constructing code-based cryptosystems. This approach shares the same spirit as Alekhnovich's blueprint [Ale03] on random matrices. The main advantage of our new cryptosystem over previous efficient constructions is that there are no hidden trapdoor in the public matrix. Our construction is generic enough so

Cryptosystem		Code Length	Public Key Size	Ciphertext Size	Hidden Structure	Cyclic Structure
Goppa-McEliece	[McE78]	$\mathcal{O}(\lambda \log \lambda)$	$\mathcal{O}\left(\lambda^2 (\log \lambda)^2\right)$	$\mathcal{O}(\lambda \log \lambda)$	Strong	No
MDPC	[MTSB13]	$\mathcal{O}(\lambda^2)$	$\mathcal{O}(\lambda^2)$	$\mathcal{O}(\lambda^2)$	Weak	Yes
LRPC	[GMRZ13]	$\mathcal{O}\left(\lambda^{\frac{2}{3}}\right)$	$\mathcal{O}\left(\lambda^{\frac{4}{3}}\right)$	$\mathcal{O}\left(\lambda^{\frac{4}{3}}\right)$	Weak	Yes
HQC	[Sec. 6.7.1]	$\mathcal{O}(\lambda^2)$	$\mathcal{O}(\lambda^2)$	$\mathcal{O}(\lambda^2)$	No	Yes
RQC	[Sec. 6.7.2]	$\mathcal{O}\left(\lambda^{\frac{2}{3}}\right)$	$\mathcal{O}\left(\lambda^{\frac{4}{3}}\right)$	$\mathcal{O}\left(\lambda^{\frac{4}{3}}\right)$	No	Yes

TABLE 6.4 – Parameters comparison for different code-based cryptosystems with respect to the security parameter λ

that we provide two instantiations of our cryptosystem: one for the Hamming metric (HQC), and one for the Rank metric (RQC). Both constructions are pretty efficient and compare favourably to previous works, especially for the rank metric setting. Additionally, we provide for the Hamming setting a detailed analysis of the error term yielding a concrete, precise and easy-to-verify decryption failure.

This analysis was facilitated by the shape of the tensor product code, and more complex-to-analyze tensor product codes might yield slightly shorter keys and more efficiency.

However, for such a tensor product code the analysis of the decryption failure probability becomes much more tricky, and finding suitable upper bounds for it will be part of a future work.

Quatrième partie

Conclusions et Perspectives

Chapitre 7

Conclusions et Perspectives

7.1 Conclusions Générales

La science du secret a bien évolué depuis ses débuts, passant progressivement d'un art de protéger l'information à une discipline visant à apporter de la confiance dans les échanges, numériques pour la plupart. Le siècle qui vient de s'écouler a vu se développer l'intégralité de la cryptographie que nous qualifions de moderne, avec notamment des algorithmes symétriques en lesquels nous avons dorénavant une confiance quasi aveugle, ainsi que des algorithmes asymétriques qui ont permis de déverrouiller de nouvelles fonctionnalités utilisées aujourd'hui quotidiennement sur Internet. Malheureusement, ce siècle se conclut par une nouvelle dramatique annoncée par Peter Shor : l'arrivée d'un nouveau modèle de calcul, capable de mettre à terre les paramètres des cryptosystèmes existants et à terme, impactant de manière significative les cryptosystèmes basés sur la théorie des nombres.

Ce nouveau millénaire a donc commencé par un mouvement de foule vers le développement de primitives cryptographiques *a priori* résistantes à ce nouveau modèle de calcul quantique. Fait rassurant, les outils mathématiques candidats à cette cryptographie post-quantique sont nombreux, et variés, et des travaux récents ont permis de mieux comprendre leur complexité ainsi que les fonctionnalités qu'ils apportaient. Dans un soucis d'uniformisation, le NIST a lancé un appel à standardisation des primitives cryptographiques les plus utilisées aujourd'hui à savoir : la signature, le chiffrement, et l'échange de clés. Les soumissions pour cet appel à projet sont attendues pour novembre 2017.

Dans ce contexte, nous avons travaillé au cours de cette thèse (entre autres) à la réparation d'une primitive de signature basée sur les réseaux. Cette réparation perdait en efficacité par rapport au schéma initial non sûr, mais la réparation de ce schéma n'était pas si coûteuse. Ainsi, les techniques dont nous nous sommes servi conduisent à de meilleurs paramètres que le papier d'où elles proviennent. Cependant, d'autres avancées ont été réalisées entre temps en termes de signature basée sur les réseaux et les schémas existants aujourd'hui surpassent notre proposition.

En utilisant cette signature, nous avons également proposé un modèle de signature plus exotique, appelé signature traçable. Les signatures traçables sont une amélioration des signatures de groupe à la fois en termes de vie privée et d'efficacité. Notre proposition permet en plus de garantir aux utilisateurs du groupe une propriété essentielle pour la responsabilité des utilisateurs du groupe : la non-framability. Cette dernière garantie que

même un administrateur de groupe corrompu ne pourra abuser les utilisateurs du groupe. Malheureusement, le problème auquel nous avons pu nous ramener décrit relativement mal la configuration du schéma, et il est raisonnable de penser que ce schéma ne saurait être efficace en pratique. Son objectif premier était principalement de montrer qu'il est possible d'atteindre la non-framability.

Pour finir avec les réseaux, nous avons travaillé sur l'élaboration d'un protocole efficace de délégation de signature, utilisant du chiffrement complètement homomorphe. Longtemps présenté comme le Saint Graal de la cryptographie, le chiffrement complètement homomorphe – qui permet à n'importe qui d'effectuer des opérations sur les chiffrés – a connu des débuts difficiles notamment à cause de ses performances dissuasives. Avec les récentes améliorations, nous avons proposé un protocole où l'utilisation d'un tel chiffrement permet – de manière étonnante – d'*accélérer* les calculs. Bien que ce schéma souffre d'un modèle de sécurité plus faible, il reste intéressant de voir que l'utilisation du chiffrement complètement homomorphe peut, non pas seulement apporter de nouvelles fonctionnalités, mais aussi améliorer les performances de systèmes établis.

Au cours de cette thèse, nous avons également proposé une nouvelle approche pour le chiffrement. Nos travaux ont été motivés par une hypothèse relativement forte concernant les cryptosystèmes basés sur la théorie des codes correcteurs d'erreurs, à savoir que la structure du code utilisée doit être suffisamment bien masquée pour que le schéma reste sûr. Nous avons donc proposé une méthode pour construire un cryptosystème qui ne dépend ni de la métrique utilisée, ni des familles de codes utilisés, et qui permet de se soustraire à cette hypothèse contraignante. Nous avons instancié ce schéma à la fois en métrique rang et en métrique de Hamming, et avons obtenu des paramètres tout à fait raisonnables. Ce schéma fera par ailleurs l'objet de plusieurs soumissions en réponse à l'appel à projet du NIST.

À travers ces différents travaux, nous avons contribué au bon développement de la cryptographie dite "quantum-resistant". Cette dernière est réellement en train d'émerger, certaines primitives sont d'ores et déjà en phase de test, et des standards émergerons très prochainement. Si certains aspects de la cryptographie post-quantique sont mieux compris, il reste cependant du chemin à accomplir avant que celle-ci ne soit complètement opérationnelle. Nous développons certains de ces aspects dans la prochaine section.

7.2 Perspectives

7.2.1 Réseaux Euclidiens

Les problèmes sur les réseaux permettant des applications cryptographiques avec des réductions pire-cas cas-moyen sont aujourd'hui bien établis et étudiés. Cependant, il est parfois difficile de se ramener à une version exacte de ces problèmes. Pour le cas de notre signature traçable, une réduction vers un autre problème aurait semblé plus naturel. En effet, les paramètres de ce schéma sont très fortement impactés par ceux des réductions au problème k-SIS de [BF11] et de [LPSS14]. Une réduction vers un problème différent où l'adversaire doit trouver un nouveau vecteur de poids égal (au lieu de à un facteur près) n'appartenant pas au span pourrait conduire à des paramètres plus fins. Mais il resterait à étudier la difficulté même de ce nouveau problème ainsi que ses relations avec les problèmes

existants. Un tel problème a récemment été formulé en code [ABCG16], et il n'est pas exclus que nous étudions ce problème à l'avenir.

Un autre inconvénient majeur de la mise en pratique de la cryptographie basée sur les réseaux est que seulement peu de primitives cryptographiques proposent des paramètres concrets. La cryptographie ne peut se développer sans cryptanalyse et inversement. Bien que de plus en plus d'efforts soient faits dans ce sens, toute proposition vouée à être utilisée en pratique devrait être accompagnée de paramètres concrets et non asymptotiques. D'autant plus que nos connaissances en matière de cryptanalyse sur les réseaux restent limitées. Dans cette optique, nous avons également travaillé à comprendre l'efficacité pratique de différents algorithmes de réductions de réseaux, notamment dans le cas des réseaux idéaux. Ces travaux nous ont permis de résoudre certains des challenges proposés par l'université de Darmstadt [1]. Nous souhaitons à terme mener de plus amples expériences afin de mieux comprendre comment paramétrer les primitives basées sur les réseaux.

7.2.2 Codes Correcteur d'Erreurs

Concernant les codes correcteurs d'erreurs en métrique de Hamming, nous avons étudié différentes configurations pour le schéma que nous avons proposé en chapitre 6, et retenu le cas d'un code produit BCH-répétition. Si l'algorithme de décodage obtenu permet de décoder efficacement un bon nombre d'erreurs, il n'est pas exclus que d'autres combinaisons puissent soit mieux décoder, auquel cas notre cryptosystème deviendrait plus performant, soit décoder plus d'erreurs, auquel cas les paramètres pourraient être améliorer. Une étude plus poussée concernant ces différentes combinaisons est envisagée en vue de la soumission au NIST.

Concernant la métrique rang, les paramètres obtenus dans ce même chapitre sont déjà satisfaisant et à notre humble avis, les efforts devraient plutôt être concentrés sur la cryptanalyse des instances atteignables, afin de s'assurer que la sécurité affirmée tient en pratique. L'implémentation et la cryptanalyse de ce schéma feront partie d'un prochain travail.

1. Les challenges sont disponibles à cette ici.

Références

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005. *Cité Section 4.4.1*

[ABCG15] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. In *The 9th International Workshop on Coding and Cryptography 2015.* `https://hal.inria.fr/hal-01276464`, *full version available on* `eprint.iacr.org`, WCC 2015. *Cité Sections 4.1, 4.1, 4.3, et 4.5*

[ABCG16] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A practical group signature scheme based on rank metric. In *Waifi 2016*, July 2016. *Cité Section 7.2.1*

[ABDG14] Carlos Aguilar Melchor, Xavier Boyen, Jean-Christophe Deneuville, and Philippe Gaborit. Sealing the leak on classical ntru signatures. In *PQCrypto'14*, pages 1–21, 2014. *Cité Sections 4.1, 4.1, 4.3, 4.4.2, 10, 1, 2, et 4.4.4*

[ABG+16] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. NFLlib: NTT-based fast lattice library. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 341–356. Springer, Heidelberg, February / March 2016. *Cité Section 5.1.2*

[AFF+13] Carlos Aguilar Melchor, Simon Fau, Caroline Fontaine, Guy Gogniat, and Renaud Sirdey. Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *Signal Processing Magazine, IEEE*, 30(2):108–117, 2013. *Cité Section 5.1.2*

[AIK07] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In Menezes [Men07], pages 92–110. *Cité Sections 2.3.2, 2.3.2, et 2.3.2*

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. *Cité Sections 1.4.1, 2.2.2, 3.1, et 3.4*

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd ACM STOC*, pages 601–610. ACM Press, July 2001. *Cité Section 2.2.3.3*

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004. *Cité Sections 1.3.1 et 1.3.2*

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003. *Cité Sections 6.1.1 et 6.8*

[Bab85] László Babai. On lovász' lattice reduction and the nearest lattice point problem (shortened version). In *Proceedings of the 2nd Symposium of Theoretical Aspects of Computer Science*, STACS '85, pages 13–20, London, UK, UK, 1985. Springer-Verlag. *Cité Section 3.3*

[BCC+07] Julien Bringer, Hervé Chabanne, Gérard Cohen, Bruno Kindarji, and Gilles Zémor. Optimal iris fuzzy sketches. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*, pages 1–6. IEEE, 2007. *Cité Section 6.6*

[BCGO09] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. Reducing key length of the McEliece cryptosystem. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 77–97. Springer, Heidelberg, June 2009. *Cité Section 6.1.1*

[BCJR15] Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff, editors. *FC 2015 Workshops*, volume 8976 of *LNCS*. Springer, Heidelberg, January 2015. *Cité Section 7.2.2*

[Ber10] Daniel J Bernstein. Grover vs. mceliece. In *Post-Quantum Cryptography*, pages 73–80. Springer, 2010. *Cité Sections 6.7.1 et 6.1*

[BF11] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 1–16, 2011. *Cité Sections 2.2.2.2, 3, 4, 4.1, 4.1, 4.2, 4.3, 4.4.4, 4.C, et 7.2.1*

[BGH+13] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. Private database queries using somewhat homomorphic encryption. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 102–118. Springer, Heidelberg, June 2013. *Cité Section 5.1.2*

[BGMW93] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David Bruce Wilson. Fast exponentiation with precomputation (extended abstract). In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 200–207. Springer, Heidelberg, May 1993. *Cité Section 5.1.2*

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. *Cité Sections 2.2.2.2, 5.1, 5.2.2, 5.2.2.3, et 5.5.1*

[BI04] Jean-Claude Bajard and Laurent Imbert. A full rns implementation of rsa. *Computers, IEEE Transactions on*, 53(6):769–774, 2004. *Cité Section 5.1.1.1*

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In Pointcheval and Johansson [PJ12], pages 520–536. *Cité Sections 2.3.3 et 6.7.1*

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 3–24. Springer, Heidelberg, November / December 2015. *Cité Sections 2.1.2 et 4.3*

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. Cryptology ePrint Archive, Report 2013/075, 2013. http://eprint.iacr.org/. *Cité Section 2.1.4.2*

[BLN14] Joppe W Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014. *Cité Section 5.1.2*

[BLP08] Daniel J Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. In *Post-Quantum Cryptography*, pages 31–46. Springer, 2008. *Cité Section 6.7.1*

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. *Cité Section 2.2.2.2*

[BMvT78] Elwyn R Berlekamp, Robert J McEliece, and Henk CA van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. *Cité Section 2.3.2*

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003. *Cité Sections 2.1.4.4 et 4.1*

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM. *Cité Section 3.A.2*

[BP12] Olivier Blazy and David Pointcheval. Traceable signature with stepping capabilities. In David Naccache, editor, *Quisquater Festschrift*, Lecture Notes in Computer Science. Springer, 2012. Full version available from the web page of the authors. *Cité Section 4.1*

[BPV98] Victor Boyko, Marcus Peinado, and Ramarathnam Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 221–235. Springer, Heidelberg, May / June 1998. *Cité Section 5.1.2*

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005. *Cité Sections 2.1.4.4 et 4.1*

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014. *Cité Section 5.1.1.1*

[CC98] Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum weight words in a linear code: application to mceliece cryptosystem and to narrow-sense bch codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998. *Cité Section 6.7.1*

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335. Springer, Heidelberg, May 2013. *Cité Section 5.2.2.3*

[CKK15] Jung Hee Cheon, Miran Kim, and Myungsun Kim. Search-and-compute on encrypted data. In Brenner et al. [BCJR15], pages 142–159. *Cité Section 5.1.2*

[CKL15] Jung Hee Cheon, Miran Kim, and Kristin E. Lauter. Homomorphic computation of edit distance. In Brenner et al. [BCJR15], pages 194–212. *Cité Section 5.1.2*

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011. *Cité Section 2.2.3.4*

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Davies [Dav91], pages 257–265. *Cité Sections 2.1.4.4, 4.1, et 4.4.1*

[Dav91] Donald W. Davies, editor. *EUROCRYPT'91*, volume 547 of *LNCS*. Springer, Heidelberg, April 1991. *Cité Section 7.2.2*

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013. *Cité Sections 1.5, 3.1, 2, et 3.5.3*

[de 95] Peter de Rooij. Efficient exponentiation using procomputation and vector addition chains. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 389–399. Springer, Heidelberg, May 1995. *Cité Section 5.1.2*

[Den12] Jean-Christophe Deneuville. Cryptographie homomorphe. Master's thesis, Université de Limoges, September 2012. *Cité Section 2.1.4.2*

[Dev16] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.1)*, 2016. `http://www.sagemath.org`. *Cité Section 6.7.1*

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976. *Cité Sections 1.2.2 et 1.2.3*

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Heidelberg, December 2014. *Cité Sections 2.1.2 et 4.1*

[DN12] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 433–450. Springer, Heidelberg, December 2012. *Cité Sections 1.5, 3, 3.1, 3.3, 3.5.3, et 3.5.3*

[DP12] Ivan Damgård and Sunoo Park. Is public-key encryption based on lpn practical ? *IACR Cryptology ePrint Archive*, 2012:699, 2012. *Cité Sections 6.1.1 et 6.1.3*

[DV13] Alexandre Duc and Serge Vaudenay. Helen: a public-key cryptosystem based on the lpn and the decisional minimal distance problems. In *International Conference on Cryptology in Africa*, pages 107–126. Springer, 2013. *Cité Section 6.1.1*

[Fei86] Joan Feigenbaum. Encrypting problem instances: Or ..., can you take advantage of someone without having to trust him ? In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 477–488. Springer, Heidelberg, August 1986. *Cité Section 5.1.2*

[Fel08] Willliam Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008. *Cité Section 2.1.2*

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 844–855. ACM Press, November 2014. *Cité Section 5.1.2*

[FLDVP08] Jean-Charles Faugere, Françoise Levy-Dit-Vehel, and Ludovic Perret. Cryptanalysis of minrank. In *Annual International Cryptology Conference*, pages 280–296. Springer, 2008. *Cité Section 2.3.3*

[FOPT10] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In Gilbert [Gil10], pages 279–298. *Cité Section 6.1.1*

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. *Cité Section 4.3*

[FS09] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, Heidelberg, December 2009. *Cité Section 6.7.1*

[Gab85] Ernest Mukhamedovich Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985. *Cité Section 2.3.1*

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, 2005. *Cité Section 2.3.1*

[Gab12] Ernst Gabidulin. A brief survey of metrics in coding theory. *Mathematics of Distances and Applications*, 66, 2012. *Cité Section 2.3.1*

[Gen09a] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729. *Cité Section 2.1.4.2*

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. *Cité Sections 5.1 et 5.2.2*

[GG07] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *2007 IEEE International Symposium on Information Theory*, pages 191–195. IEEE, 2007. *Cité Section 2.3.1*

[GG14] Juan A. Garay and Rosario Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*. Springer, Heidelberg, August 2014. *Cité Section 7.2.2*

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK, 1997. Springer-Verlag. *Cité Sections 3 et 3.1*

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012. *Cité Sections 5.2.2 et 5.2.2.4*

[GHT16] Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. Ranksynd a PRNG based on rank metric. In *2016*, pages 18–28, Fukuoka, Japan, February 2016. *Cité Section 6.3*

[Gil52] Edgar N Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31(3):504–522, 1952. *Cité Section 3.32*

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, Heidelberg, May 2010. *Cité Section 7.2.2*

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on mdpc with cca security using decoding errors. In *22nd Annual International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT), 2016*, 2016. *Cité Sections 1.5, 2.3.4.2, et 6.7.1*

[GKV10] S Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *Advances in Cryptology-ASIACRYPT 2010*, pages 395–412. Springer, 2010. *Cité Section 4.1*

[GLN13] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC 12*, volume 7839 of *LNCS*, pages 1–21. Springer, Heidelberg, November 2013. *Cité Section 5.1.2*

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. *Cité Section 1.6*

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988. *Cité Section 4.3*

[GMRZ13] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC'2013*, Bergen, Norway, 2013. Available on www.selmer.uib.no/WCC2013/pdfs/Gaborit.pdf. *Cité Sections 2.3.4, 6.1.1, 6.1.3, et 6.7.3*

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999. *Cité Section 2.2.2.1*

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008. *Cité Section 3.5.3*

[GPT91] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and thier applications in cryptology. In Davies [Dav91], pages 482–489. *Cité Section 2.3.3*

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. *Cité Sections 3, 3.1.1, et 3.5.3*

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996. *Cité Section 1.3.4*

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Heidelberg, December 2007. *Cité Section 4.1*

[GRS16] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Transactions on Information Theory*, 62(2):1006–1019, 2016. *Cité Section 2.3.3*

[GRSZ14] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In Pointcheval and Vergnaud [PV14], pages 1–12. *Cité Section 6.7.2*

[GZ14] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *CoRR*, abs/1404.3482, 2014. *Cité Sections 2.3.2 et 2.3.2*

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Menezes [Men07], pages 150–169. *Cité Section 3.5.3*

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Heidelberg, April 2003. *Cité Sections 1.5, 3, 3.1, 3.3, 3.3, 6, 4, 4, et 3.5.3*

[HHgP+05] Jeff Hoffstein, Nicholas Howgrave-graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Performance Improvements and a Baseline Parameter Generation Algorithm for NTRUSign. In *In Proc. of Workshop on Mathematical Problems and Techniques in Cryptology*, pages 99–126, 2005. *Cité Sections 3.3, 6, 3.5.1, et 3.5.3*

[HHGPW09] Jeff Hoffstein, Nick Howgrave-Graham, Jill Pipher, and William Whyte. Practical lattice-based cryptography: Ntruencrypt and ntrusign. In *The LLL Algorithm*, pages 349–390. Springer, 2009. *Cité Sections 3.1, 3.3, et 3.5.3*

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. *Cité Section 2.1.2*

[HKL+12] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-lpn. In *Fast Software Encryption*, pages 346–365. Springer, 2012. *Cité Section 6.1.1*

[HL05] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 264–282. Springer, Heidelberg, February 2005. *Cité Section 5.1.2*

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003. *Cité Sections 5.2.1 et 5.4.4*

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS-III*, volume 1423 of *LNCS*, pages 267–288. Springer, June 1998. *Cité Sections 5.1.1.2 et 6.1.1*

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 211–228. Springer, Heidelberg, May 2001. *Cité Sections 1.5 et 3.1*

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Garay and Gennaro [GG14], pages 554–571. *Cité Sections 2.1.4.2, 5.1.1.1, 5.1.1.2, 5.1.2, 5.2.2, 5.2.2.4, et 5.6*

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670. Springer, Heidelberg, April 2015. *Cité Sections 5.1.1.1, 5.1.1.2, 5.1.2, 5.2.2, 5.2.2.4, et 5.6*

[HT15] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the lrpc cryptosystem. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2747–2751. IEEE, 2015. *Cité Sections 2.3.2 et 6.7.2*

[HWH08] Yupu Hu, Baocang Wang, and Wencai He. Ntrusign with a new perturbation. *IEEE Trans. Inf. Theor.*, 54(7):3216–3221, July 2008. *Cité Sections 1.5, 3.1, 3.3, et 3.5.3*

[IN10] Johan Ivarsson and Andreas Nilsson. A review of hardware security modules fall 2010. Technical report, Certezza, 2010. Available at http://www.opendnssec.org/wp-content/uploads/2011/01/A-Review-of-Hardware-Security-Modules-Fall-2010.pdf. *Cité Section 5.1*

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, Heidelberg, February 2002. *Cité Section 5.1.2*

[Kat15] Jonathan Katz, editor. *PKC 2015*, volume 9020 of *LNCS*. Springer, Heidelberg, March / April 2015. *Cité Section 7.2.2*

[KMP14] Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In Krawczyk [Kra14], pages 1–18. *Cité Section 6.1.1*

[Knu97] Donald Knuth. The art of computer programming: Semi-numerical algorithms, volume vol. 2, 1997. *Cité Section 5.1.2*

[Knu98] Donald Ervin Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1998. *Cité Section I*

[Kra14] Hugo Krawczyk, editor. *PKC 2014*, volume 8383 of *LNCS*. Springer, Heidelberg, March 2014. *Cité Section 7.2.2*

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, Heidelberg, May 2004. *Cité Sections 2.1.4.4, 4.1, 4.1, 4.4.1, et 4.4.1*

[KY06] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *Int. J. Secur. Netw.*, 1(1/2):24–45, September 2006. *Cité Section 4.1*

[Laa15] Thijs Laarhoven. *Search problems in cryptography*. PhD thesis, PhD thesis, Eindhoven University of Technology, 2015. http://www. thijs. com/docs/phd-final. pdf. 8, 2015. *Cité Section 4*

[LdVP06] Françoise Levy-dit Vehel and L Perret. Algebraic decoding of rank metric codes. *Proceedings of YACC*, 2006. *Cité Section 2.3.3*

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 1–17. Springer, Heidelberg, August 2013. *Cité Section 5.5.3*

[LL94] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 95–107. Springer, Heidelberg, August 1994. *Cité Section 5.1.2*

[LLL82] Arjen K. Lenstra, Handrik W. jr. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. *Cité Section 2.2.3.2*

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 41–61. Springer, Heidelberg, December 2013. *Cité Sections 1.5 et 4.1*

[LLM+16] Benoît Libert, S Ling, F Mouhartem, K Nguyen, and H Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. Technical report, Cryptology ePrint Archive: Report 2016/101 (to appear AsiaCrypt'16), 2016. *Cité Section 1.5*

[LLN15] Kristin E. Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 3–27. Springer, Heidelberg, September 2015. *Cité Section 5.1.2*

[LLNW14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In Krawczyk [Kra14], pages 345–361. *Cité Sections 1.5 et 4.1*

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *Eurocrypt 2016*, volume 9666. Springer, 2016. *Cité Section 4.1*

[LN09] Gaëtan Leurent and Phong Q. Nguyen. How risky is the random-oracle model ? In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 445–464. Springer, Heidelberg, August 2009. *Cité Section 2.1.3*

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In Pointcheval and Vergnaud [PV14], pages 318–335. *Cité Section 5.1.2*

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2013. *Cité Sections 4.1, 4.3, 4.4.2, et 4.4.4*

[LNW15] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In Katz [Kat15], pages 427–449. *Cité Sections 1.5, 4.1, et 4.1*

[Loi06] Pierre Loidreau. Properties of codes in rank metric. *arXiv preprint cs/0610057*, 2006. *Cité Sections 2.3.1 et 2.3.1*

[LP11] Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for LWE-based Encryption. In *Proceedings of the 11th international conference on Topics in cryptology: CT-RSA 2011*, CT-RSA'11, pages 319–339, Berlin, Heidelberg, 2011. Springer-Verlag. *Cité Sections 2.2.3.4 et 3.5.3*

[LPR10a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010. *Cité Section 2.2.2.2*

[LPR10b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23. *Cité Section 6.1.1*

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Garay and Gennaro [GG14], pages 315–334. *Cité Sections 4.1, 4.2, 4.2, 4.3, 4.4.2, 4.4.4, 4.C, et 7.2.1*

[LY09] Benoît Libert and Moti Yung. Efficient traceable signatures in the standard model. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 187–205. Springer, Heidelberg, August 2009. *Cité Sections 4.1, 4.4.1, et 4.5*

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In Pointcheval and Johansson [PJ12], pages 738–755. *Cité Sections 1.5, 1.1, 3, 3.1.1, 1, 2.1, 2.2, 3.2, 2.3, 3.2.1, 3.4, 3, 5, 3.5.2, 3.5.3, 3.5.3, 3.5.3, 3.5.1, 5.1, 5.2, 3.5.3, 3.A.1, 3.A.2, 4.1, 4.3, et 3*

[MB09] Rafael Misoczki and Paulo S. L. M. Barreto. Compact McEliece keys from goppa codes. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 376–392. Springer, Heidelberg, August 2009. *Cité Section 6.1.1*

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, January 1978. *Cité Sections 2.3.4, 6.7.3, et 6.7.3*

[Men07] Alfred Menezes, editor. *CRYPTO 2007*, volume 4622 of *LNCS*. Springer, Heidelberg, August 2007. *Cité Section 7.2.2*

[MKI90] Tsutomu Matsumoto, Koki Kato, and Hideki Imai. Speeding up secret computations with insecure auxiliary devices. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 497–506. Springer, Heidelberg, August 1990. *Cité Section 5.1.2*

[Mos14] Michele Mosca, editor. *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*. Springer, 2014. *Cité Section 3*

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012. *Cité Section 4.1*

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. *Cité Section 4.3*

[MR09] Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, 2009. *Cité Sections 2.2.2 et 3.5.3*

[MS01] Alexander May and Joseph H. Silverman. Dimension reduction methods for convolution modular lattices. In Joseph H. Silverman, editor, *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2001. *Cité Section 3.5.3*

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo SLM Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 2069–2073. IEEE, 2013. *Cité Sections 3.29, 2.3.1, 2.3.4, 6.1.1, 6.1.3, et 6.7.3*

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *ACM CCS 96*, pages 48–57. ACM Press, March 1996. *Cité Section 5.1.2*

[Mus09] Jöelle Musset. Sécurité informatique: Ethical hacking: Apprendre l'attaque pour mieux se défendre, 2009. *Cité Section I*

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996. *Cité Section I*

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory - Problemy Upravleniya I Teorii Informatsii*, 15(2):159–166, 1986. *Cité Section 2.3.4.1*

[NIS15] NIST. Recommendation for key management: Part 1: General (revision 4), September 2015. SP 800-57 Part 1-Rev. 4. http://csrc.nist.gov/publications/drafts/800-57/sp800-57p1r4_draft.pdf. *Cité Section 5.2*

[NLV11] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124, 2011. *Cité Section 5.1.2*

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer, 2006. *Cité Sections 3, 3.1, et 3.3*

[NS01] Phong Q. Nguyen and Igor Shparlinski. On the insecurity of a server-aided RSA protocol. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 21–35. Springer, Heidelberg, December 2001. *Cité Section 5.1.2*

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989. *Cité Section 1.4.3*

[NZZ15] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In Katz [Kat15], pages 401–426. *Cité Sections 1.5, 3, 4, 4.1, 4.2, 2.39, 4.2, 4.2, et 4.4.2*

[Ove07] Raphael Overbeck. *Public key cryptography based on coding theory*. PhD thesis, TU Darmstadt, 2007. *Cité Section 2.3*

[Par13] Srinivas Parthasarathy. Alice and bob can go on a holiday. Technical report, Algologic Technical Report 11/2012, 2013. *Cité Section 1*

[Pat96] Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–48. Springer, 1996. *Cité Section 1.4.3*

[Per69] Georges Perec. *La Disparition*. Paris, Gallimard, 1969. *Cité Section 3*

[PJ12] David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012. *Cité Section 7.2.2*

[Pol93] John M Pollard. Factoring with cubic integers. In *The development of the number field sieve*, pages 4–10. Springer, 1993. *Cité Section 1.3.2*

[PV14] David Pointcheval and Damien Vergnaud, editors. *AFRICACRYPT 14*, volume 8469 of *LNCS*. Springer, Heidelberg, May 2014. *Cité Section 7.2.2*

[PW72] William Wesley Peterson and Edward J Weldon. *Error-correcting codes*. MIT press, 1972. *Cité Section 6.1*

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978. *Cité Sections 1.4.1 et 2.1.4.2*

[RCB15] Joost Renes, Craig Costello, and Lejla Batina. Complete addition formulas for prime order elliptic curves. Cryptology ePrint Archive, Report 2015/1060, 2015. http://eprint.iacr.org/. *Cité Sections 5.4.4, 5.A, et 5.A.1*

[Reg03] Oded Regev. New lattice based cryptographic constructions. In *35th ACM STOC*, pages 407–416. ACM Press, June 2003. *Cité Section 6.1.1*

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. *Cité Section 2.2.2.2*

[Riv11] Matthieu Rivain. Fast and regular algorithms for scalar multiplication over elliptic curves. Cryptology ePrint Archive, Report 2011/338, 2011. http://eprint.iacr.org/2011/338. *Cité Section 6*

[RS60] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960. *Cité Section 2.3.1*

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. *Cité Sections 1.2.3, 1.2.3, et 2.1.4.2*

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. *Cité Section 5.1.2*

[SdM04] George Sand and Alfred de Musset. *Correspondance de George Sand et d'Alfred de Musset*. Deman, 1904. *Cité Section 1*

[Sen11] Nicolas Sendrier. Decoding one out of many. In *International Workshop on Post-Quantum Cryptography*, pages 51–67. Springer, 2011. *Cité Section 2.3.2*

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. *Cité Section 1.3.4*

[Sho15] Victor Shoup. NTL: A library for doing number theory, 2015. Version 9.4.0. `http://www.shoup.net/ntl`. *Cité Sections 5.2.2.4 et 5.6*

[Sin64] Richard Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964. *Cité Section 3.31*

[Sin99] Simon Singh. Histoire des codes secrets, 1999. *Cité Sections I, 1, 1.1, 1.1, et 4*

[SKK10] Danilo Silva, Frank R Kschischang, and Ralf Kotter. Communication over finite-field matrix channels. *IEEE Transactions on Information Theory*, 56(3):1296–1305, 2010. *Cité Section 2.3.1*

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 27–47, Berlin, Heidelberg, 2011. Springer-Verlag. *Cité Section 3.1.1*

[SS13] Damien Stehlé and Ron Steinfeld. Making ntruencrypt and ntrusign as secure as standard worst-case problems over ideal lattices. Cryptology ePrint Archive, Report 2013/004, 2013. `http://eprint.iacr.org/`. *Cité Sections 3 et 3.5.3*

[Ste94] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, August 1994. *Cité Section 4.1*

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, Heidelberg, May 2010. *Cité Section 5.2.2.3*

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014. *Cité Section 5.2.2.3*

[TK03] Harold F Tipton and Micki Krause. *Information security management handbook*. CRC Press, 2003. *Cité Section I*

[TS16] Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *International Workshop on Post-Quantum Cryptography*, pages 144–161. Springer, 2016. *Cité Section 6.7.1*

[Var57] Rom R. Varshamov. Estimate of the number of signals in error correcting codes. In *Dokl. Akad. Nauk SSSR*, volume 117, pages 739–741, 1957. *Cité Section 3.32*

[Vau03] Serge Vaudenay. The security of DSA and ECDSA. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 309–323. Springer, Heidelberg, January 2003. *Cité Section 5.5.1*

[VDCG+06] Marten Van Dijk, Dwaine Clarke, Blaise Gassend, G Edward Suh, and Srinivas Devadas. Speeding up exponentiation using an untrusted computational resource. *Designs, Codes and Cryptography*, 39(2):253–273, 2006. *Cité Section 5.1.2*

[Vou11] Panagiotis Voulgaris. *Algorithms for the closest and shortest vector problems on general lattices*. PhD thesis, PhD thesis, University of California, San Diego, 2011. *Cité Section 4*

[YW09] Alain Yger and Jacques-Arthur Weil. *Mathématiques L3-Mathématiques appliquées: Cours complet avec 500 tests et exercices corrigés*. Pearson Education France, 2009. *Cité Sections I et 2*

Contributions à la Cryptographie Post-Quantique

Résumé : Avec la possibilité de l'existence d'un ordinateur quantique, les primitives cryptographiques basées sur la théorie des nombres risquent de devenir caduques. Il devient donc important de concevoir des schémas résistants à ce nouveau type de menaces. Les réseaux euclidiens et les codes correcteurs d'erreurs sont deux outils mathématiques permettant de construire des problèmes d'algèbre linéaire, pour lesquels il n'existe aujourd'hui pas d'algorithme quantique permettant d'accélérer significativement leur résolution.
Dans cette thèse, nous proposons quatre primitives cryptographiques de ce type : deux schémas de signatures (dont une signature traçable) basés sur les réseaux, un protocole de délégation de signature utilisant du chiffrement complètement homomorphe, et une nouvelle approche permettant de construire des cryptosystèmes très efficaces en pratique basés sur les codes.
Ces contributions sont accompagnées de paramètres concrets permettant de jauger les coûts calculatoires des primitives cryptographiques dans un monde post-quantique.

Mots clés : Cryptographie Post-Quantique · Chiffrement · Signature · Sécurité

Contributions to Post-Quantum Cryptography

Abstract: In the likely event where a quantum computer sees the light, number theoretic based cryptographic primitives being actually in use might become deciduous. This results in an important need to design schemes that could face off this new threat.
Lattices and Error Correcting Codes are mathematical tools allowing to build algebraic problems, for which – up to-date – no quantum algorithm significantly speeding up their resolution is known.
In this thesis, we propose four such kind cryptographic primitives: two signatures schemes (among those a traceable one) based on lattices, a signature delegation protocol using fully homomorphic encryption, and a new framework for building very efficient and practical code-based cryptosystems.
These contributions are fed with concrete parameters allowing to gauge the concrete costs of security in a post-quantum world.

Keywords: Post-Quantum Cryptography · Encryption · Signature · Security

---

XLIM-DMI - UMR CNRS n° 7252
123, avenue Albert Thomas
87 060 Limoges Cedex