Contribution à la cryptographie post-quantique basée sur les codes correcteurs d'erreurs en métrique rang : Hash Proof Systems et cryptographie à bas coût

Yann Connan

Thèse de doctorat

Université de Limoges

Université de Limoges
École doctorale nº 610 : SISMI
Laboratoire XLIM

Thèse pour obtenir le grade de
Docteur de l'université de Limoges

Spécialité : Informatique

Présentée et soutenue par
Yann CONNAN

Le 9 avril 2020

Contribution à la cryptographie post-quantique basée sur les codes correcteurs d'erreurs en métrique rang : Hash Proof Systems et cryptographie à bas coût

Thèse dirigée par Philippe Gaborit

JURY :
Président du jury
M. Phan Duong Hieu

Rapporteurs
M. Otmani Ayoub
M. Aguilar Carlos

Examinateurs
M. Blazy Olivier
M. Bidoux Loïc
M. Gaborit Philippe

« Nos connaissances sont une goutte,
notre ignorance, un océan »
Isaac Newton

À ma famille,
à mes amis,

Table des matières

Liste des tableaux

Table des figures

Notations

Notations mathématiques classiques	
$\mathbb{P}$	Fonction de probabilité
$\mathbb{F}_q$	Corps fini à q éléments, q puissance d'un nombre premier
$\mathbb{F}_{q^m}$	Corps fini à q^m éléments, q puissance d'un nombre premier
$\mathcal{B}$	Base d'un espace vectoriel
$\mathbf{I}_n$	Matrice identité de dimension $n \times n$
$\xleftarrow{\$}$	Choix d'un élément aléatoire uniforme d'un ensemble donné
$\triangleq$	Egal par définition
$\lvert A \rvert$	Cardinal de l'ensemble A
$\langle P \rangle$	Idéal engendré par le polynôme P
$\binom{n}{k}_q$	Coefficient binomial de Gauss
$\mathcal{P}$	Problèmes résolubles en temps polynomial
$\mathcal{NP}$	Problèmes dont le résultat est vérifiable en temps polynomial
$\mathcal{O}$	Complexité asymptotique d'un algorithme
$\mathcal{U}$	Distribution uniforme
$\sim$	Suivant la loi de distribution
$\lfloor x \rfloor$	Plus grand entier inférieur à x
$\perp$	Symbole d'interruption
$a \lvert b$	Concatenation de a et b
$\mathsf{GL}_m(q)$	Groupe linéaire des matrices inversibles $m \times m$ dans $\mathbb{F}_q$
$\langle E, F \rangle$	Espace produit de E par F

Notations mathématiques spécifiques

$\mathcal{V}$	Espace vectoriel ambiant égal à $\mathbb{F}_{q^m}^n$
$\|\|\mathbf{x}\|\|$	Poids ou rang du vecteur $\mathbf{x}$, selon le contexte
$x^{[i]}$	q-degré i de x, *i.e.* x^{q^i}
$\mathsf{Supp}(\mathbf{x})$	Support du vecteur $\mathbf{x}$
$\mathbf{M}_{\mathbf{x}}$	Matrice de $\mathbb{F}_q^{m\times n}$ associée à $\mathbf{x} \in \mathbb{F}_{q^m}^n$
$\mathcal{S}_w(\mathcal{V})$	Sphères de rayon w dans $\mathcal{V}$
$\mathcal{B}_w(\mathcal{V})$	Boules de rayon w dans $\mathcal{V}$
$\mathfrak{S}_w^n(\mathcal{V})$	n-uplets de vecteurs de même support de dimension w
$\mathsf{rot}(\mathbf{v})$	Matrice circulante associée à $\mathbf{v}$
θ	Graine aléatoire
$\sim$	Relation d'équivalence pour deux vecteurs de même support
$\|\|_{i=0}^n a_i$	Concatenation d'une suite d'éléments, équivalent à $a_1\|a_2\|\ldots\|a_n$
$\phi_{\mathcal{B}}$	Transforme un vecteur de $\mathbb{F}_{q^m}^n$ en sa matrice associée dans $\mathbb{F}_q^{m\times n}$

Codes correcteurs

d_H	Distance de Hamming
d_R	Distance rang
$\mathcal{C}$	Désigne un code correcteur d'erreurs
$\mathbf{G}$	Matrice génératrice d'un code
$\mathbf{H}$	Matrice de parité d'un code
δ	Distance minimale d'un code
$[n,k]_q$	Code $\mathbb{F}_q$-linéaire de longueur n et dimension k
$[n,k]_{q^m}$	Code $\mathbb{F}_{q^m}$-linéaire de longueur n et dimension k
$[n,k,\delta]_q$	Code $\mathbb{F}_q$-linéaire de longueur n, dimension k et dist. min. δ
$[n,k,\delta]_{q^m}$	Code $\mathbb{F}_{q^m}$-linéaire de longueur n, dimension k et dist. min. δ
(n,M,δ)	Code de longueur n, contenant M éléments et de dist. min. δ
$\mathsf{Gab}_{\mathbf{g}}$	Matrice génétrice du code de Gabidulin associée au vecteur $\mathbf{g}$
$\mathcal{I}(\mathbf{v})$	Matrice idéale induite par $\mathbf{v}$
$\mathcal{QC}_{\mathbf{s}}^{syst}(\mathbb{F}_{q^m})$	Ensemble de matrices quasi-cycliques, voir chapitre 2
$\mathcal{I}_{\mathbf{s}}^{syst}(\mathbb{F}_{q^m})$	Ensemble de matrices idéales, voir chapitre 2

Relatif aux preuves de connaissance

$\mathcal{K}$	Extracteur de connaissance
P	Prouveur honnête
$\tilde{\mathsf{P}}$	Prouveur malhonnête
S	Simulateur
V	Vérifieur honnête
$\tilde{\mathsf{V}}$	Vérifieur malhonnête
Π	Preuve de connaissance non-interactive
$\mathbf{Q} * \mathbf{v}$	Produit défini par $\mathbf{Q} * \mathbf{v} = \phi_{\mathcal{B}}^{-1}(\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{v}))$

Relatif au HPS et ses applications

$\mathcal{X}$	Espace ambiant
$\mathcal{CT}$	Espace des chiffrés / CipherText
$\mathcal{PT}$	Espace des messages en clair / PlainText
pw	Password / Mot de passe
$\mathcal{L}$	Langage
$\mathcal{L}_\mu$	Ensemble des chiffrés d'un message μ
$\mathcal{L}_\mu^*$	Ensemble des éléments qui se déchiffrent en μ
hk	Clé de hachage dans un HPS
hp	Clé de projection
W	Mot du langage
$\mathcal{H}_{\mathsf{hk}}$	Haché généré à partir de hk
$\mathcal{H}_{\mathsf{hp}}$	Haché généré à partir de hp
HashKG	Algorithme de HPS générant la clé de hachage hk
ProjKG	Algorithme du HPS générant la clé de projection hp
Hash	Algorithme du HPS générant le hashé $\mathcal{H}_{\mathsf{hk}}$
$\mathsf{ProjHash}$	Algorithme du HPS générant le haché $\mathcal{H}_{\mathsf{hp}}$
$\mathsf{Exp}_{\mathfrak{K}}^{smooth-i}$	Expérience utilisée pour définir la KV-smoothness
$\mathsf{Adv}_{\mathcal{A}}^{smooth}$	Avantage de l'adversaire dans la preuve de KV-smoothness
$\mathcal{O}_{\mathbf{s},\mathbf{g}}$ / $\tilde{\mathcal{O}}_{\mathbf{s},\mathbf{g}}$	Oracles définis dans la preuve de KV-smoothness
$G_{i,\mathcal{A}}(\mathfrak{K})$ / G_i	Jeux définis dans la preuve de KV-smoothness
$\mathcal{D}_k$ / $\mathcal{D}_k^*$	Algorithmes définis dans la preuve de KV-smoothness

Acronymes

AES	Advanced Encryption Standard
AKE	Authenticated Key Exchange
ALU	Arithmetic Logic Unit
API	Application Programming Interface
AVX	Advanced Vectors Extensions
BIKE	Bit Flipping Key Encapsulation
BPR	Modèle de sécurité introduit par Bellare, Pointcheval et Rogaway
CLMUL	Carry-Less Multiplication
CCM	Counter with Cipher block chaining Message authentication code
CMT	Commitment
CRS	Common Reference String
CS	Cramer et Shoup
DES	Data Encryption Standard
DFR	Decryption Faillure Rate
FIPS	Federal Information Processing Standards
FIRSD	Flexible Ideal Rank Syndrome Decoding
GCM	Galois Counter Mode
GL	Gennaro et Lindell
HB	Protocole d'authentification symétrique d'Hopper et Blum
HPS	Hash Proof System
IND-CCA	Indistinguable contre les attaques à chiffré choisi
IND-CPA	Indistinguable contre les attaques à clair choisi
IRSD	Ideal Rank Syndrome Decoding
KEM	Key Exchange Mechanism
KV	Katz et Vaikutanathan
HB_{LRE}	Variante du protocole HB adaptée au problème LRE
LAKE	Low rAnk parity check code Key Exchange
LOCKER	LOw rank parity ChecK codes EncRyption
LPN	Learning from Parity with Noise
LRE	Learning with Rank Errors
LRPC	Low Rank Parity Check
LWE	Learn With Error
MD5	Message Digest 5
MITM	Attaque de type Man In The Middle (Homme du milieu)
NH	Non-Homogeneous

Acronymes

NHIRSD	Non-Homogeneous Ideal Rank Syndrome Decoding
NIST	National Institute of Standard and Technology
OB	Oblivious Transfert
PAKE	Password Authenticated Key Exchange
PKE	Public Key Encryption
RAM	Random Access Memory
$\mathcal{RCSP}$	Rank Concatenated Stern's Protocol
RFID	Radio Frequency IDentification
RQC	Rank Quasi Cyclic
ROLLO	Réunion des schémas LAKE, LOCKER et Ouroboros-R
RQCSD	Rank Quasi Cyclic Syndrome Decoding
RSD	Rank Syndrome Decoding
RBC	Rank-Based Cryptography library
RSP	Response
RSR	Rank Support Recovery
SD	Syndrome Decoding
SHA	Secure Hash Algorithm
UC	Universal Composability
WE	Witness Encryption
XOR	Opération OU Exclusif (eXclusive OR)
ZK	Zero-Knowledge

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Philippe Gaborit, pour la bienveillance et la confiance qu'il a su m'apporter. Je remercie également Loïc Bidoux et Slim Bettaieb qui ont encadré ma thèse, merci pour votre aide, vos conseils et pour tous les bons moments qu'on a pu partager ensemble. Merci aussi à Jean-Claude Barbezange, qui a été mon manager à Worldline pendant une grande partie de la thèse.

Je remercie tous les collègues de Worldline que j'ai connus, et en particulier ceux du pôle recherche et développement. Une mention spéciale à Olivier et Céline qui m'ont épaulé dans mes premiers pas avec les microcontrôleurs.

Merci à Olivier Blazy et Damien Sauveron pour votre aide et les échanges que l'on a pu avoir. Merci également aux doctorants de l'université de Limoges, Nicolas, Mathieu, Maxime, Neals, Adrien et Laura, qui m'ont facilement accueilli parmi eux, votre bonne humeur était contagieuse.

Merci à tous mes amis de Bretagne qui, malgré la distance qui nous a séparés, ont toujours été présents et qui tiennent une place importante dans ma vie. Merci également à mes amis de Lille dont je dois malheureusement m'éloigner et qui vont beaucoup me manquer. Enfin je remercie ma famille qui m'a toujours soutenu même dans les moments difficiles.

Chapitre 1

Introduction

1.1 Naissance et évolution de la cryptologie

Etymologie et définition

Le mot cryptographie vient des mots en grec ancien kruptos « caché » et graphein « écrire ». La cryptographie est la discipline scientifique qui étudie les techniques permettant de protéger des communications (ou plus généralement des données, quelles qu'elles soient) en la présence d'une tierce partie que l'on nomme généralement adversaire. Elle peut viser différents objectifs, comme par exemple assurer la confidentialité d'un message (le message ne doit pas pouvoir être lu par un adversaire), son intégrité (son contenu ne doit pas pouvoir être altéré de manière indécelable), l'authentification du message (vérifier qu'une entité est bien la source du message, lorsque cette entité est une personne on parle alors d'identification) ou encore la non-répudiation (l'auteur du message ne peut alors plus nier son implication).

Elle est un domaine en constante évolution, contrainte à s'adapter à la capacité de calculs dont un adversaire dispose, qui augmente au fil du temps, et à la découverte de nouvelles attaques plus performantes.

Loin de moi l'idée de décrire dans cette partie, de façon exhaustive, l'ensemble de la cryptographie de ses débuts à la cryptographie dite moderne, des ouvrages entiers sur le sujet existant d'ores et déjà, mais plutôt d'en parcourir certaines étapes marquantes, afin de contextualiser l'écriture de cette thèse et d'introduire la cryptographie dite post-quantique et les enjeux auxquels elle est

confrontée.

Chiffrement par décalage

A ses début, les systèmes de chiffrement étaient très simples à comprendre et quelque peu naïfs. Un des premiers systèmes de chiffrement a été le chiffrement à décalage utilisé historiquement par Jules César. Celui-ci consistait simplement à décaler chacune des lettres de l'alphabet d'un certain nombre de rangs, en recommençant à partir du A lorsque la lettre Z était atteinte. Pour déchiffrer le message, il suffisait de décaler les lettres dans l'autre sens pour retrouver le message d'origine. La sécurité reposait principalement sur le fait qu'un ennemi ne connaisse pas le système de chiffrement utilisé (et que peu d'entre-eux savaient lire à cette époque).

Un des principes de Kerckhoffs

A noter que de nos jours, c'est exactement l'inverse qui est préconnisé, à savoir que la sécurité d'un système de chiffrement ne doit justement pas résider dans la dissimulation dudit système mais dans la clé utilisée pour effectuer le chiffrement. Autrement dit, plus un système est connu de tous et résiste au cours du temps, et plus celui-ci peut être considéré comme fiable. Cet enseignement est tiré d'un des fameux principes de Kerckhoffs énoncé dans un journal des sciences militaires [Pet83], où il est écrit « Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ».

Pour faire le parallèle avec le chiffrement à décalage décrit précédemment, le système de chiffrement serait le fait de décaler les lettres de l'alphabet d'un certain pas et la clé utilisée serait le décalage en question. Une fois le système à décalage connu par un adversaire, il n'y a alors plus que 25 décalages possibles, autrement dit 25 clés possibles, pour chiffrer un message, ce qui est extrêmement simple à déchiffrer.

Chiffrement par substitution

Les systèmes de chiffrement se sont peu à peu améliorés et complexifiés. Un exemple classique est le chiffrement par substitution qui consiste à remplacer chaque lettre de l'alphabet par une autre (potentiellement la même). Ainsi, la lettre A serait remplacée par une des lettres de l'alphabet, soit 26 possibilités, la lettre B par une des lettres restantes, soit 25 possibilités, etc. Attaquer un tel

système en essayant toutes les possibilités, ce que l'on appelle une attaque par force brute, représenterait ainsi 26! possibilités, soit environ $4 \cdot 10^{26}$.

Bien que ce système soit plus difficile à casser qu'un chiffrement par décalage, il n'en est pas dénué de faiblesse pour autant. Au IX^{e} siècle, un philosophe et mathématicien dénommé Al-Kindi écrit, dans ce qui est considéré comme le premier manuscrit traitant de cryptanalyse, une méthode de déchiffrement par analyse fréquentielle. En effet, on peut remarquer qu'avec un chiffrement par substitution, une même lettre sera toujours chiffrée de la même façon et que dans une langue donnée, la fréquence d'apparition des lettres n'est pas la même. En français par exemple, la lettre « e » apparaît bien plus souvent que la lettre « k ». Il est ainsi possible, à partir d'un texte chiffré, de retrouver la langue dans laquelle le message initial a été écrit en observant la fréquence des lettres qui le composent. Par suite, on peut associer, avec une certaine probabilité d'erreur, une lettre du chiffré avec son équivalent dans le texte en clair, et déchiffrer ainsi le message originel. On peut remarquer que plus le message chiffré est long, et plus une telle méthode s'avérera efficace.

Chiffrement de Vigénère

Décrit pour la première fois par Giovan Battista Bellaso en 1553, il peut être vu comme un prolongement du chiffrement par substitution, avec l'avantage de réduire fortement la principale faiblesse de ce dernier, à savoir sa vulnérabilité à l'analyse fréquentielle.

Son principe est simple. On associe à chaque lettre de l'alphabet sa position dans l'alphabet sous la forme d'un nombre entre 0 et 25. Ainsi A correspond au zéro, B au 1, etc. On utilise un mot ou une phrase comme clé de chiffrement, prenons un exemple simple comme le mot CLEF, et considérons le message initial LECHIFFREINDECHIFFRABLE. Pour chiffrer le message, on va superposer chacune des lettres du message initial avec une lettre de la clé de chiffrement, et répéter la clé de chiffrement autant de fois que nécessaire pour que chaque lettre soit chiffrée.

Il suffit ensuite d'ajouter les nombres correspondants aux lettres pour obtenir la lettre du chiffré. Si la somme excède 25, on repart à partir de zéro, autrement dit nous effectuons les sommes modulo 26.

Message	L	E	C	H	I	F	F	R	E	I	N	D	E	C	H	I	F	F	R	A	B	L	E
	11	4	2	7	8	5	5	17	4	8	13	3	4	2	7	8	5	5	17	0	1	11	4
Clé	C	L	E	F	C	L	E	F	C	L	E	F	C	L	E	F	C	L	E	F	C	L	E
	2	11	4	5	2	11	4	5	2	11	4	5	2	11	4	5	2	11	4	5	2	11	4
	13	15	6	13	10	16	9	22	6	19	17	8	6	13	11	12	7	16	21	5	3	22	8
Chiffré	N	P	G	M	K	Q	J	W	G	T	R	I	G	N	L	N	H	Q	V	F	D	W	I

On peut constater qu'une même lettre ne sera plus toujours chiffrée de la même façon en fonction de sa position dans le message, ce qui réduit l'impact d'une attaque par analyse fréquentielle. Ce chiffrement aura résisté pendant près de trois siècles, mais en 1863, Friedrich Kasiski sera le premier à publier une méthode générale de déchiffrement. La méthode consiste à chercher des digrammes ou trigrammes (ou des ensembles de taille plus élevée si nécessaire) qui se répètent dans le chiffré afin d'en déduire des multiples de la longueur de la clé. En effet, certains digrammes ou trigrammes sont plus fréquents que d'autres selon les langues. En français, le digramme « ER » est très fréquent par exemple. Si ce digramme est chiffré avec les mêmes lettres de la clé de chiffrement, alors les chiffrés seront des digrammes égaux. On peut utiliser cette astuce afin de retrouver la taille de la clé. Bien sûr, certains digrammes égaux seront des faux positifs, il faudra donc chercher le multiple de la taille de la clé le plus probable. Une fois la taille de la clé déduite correctement, une analyse fréquentielle peut alors s'effectuer en considérant les sous-ensembles de lettres qui sont chiffrés de la même manière. Il devient alors assez simple de faire céder le chiffrement et de retrouver le message initial.

1.2 Première révolution

Une première révolution dans le domaine de la cryptographie va s'opérer durant la seconde guerre mondiale avec l'utilisation d'une famille de machines appelées « Enigma ». La première machine fut inventée par l'Allemand Arthur Scherbius à partir d'un brevet du Néerlandais Hugo Koch, datant de 1919. Utilisée pour chiffrer et déchiffrer, elle fut alors considérée comme « inviolable ».

Ces machines étaient constituées de rotors. A chaque pression sur une touche du clavier, le premier rotor était décalé d'un cran. Si ce dernier avait effectué un tour complet, il entrainait avec lui le déplacement du second rotor, etc.

Chaque rotor peut être considéré comme un chiffrement par substitution, et à

chaque pression sur une touche, la substitution est modifiée. Pour qu'un même caractère soit chiffré avec la même substitution, il faut donc attendre que l'ensemble des rotors soit revenu à leur position initiale. Ainsi, un même caractère sera chiffré de multiples façons différentes. Avec trois rotors composés chacun de 26 crans, cela représente une période d'environ $26^3 = 17\ 576$ (ce nombre n'est pas tout à fait exact mais nous n'entrerons pas dans les détails ici). Les allemands limitaient la longueur de leurs messages à quelques centaines de lettres évitant ainsi toute chance de répétition de combinaisons des rotors, empêchant par là même les cryptanalystes d'accéder à des indices précieux.

Pour venir à bout des chiffrés générés par les machines Enigma, les Alliés décidèrent de centraliser tous les éléments connus sur Enigma à Bletchley Park. Les cryptanalistes Britanniques dont Alan Turing purent poursuivre les travaux initiés par le mathématicien polonais Marian Rejewski sur la mise au point d'une machine appelée « bombe électromécanique ». Ce même Alan Turing qui en 1936 imagina un modèle abstrait permettant à un appareil mécanique de fournir le résultat de calculs, introduisant par la même occasion la notion d'algorithme. On nomma par la suite cet appareil une machine de Turing. Le modèle de la machine de Turing et le succès de la « bombe électromécanique » furent à l'origine de la conception des premiers ordinateurs et de l'informatique, qui marqua un réel tournant pour le domaine de la cryptographie.

1.3 Seconde révolution

Jusqu'à présent, tous les systèmes de chiffrement que nous avons présentés supposaient que l'émeteur et le récepteur du message disposent au préalable d'un secret partagé, à savoir la clé de chiffrement utilisée. Cette même clé de chiffrement était également utilisée pour déchiffrer et retrouver le message initial. En 1976, dans leur article intitulé « New directions in cryptography » [DH76], Whitfield Diffie et Martin Edward Hellman vont établir un tournant dans l'histoire de la cryptographie en proposant un schéma exploitant le problème du logarithme discret et permettant à deux entités/individus de pouvoir s'échanger confidentiellement une clé commune sans pour autant partager un secret commun au préalable, ce problème étant resté ouvert jusqu'alors. Ils émettent alors l'idée d'un schéma de chiffrement contenant deux clés distinctes, l'une qui aurait pour fonction de chiffrer les messages, et l'autre de les déchiffrer, sans pour autant savoir si un tel schéma est concevable. Une telle fonc-

tionnalité permettrait de pouvoir communiquer de manière confidentielle sans le partage préalable d'une clé commune. Par ailleurs, cela permettrait par là même d'envisager la construction d'une signature numérique, équivalent numérique de la signature manuscrite et permettant d'authentifier la source d'un message.

En 1977, Ronald Rivest, Adi Shamir, et Léonard Adleman parvinrent à concevoir un tel schéma, le fameux chiffrement RSA [RSA78], nommé ainsi de par les initiales de leurs auteurs. Un tel schéma sera dénommé schéma de chiffrement asymétrique, étant donné que les clés sont différentes pour l'émetteur et pour le récepteur du message. Ils parvinrent également à concevoir une signature numérique basée sur ce problème. Celui-ci se base essentiellement sur la difficulté de factoriser un nombre produit de deux grands nombres premiers et sur le théorème d'Euler en arithmétique modulaire.

Ces deux protocoles ont eu un impact majeur dans la cryptographie moderne, puisqu'ils ont permis de s'affranchir du problème du partage du secret nécessaire à l'emploi de la cryptographie symétrique.

1.4 Vers une troisième révolution ?

Au début des années 1980, Richard Feynman va le premier, dans son article intitulé « Simulating physics with computers » [Fey82] suggérer qu'un ordinateur basé sur la mécanique quantique puisse être plus puissant qu'une machine de Turing. Les ordinateurs classiques que nous utilisons de nos jours stoquent et manipulent l'information sous la forme de bits, qui sont en fait des transistors possédant deux états possibles, 0 ou 1. Toute information est *in fine* représentée sous la forme d'une succession de 0 et de 1. L'ordinateur quantique quant à lui va représenter l'information sous la forme de qubits qui sont des éléments pouvant avoir l'état 0, l'état 1, mais également tout un ensemble d'états intermédiaires que l'on considère alors comme étant à la fois 0 et 1. Cette propriété est appelée *principe de superposition*. De plus, le changement d'état d'un qubit peut entraîner le changement d'état d'un autre qubit, même si celui-ci est situé à grande distance du premier. Cette deuxième propriété des qubits est appelée *intrication quantique*.

Il est alors théoriquement possible d'exploiter ces deux propriétés pour mettre au point des algorithmes d'un genre nouveau et permettant de résoudre cer-

tains problèmes bien plus rapidement que ne le ferait un ordinateur classique. La supériorité d'un ordinateur quantique à résoudre certains problèmes face à un ordinateur classique fut démontrée en 1994 dans un article intitulé « On the power of quantum computation » [Sim94] écrit par Daniel R. Simon .

En 1996, dans son article intitulé « A fast quantum mechanical algorithm for database search » [Gro96], Lov Kumar Grover décrit un algorithme quantique permettant de rechercher un ou plusieurs éléments parmi un ensemble à N éléments non classés en un temps en $\mathcal{O}(\sqrt{N})$ et en utilisant un espace de stockage en $\mathcal{O}[\log(N)]$, contre un temps en $\mathcal{O}(N)$ pour un ordinateur classique. Cet algorithme a des répercutions concrètes sur l'utilisation de l'algorithme de chiffrement symétrique standard que nous utilisons aujourd'hui, à savoir l'AES (Advanced Encryption Standard [FIPS 197]) ainsi que sur les fonctions de hachage SHA2 [FIPS 180-2] et SHA3 [FIPS 180-4], imposant de doubler la taille des clés utilisées pour l'AES et de tripler la longueur des hashés SHA2 et SHA3 pour un niveau de sécurité équivalent, dû au fait que l'algorithme de Grover peut être utilisé conjointement avec le paradoxe des anniversaires pour obtenir une collision dans la fonction de hachage [MVZJ18].

Trois ans plus tard, Peter Shor présente dans son article intitulé « Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer » [Sho99] deux algorithmes quantiques, l'un permettant de résoudre plus efficacement le problème de la factorisation d'un entier produit de deux grands nombres premiers et l'autre permettant de résoudre plus efficacement le problème du logarithme discret, passant ainsi d'une complexité qui était alors en temps exponentiel dans les meilleurs algorithmes existants en une complexité en temps polynomial, rendant ainsi les protocoles RSA, DHE, DSA, ECDSA et ECDHE massivement utilisés de nos jours obsolètes si un ordinateur quantique suffisamment puissant (en terme de nombre de qubits interconnectés et de la maitrise de leurs précisions) venait à être mis au point [MVZJ18]. De nos jours, de grands acteurs tels que la NASA ou Google investissent fortement dans la conception de telles machines.

1.5 De futurs standards

Cette menace sans précédent à conduit à l'émergence d'un nouveau domaine d'étude, à savoir la cryptographie dite post-quantique, *i.e.* la cryptographie résistante à la fois aux attaques par ordinateurs quantiques et par ordinateurs classiques. La communauté scientifique a ainsi été en mesure d'élaborer des schémas cryptographiques résistants aux algorithmes de Grover et de Shor, et supposés résistants aux attaques menées par ordinateur quantique, dans le sens ou aucun algorithme quantique connu à ce jour n'est capable de les attaquer en un temps à complexité polynomiale par rapport à la taille n de la clé de sécurité utilisée [BBD09] (précisons cependant qu'il n'est pas prouvé que de tels algorithmes quantiques ne puissent voir le jour à l'avenir). Ces derniers reposent sur des domaines variés des mathématiques, à l'image de la cryptographie basée sur les réseaux euclidiens, sur les codes correcteurs d'erreurs, sur les fonctions de hachage, sur les polynômes multivariés ou encore sur la recherche d'isogénies de courbes elliptiques super-singulières.

Pour préparer l'avenir et pouvoir poursuivre l'échange de données de manière sécurisée dans un monde où la suprémacie quantique aurait été atteinte, le National Institute of Standards and Technology (NIST) a publié en 2016 un appel à candidature en vue de la standardisation d'algorithmes post-quantiques pour le chiffrement asymétrique, la signature numérique et des protocoles d'échange de clés [Nis16a].

1.6 L'internet des objets

L'Internet des objets (Internet of things ou IoT, en anglais) désigne la tendance actuelle consistant à connecter de plus en plus d'objets du quotidien entre-eux et au réseau internet [San10]. Les enjeux de protection de la vie privée étant de plus en plus présents dans la société (en témoigne le règlement général sur la protection des données (RGPD) de 2016), la problématique de la sécurisation des données générées par l'IoT est un challenge majeur adressé à la communauté scientifique. La résolution de cette problématique est d'autant plus complexe que les objets connectés sont contraints en terme de consommation énergétique, de puissance de calculs ou encore de mémoire disponible. Le domaine de la cryptographie visant à répondre à ces contraintes est alors appelé cryptographie à bas coût (ou lightweight cryptography) [EKP+07].

1.7 Contexte de la thèse

Cette thèse CIFRE s'est inscrite dans la continuité d'un partenariat entre la société Worldline, leader européen dans le secteur du paiement et des services transactionnels, sur le site de Noyelles-lès-Seclin et le laboratoire de recherche XLIM situé à Limoges. En effet, ces deux entités ont participé conjointement à l'appel à standardisation du NIST en proposant quatre schémas : deux schémas de chiffrement asymétrique , HQC et RQC [AMBD+18] (ces deux schémas possédent également une version protocole d'échange de clés), et deux protocoles d'échanges de clés, BIKE [ABB+17] et Ouroboros-R [MAB+17]. Tous ces schémas sont basés sur les codes correcteurs d'erreurs et chaque type de schéma est décliné selon deux métriques différentes : RQC correspond à la déclinaison de HQC en métrique rang, et Ouroboros-R à BIKE en métrique rang.

	Métrique de Hamming	Métrique rang
Chiffrement asymétrique	HQC	RQC
Protocole d'échange de clés	BIKE / HQC	Ouroboros-R / RQC

Les quatre schémas soumis au NIST par XLIM, Worldline et leurs partenaires

La thèse ne s'est pas focalisée sur un sujet spécifiquement mais plutôt sur un ensemble de plusieurs problématiques autour de la cryptographie basée sur la métrique rang et la cryptographie à bas coût, qui ont débouché principalement sur trois axes d'étude.

1.7.1 Un Hash Proof System basé sur le schéma RQC

Concept introduit en 2002 par Cramer et Shoup, un Hash Proof System (HPS) est une primitive cryptographique conçue à la base pour construire le premier schéma de chiffrement à clé publique sécurisé contre des attaques à chiffrés choisis (CCA-2 plus exactement) [CS02a]. Cette primitive a depuis trouvé d'autres applications, comme par exemple la construction de protocoles d'authentification, le plus souvent basés sur des mots de passe, des preuves à divulgation nulle de connaissance avec vérifieur honnête (HVZK, Honest Verifier Zero-Knowledge proofs) ou encore des chiffrements à témoin (WE, Witness Encryption). Peu de constructions existent à ce jour et un seul HPS basé sur

les codes correcteurs d'erreurs a été élaboré [Per13]. Celui-ci souffre cependant d'une erreur lors de la preuve de sa propriété d'universalité en considérant des clés choisies aléatoirement au lieu de démontrer le résultat pour toutes les clés possibles.

La deuxième problématique a ainsi porté sur l'élaboration d'un autre HPS post-quantique basé sur les codes correcteurs d'erreurs. Sa construction s'est basée sur le schéma de chiffrement à clé publique RQC et tente de répondre à une problématique qui semble inhérente à la conception des HPS post-quantiques, à savoir la présence d'éléments qui ne sont pas des chiffrés valides d'un message donné m mais qui se déchiffrent tout de même en m. Ces éléments posent problème car ils sont bien souvent indétectables dans le déroulement des protocoles cryptographiques et n'ont pas la forme attendue d'un chiffré valide, ouvrant possiblement la voie à des attaques exploitant cette faiblesse.

1.7.2 Le problème LRE

Un problème bien connu en cryptographie, et notamment en cryptographie à bas coût est le problème nommé « Learn from parity with noise » (LPN). Le problème consiste à retrouver une valeur secrète s à partir d'un ensemble de valeurs qui ont une probabilité fixée $p \in]0, 1[$ d'être erronées. Le problème LPN est intéressant à la fois d'un point de vue théorique et d'un point de vue pratique. En effet, d'un point de vue théorique, il offre une garantie de sécurité très importante. En effet, ce problème est équivalent au problème du décodage d'un code linéaire aléatoire, un problème largement étudié durant la deuxième moitié du siècle dernier et qui a été prouvé $\mathcal{NP}$-complet [BMVT78]. De plus, aucun algorithme quantique connu ne permet d'affaiblir sa sécurité. D'un point de vue pratique, les protocoles basés sur ce problème sont souvent simples et efficaces en terme de taille des messages durant l'échange. Ces protocoles sont d'ailleurs des candidats privilégiés pour le fonctionnement d'objets aux capacités très restreintes comme les tags RFID. Ce problème peut avoir de multiples applications différentes en cryptographie [Pie12].

Le premier protocole d'authentification basé sur le problème LPN est le protocole HB des initiales de ses auteurs Hopper and Blum [HB01]. Ce protocole est relativement simple et bien adapté à la cryptographie à bas coût. La première problématique a été la suivante : Est-il possible d'adapter le protocole HB à la métrique rang ? Pour y répondre, un nouveau problème va être introduit, le

problème Learning with Rank Errors (LRE), adaptation du problème LPN à la métrique rang.

1.7.3 Implémentations sur architecture 32 bits

Une partie de la thèse a consisté à adapter et implémenter des schémas cryptographiques post-quantiques sur différents microcontrôleurs dotés de registres dont la taille maximale n'excédait jamais 32 bits. A titre de comparaison, la taille des registres utilisés par défaut sur nos ordinateurs modernes est généralement de 64 bits, et les processeurs peuvent disposer de registres de 128, 256 voire 512 bits. Des registres plus larges peuvent permettre d'effectuer plusieurs opérations requérant une taille de registre plus petite simultanément et donc avoir un impact fort sur les performances des algorithmes cryptographiques, c'est ce que l'on appelle la *vectorisation* [Lom11]. A l'inverse, des tailles de registres plus restreintes vont accroître le nombre d'opérations à effectuer pour un algorithme donné, résultant en des temps d'exécution plus long. De plus, la mémoire disponible sur ces microarchitectures est bien plus limitée, ce qui peut s'avérer problématique dans certains cas, et des adaptations sont alors nécessaires. Enfin, les améliorations de performances sur ces microarchitectures se font parfois au détriment d'un espace mémoire alloué plus important, et un équilibre est alors à trouver pour tirer parti au mieux des ressources disponibles.

La troisième problématique a donc été de savoir si les schémas soumis au NIST par le laboratoire XLIM et l'entreprise Worldline pouvaient être portés sur des microcontrôleurs 32 bits, et, dans l'affirmative, d'en améliorer les performances en choisissant judicieusement les algorithmes les plus adaptés, en les comparant à d'autres algorithmes existants, et en améliorant leurs implémentations.

1.7.4 Plan de la thèse

Le chapitre 2 sera dédié à la théorie des codes correcteurs d'erreurs. En section 2.1, nous présenterons les diverses branches de la cryptographie post-quantique. Puis, en section 2.2 nous introduirons des notions liées à la théorie des codes correcteurs d'erreurs. La section 2.3 introduira la métrique rang et prolongera les notions existantes en métrique de Hamming. La section 2.4 décrira quelques codes utilisés en cryptographie post-quantique. La section 2.5 quant à elle explicitera des problèmes difficiles en métrique rang. Enfin, la section 2.6 présentera

les schémas RQC et ROLLO, candidats à l'appel à standardisation du NIST.

Chapitre 2

Codes correcteurs et cryptographie

Ce chapitre a pour objectif d'introduire les notions de théorie des codes correcteurs nécéssaires à la compréhension des différents travaux effectués durant cette thèse. Elle débute, en section 2.1 par une présentation des principales branches constituant la cryptographie post-quantique à l'heure actuelle. La section 2.2 décrira des notions importantes de la théorie des codes correcteurs comme la métrique de Hamming, la notion de code $\mathbb{F}_q$-linéaire, de matrice génératrice et de parité d'un code, de syndrome, le poids de Hamming d'un vecteur, la distance minimale d'un code, la borne de singleton et les codes MDS, ainsi que le problème classique du décodage par syndrome. La section 2.3 introduit la métrique rang et prolonge ainsi les notions abordées dans la section précédente. Elle décrit la notion de support et de rang d'un vecteur, définit la métrique rang, explicite la notion de code $\mathbb{F}_{q^m}$-linéaire, la borne de singleton en métrique rang et les codes MRD, le coefficient binomial de Gauss, la borne d'empilement des sphères et la borne de Gilbert-Varshamov. La section 2.4 décrit certains codes correcteurs d'erreurs utilisés en métrique rang, à savoir les codes de Gabidulin, les codes quasi-cycliques, les codes idéaux et les codes LRPC. La section 2.5 quant à elle définit des problèmes difficiles utiles à l'élaboration de schémas cryptographiques en métrique rang, le problème RSD, sa version quasi-cyclique s-RQCSD, la version idéale s-IRSD et enfin la version idéale non-homogène NHIRSD permettant de réduire l'impact de récentes attaques algébriques [BBC$^+$20a, BBC$^+$20b].

2.1 Les diverses branches de la cryptographie post-quantique

L'appel à standardisation du NIST pour de nouveaux standards cryptographiques post-quantiques a débuté en janvier 2017. En réponse à l'impact qu'aurait l'algorithme de Shor, incluant le chiffrement asymétrique RSA, la signature DSA et les versions basées sur les courbes elliptiques comme ECDSA et ECDHE, le NIST a demandé qu'on lui soumette de nouveaux schémas de chiffrement asymétrique, d'échange de clés et de signatures. Ces différents schémas sont basés sur des problèmes mathématiques difficiles à résoudre et appartiennent à différents champs des mathématiques.

La majeure partie des soumissions sont basées sur des problèmes liés aux réseaux euclidiens [BDK+18, ZCH+19, DKRV18, DKL+18, BCLvV17, FHK+18, BCD+16]. Un réseau d'un espace vectoriel euclidien est un sous-groupe discret de $\mathbb{R}$, un ensemble de points répartis uniformément dans un espace donné. Il s'agit de la branche la plus étudiée en cryptographie post-quantique.

La deuxième branche de la cryptographie la plus étudiée est celle basée sur les codes correcteurs d'erreurs [BCL+17, MAB+18, AMBD+18, ABB+17, MAB+19]. Un code correcteur d'erreurs est une technique de codage de l'information permettant de détecter et de corriger des erreurs survenues lors de la transmission d'un message à travers un canal de communication. Ils sont largement utilisés de nos jours sans que nous en ayons forcément conscience, de la radio à la fibre optique en passant par les transmissions par satellite, dès lors qu'une information doit être parfaitement retranscrite. C'est cette branche de la cryptographie que nous développerons le plus ici.

Une troisième branche de la cryptographie post-quantique est celle utilisant les polynômes multivariés [CDK+20], *i.e.* des polynômes en plusieurs indéterminées à coefficients dans un anneau commutatif unitaire.

Enfin, d'autres alternatives plus minoritaires ont également été proposées, comme par exemple des signatures basées uniquement sur l'utilisation de fonctions de hashage [BHH+15], la recherche d'isogénie sur des courbes elliptiques supersingulières [JAC+17], ou encore des problèmes exploitant la forme particulière des nombres premiers de Mersenne [AJPS17].

2.2 Les codes correcteurs d'erreurs

La théorie des codes correcteurs a débuté dans les années 1940 avec le mathématicien américain Richard Hamming, et le premier code correcteur fut publié en 1950 [Ham50]. L'idée a été d'ajouter à un message donné des bits d'information supplémentaires permettant de détecter si des erreurs ont eu lieu durant la communication et de les corriger, le nombre d'erreurs pouvant être détecté n'étant d'ailleurs pas forcément égal au nombre d'erreurs corrigeables par le code. La part d'information ajoutée au message initial est appelée redondance, et un tel type de code, où le message initial apparaît en clair, est appelé code sous forme systématique.

D'une manière générale, un code peut être vu comme un ensemble de mots de longueur n à coefficients dans un ensemble fini d'éléments $\mathcal{A}$ appelé alphabet, *i.e.* des éléments de $\mathcal{A}^n$. La distance dite de Hamming entre deux mots du code a et b est alors égale au nombre de coefficients qui diffèrent entre a et b. Formellement :

Définition 1 (Métrique de Hamming)**.**

$$\forall a, b \in \mathcal{A}^n, a = (a_i)_{i\in[\![1,n-1]\!]}, b = (b_i)_{i\in[\![1,n-1]\!]},\ d_H(a,b) \triangleq \#\{i : a_i \neq b_i\}.$$

Proposition 1. *L'application d_H définie précédemment est une distance sur $\mathcal{A}^n$.*

En théorie des codes correcteurs, la plupart des codes utilisés sont des codes dits linéaires. Toute combinaison linéaire de mots de code est alors encore un mot du code. Un tel code est défini comme un sous-espace vectoriel d'un espace vectoriel de dimension finie à coefficients dans un corps fini.

Soit q une puissance d'un nombre premier p, et soit $\mathbb{F}_q$ l'unique corps fini à q éléments. Soit $\mathcal{V} = \mathbb{F}_q^n$ l'espace vectoriel à n dimensions à coefficients dans $\mathbb{F}_q$.

Définition 2 (Code $\mathbb{F}_q$-linéaires)**.** *Un code $\mathbb{F}_q$-linéaire de dimension k et de longueur n est un sous espace vectoriel de dimension k de $\mathcal{V}$ et est noté $[n,k]_q$, ou simplement $[n,k]$ lorsqu'il n'y a pas d'ambiguïté. Le code $\mathcal{C}$ peut être représenté de deux façons équivalentes :*

- ◇ *En utilisant une matrice génératrice $\mathbf{G} \in \mathbb{F}_q^{k\times n}$. Les lignes de $\mathbf{G}$ définissent alors une base du code $\mathcal{C}$*

$$\mathcal{C} = \{\mathbf{x}\mathbf{G} \mid \mathbf{x} \in \mathbb{F}_q^k\}.$$

◇ *En utlisant une matrice de parité* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. *Les lignes de* $\mathbf{H}$ *déterminent un système d'équations que doivent vérifier les éléments de* $\mathcal{C}$

$$\mathcal{C} = \{\mathbf{x} \in \mathbf{v} \mid \mathbf{H}\mathbf{x}^\top = 0\}.$$

L'espace $\mathcal{V}$ désigne l'espace vectoriel ambiant, $\mathcal{C}$ est le code et $\mathbb{F}_q^k$ est l'espace des messages. Ainsi, un message $m \in \mathbb{F}_q^k$ est transformé en un mot de code en le multipliant à droite par une matrice génératrice.

Définition 3. *Une matrice génératrice* $\mathbf{G}$ *(respectivement une matrice de parité* $\mathbf{H}$*) est dite sous forme systématique si elle est de la forme* $(\mathbf{I}_k \mid \mathbf{A})$ *(respectivement* $(\mathbf{I}_{n-k} \mid \mathbf{B})$*).*

Définition 4 (Syndrome)**.** *Soit* $\mathbf{H}$ *une matrice de parité d'un code* $\mathcal{C}$*. Les éléments de la forme* $\mathbf{H}\mathbf{x}^\top$*, avec* $\mathbf{x} \in \mathcal{V}$*, sont appelés syndromes.*

Définition 5 (Poids de Hamming)**.** *Le poids de Hamming d'un vecteur* $\mathbf{v} \in \mathcal{V}$ *(ou simplement poids de* $\mathbf{v}$*), noté* $\|\mathbf{v}\|$*, est le nombre de coordonnées non nulles de* $\mathbf{v}$*.*

Définition 6. *Soit* $\mathcal{C}$ *un code. La distance minimale* δ *du code* $\mathcal{C}$ *est la plus petite distance possible entre deux mots du code* $\mathcal{C}$

$$\delta = \min\Big(d_H(\mathbf{a},\mathbf{b}), \mathbf{a},\mathbf{b} \in \mathcal{C}, \mathbf{a} \neq \mathbf{b}\Big).$$

Proposition 2. *La distance minimale* δ *d'un code* $\mathcal{C}$ *est égale au poids minimal des mots de code non nuls de* $\mathcal{C}$*.*

Démonstration. Soit $\mathcal{C}$ un code et δ sa distance minimale. Il existe donc $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, $\mathbf{x} \neq \mathbf{y}$, tels que $d_H(\mathbf{x},\mathbf{y}) = \delta$. Comme ajouter un même vecteur à chacun d'entre-eux ne changera pas la distance qui les sépare, on a aussi $d_H(\mathbf{x}-\mathbf{y}, 0) = \delta$, et donc $\|\mathbf{x}-\mathbf{y}\| = \delta$. Le code $\mathcal{C}$ étant linéaire, $\mathbf{x}-\mathbf{y} \in \mathcal{C}$. On a ainsi montré qu'il existe un mot de code de poids δ.

Montrons désormais qu'aucun mot de code non nul ne peut avoir un poids inférieur à δ. Soit $\mathbf{x} \in \mathcal{C},\ \mathbf{x} \neq 0$. On a $\|\mathbf{x}\| = d_H(\mathbf{x}, 0) \geqslant \delta$, CQFD. Ainsi δ est le poids minimal des mots de code de $\mathcal{C}$. □

Si la distance minimale d'un code est connue, le code peut alors être décrit comme étant $[n,k,\delta]_q$ ou simplement $[n,k,\delta]$, où n désigne la taille des mots de code, k la dimension du code qui correspond aussi à la taille des messages initiaux et δ est la distance minimale du code.

Définition 7 (Borne de singleton)**.** *Un code linéaire* $[n, k, d]_q$ *vérifie toujours* $d + k \leqslant n + 1$.

Définition 8 (Code MDS)**.** *Un code atteignant la borne de singleton, i.e. vérifiant* $k = n - d + 1$ *est appelé MDS (Maximum Distance Separable code).*

Lorsque l'on considère un code $\mathbb{F}_q$-linéaire aléatoire, le problème consistant à retrouver le message initial, étant donné un vecteur qui est la somme d'un mot de code et d'une erreur de poids faible, est difficile à résoudre. Or, en cryptographie, les problèmes difficiles à résoudre sont une aubaine puisqu'ils peuvent être exploités afin de garantir la sécurité des protocoles qui les utilisent. Un problème classique utilisé en cryptographie basée sur les codes correcteurs d'erreurs est le problème du décodage par syndrome, qui, comme nous allons le voir, est équivalent au problème qui consiste à retrouver le mot de code le plus proche d'un vecteur donné, que nous avons explicité précédemment.

Définition 9 (Problème du décodage générique d'un code linéaire aléatoire)**.** *Etant donné des entiers naturels* n, k *et* w, *une matrice aléatoire* $\mathbf{G} \xleftarrow{\$} \mathbb{F}_q^{k \times n}$ *et un vecteur* $\mathbf{y}$ *tel que* $\mathbf{y} = \mathbf{mG} + \mathbf{e}$, $\mathbf{m} \xleftarrow{\$} \mathbb{F}_q^k$, $\mathbf{e} \xleftarrow{\$} \mathcal{V}$ *de poids faible* w, *retrouver* $\mathbf{m}$.

Définition 10 (Distribution SD - Syndrome Decoding)**.** *Etant donné des entiers naturels* n, k *et* w, *la distribution* $\mathsf{SD}(n, k, w)$ *est la distribution obtenue en choisissant* $\mathbf{H} \xleftarrow{\$} \mathbb{F}_q^{(n-k) \times n}$ *et* $\mathbf{x} \xleftarrow{\$} \mathcal{V}$ *de poids* w, *et renvoyant le doublet* $(\mathbf{H}, \mathbf{Hx}^\top)$.

Définition 11 (Problème SD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathbb{F}_q^{(n-k) \times n} \times \mathcal{V}$ *issu de la distribution* SD, *le problème* $\mathsf{SD}(n, k, w)$ *consiste à trouver* $\mathbf{x} \in \mathcal{V}$ *de poids* w *tel que* $\mathbf{Hx}^\top = \mathbf{y}^\top$.

Définition 12 (Version décisionnelle du problème SD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathbb{F}_q^{(n-k) \times n} \times \mathcal{V}$, *la version décisionnelle du problème* SD *consiste à pouvoir déterminer, avec un avantage non négligeable, si* $(\mathbf{H}, \mathbf{y})$ *a été généré à partir de la distribution* SD *ou de la distribution uniforme sur* $\mathbb{F}_q^{(n-k) \times n} \times \mathcal{V}$.

Proposition 3. *Le problème du décodage générique d'un code linéaire aléatoire est équivalent au problème* SD.

Démonstration. Supposons que l'on sache résoudre le problème du décodage générique d'un code linéaire aléatoire et considérons une instance du problème de recherche SD. Soit $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, $\mathbf{s} \in \mathcal{V}$ et $w \in \mathbb{N}^*$ tel qu'il existe $\mathbf{x} \in \mathbb{F}_q^n$ vérifiant $\mathbf{Hx}^\top = \mathbf{s}^\top$ et $\|\mathbf{x}\| = w$ et déterminons $\mathbf{x}$.

En transposant, on obtient $\mathbf{xH}^\top = \mathbf{s}$. Soit $\mathbf{e} \in \mathbb{F}_q^{n-k}$ un vecteur de poids w. On a alors $\mathbf{xH}^\top + \mathbf{e} = \mathbf{s} + \mathbf{e}$. Cette dernière expression est une instance du problème générique d'un code linéaire aléatoire et l'on peut donc retrouver la valeur de $\mathbf{x}$.

Réciproquement, supposons que l'on sache résoudre le problème de recherche SD et considérons une instance du problème de décodage générique d'un code linéaire. Soit $\mathbf{G} \xleftarrow{\$} \mathbb{F}_q^{k\times n}$, $\mathbf{m} \xleftarrow{\$} \mathbb{F}_q^k$ et $\mathbf{e} \xleftarrow{\$} \mathcal{V}$ de poids w et $\mathbf{y} \in \mathcal{V}$ tel que $\mathbf{y} = \mathbf{mG} + \mathbf{e}$, déterminons $\mathbf{m}$.

Soit $\mathbf{H}$ une matrice de parité du code généré par $\mathbf{G}$. En multipliant à droite par $\mathbf{H}$, on obtient $\mathbf{yH} = \mathbf{eH}$ (car $\mathbf{GH} = 0$). En transposant, on a $\mathbf{H}^\top\mathbf{y}^\top = \mathbf{H}^\top\mathbf{e}^\top$. Comme $\mathbf{e}$ est de petit poids w, il s'agit d'une instance du problème de recherche SD. Ainsi, l'on peut retrouver la valeur de $\mathbf{e}$. On obtient alors $\mathbf{mG} = \mathbf{y} - \mathbf{e}$. On peut alors résoudre un système de n équations à k inconnues dans $\mathbb{F}_q$ dont on sait qu'il possède une solution par hypothèse de départ. On retrouve ainsi la valeur de $\mathbf{m}$. □

Le problème du décodage d'un code linéaire aléatoire a par ailleurs été prouvé $\mathcal{NP}$-complet [BMVT78], les problèmes suscités offrent donc une garantie très forte en terme de sécurité, puisqu'un attaquant capable de résoudre efficacement (en temps polynomial) l'un de ces problèmes serait alors capable de résoudre tous les problèmes $\mathcal{NP}$-difficiles en temps polynomial, ce qui résoudrait l'un des problèmes de mathématiques les plus importants de notre temps, à savoir si $\mathcal{P} = \mathcal{NP}$, l'un des sept problèmes du prix du millénaire.

2.3 La métrique rang

La métrique rang quant à elle est bien plus récente, apparue en même temps que les codes de Gabidulin en 1985 [Gab85] et la notion de codes $\mathbb{F}_{q^m}$ linéaire. Ce type de code est adapté à un modèle de canal de communication où les mots de code peuvent être représentés comme des matrices à coefficients dans un corps fini et où les erreurs surviennent par bloc sur les lignes ou les colonnes de ces matrices, comme c'est le cas par exemple pour le stockage des données sur bande magnétique [Loi07]. Ils exploitent la structure particulière

des q-polynômes introduits par Ore en 1933 [Ore33].

Soit m un entier naturel, on désignera par $\mathbb{F}_{q^m}$ le corps fini à q^m éléments. Notons $\mathcal{B} = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$ une base de $\mathbb{F}_{q^m}$ vu comme un espace vectoriel à m dimensions sur $\mathbb{F}_q$. $\mathcal{V}$ désignera désormais l'espace vectoriel à n dimensions à coefficients dans $\mathbb{F}_{q^m}$. Soit $P \in \mathbb{F}_q[X]$ un polynôme de degré n. L'espace vectoriel $\mathcal{V}$ peut alors être identifié avec l'anneau $\mathbb{F}_{q^m}[X]/\langle P \rangle$ où $\langle P \rangle$ désigne l'idéal de $\mathbb{F}_{q^m}[X]$ généré par P.

$$\begin{array}{rccc} \Psi & : & \mathbb{F}_{q^m}^n & \simeq & \mathbb{F}_{q^m}[X]/\langle P \rangle \\ & & \mathbf{x} = (x_0, \ldots, x_{n-1}) & \mapsto & \Psi(\mathbf{x}) = \sum_{i=0}^{n-1} x_i X^i. \end{array}$$

En effet, on peut construire un isomorphisme entre les éléments de $\mathcal{V}$ et les polynômes de $\mathbb{F}_{q^m}[X]/\langle P \rangle$. La somme de deux éléments $\mathbf{x}$ et $\mathbf{y}$ de $\mathcal{V}$ est définie naturellement par $\mathbf{x} + \mathbf{y} = (x_0 + y_0, \ldots, x_{n-1} + y_{n-1})$ et le produit est défini de façon similaire à la multiplication modulo P. Formellement, $\mathbf{x} \cdot \mathbf{y} \triangleq \Psi^{-1}[\Psi(\mathbf{x}) \cdot \Psi(\mathbf{y})]$. Pour plus de lisibilité, nous omettrons parfois le symbole Ψ et nous réfèrerons, selon le contexte, à l'espace vectoriel ou à l'espace des polynômes correspondants.

De la même façon qu'un élément de $\mathbb{F}_{q^m}$ peut être vu comme un vecteur à m coordonnées à coefficients dans $\mathbb{F}_q$, on peut associer à tout vecteur $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{V}$ une matrice $\mathbf{M}_\mathbf{x} \in \mathbb{F}_q^{m \times n}$ en exprimant chacune de ses coordonnées dans la base $\mathcal{B}$. Ainsi, $\mathbf{M}_\mathbf{x} = (x_{ij})$ avec, pour tout $j \in [\![1, n]\!]$, $x_j = \sum_{i=1}^{m} x_{i,j} \boldsymbol{\beta}_i$.

$$\begin{array}{rccc} \mathbf{M} & : & \mathbb{F}_{q^m}^n & \simeq & \mathbb{F}_q^{m \times n} \\ & & \mathbf{x} = (x_0, \ldots, x_{n-1}) & \mapsto & \mathbf{M}_\mathbf{x} = \begin{pmatrix} x_{1,0} & \cdots & x_{1,n-1} \\ x_{2,0} & \cdots & x_{2,n-1} \\ \vdots & & \vdots \\ x_{m,0} & \cdots & x_{m,n-1} \end{pmatrix} \begin{matrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_m. \end{matrix} \end{array}$$

Une notion centrale en métrique rang est la notion de support d'un vecteur, que l'on peut simplement décrire comme étant l'espace généré par les coordonnées de ce vecteur.

Définition 13 (Support d'un vecteur). *Soit* $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{V}$. *Le support de* $\mathbf{x}$, *noté* $\mathsf{Supp}(\mathbf{x})$, *est le* $\mathbb{F}_q$ *sous-espace vectoriel de* $\mathbb{F}_{q^m}$ *généré par les coordonnées de* $\mathbf{x}$, *i.e.* $\mathsf{Supp}(\mathbf{x}) = \langle x_1, \ldots, x_n \rangle_{\mathbb{F}_q}$.

Ce qui était appelé le poids du vecteur en métrique de Hamming sera désigné par rang du vecteur en métrique rang.

Définition 14 (Rang d'un vecteur). *Soit* $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{V}$. *Le rang de* $\mathbf{x}$, *noté* $\|\mathbf{x}\|$, *est égal à la dimension du support associé à* $\mathbf{x}$.

Proposition 4. *Le rang d'un vecteur est égal au rang de sa matrice associée.*

Définition 15 (Métrique rang). *Soit* $\mathbf{x}, \mathbf{y} \in \mathcal{V}$. *Soit* d_R *l'application définie par :*

$$d_R(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

Proposition 5. *L'application* d_R *définie précédemment est une distance sur* $\mathcal{V}$.

Définition 16 (Code $\mathbb{F}_{q^m}$-linéaire). *Un code* $\mathbb{F}_{q^m}$*-linéaire* $\mathcal{C}$ *de dimension* k *et de longueur* n *est un sous-espace de dimension* k *de* $\mathbb{F}_{q^m}^n$ *et se note* $[n, k]_{q^m}$ *ou simplement* $[n, k]$ *lorsqu'il n'y a pas d'ambiguïté.* $\mathcal{C}$ *peut être représenté de deux façons équivalentes :*

- ◇ *En utilisant une matrice génératrice* $\mathbf{G} \in \mathbb{F}_{q^m}^{k \times n}$

$$\mathcal{C} = \{\mathbf{x}\mathbf{G} \mid \mathbf{x} \in \mathbb{F}_{q^m}^k\}.$$

- ◇ *En utilisant une matrice de parité* $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k) \times n}$. *Les lignes de* $\mathbf{H}$ *déterminent un système d'équations que doivent vérifier les éléments de* $\mathcal{C}$

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_{q^m}^n \mid \mathbf{H}\mathbf{x}^\top = 0\}.$$

Définition 17 (Borne de singleton en métrique rang). *Un code* $[n, k, d]_{q^m}$ *vérifie toujours*

$$| \mathcal{C} | \leqslant q^{\min(m(n-d+1), n(m-d+1))}.$$

Proposition 6. *Dans le cas où* $n \leqslant m$ *et* $d > 1$, *que nous considérerons toujours, l'inégalité peut alors s'écrire comme en métrique de Hamming, à savoir :*

$$d + k \leqslant n + 1.$$

Démonstration. On a : $| \mathcal{C} | = q^{km}$. La borne de singleton vérifie donc $q^{km} \leqslant$

$q^{\min(m(n-d+1),n(m-d+1))}$. Les exposants étant positifs, cette inégalité est équivalente à $km \leqslant \min(m(n-d+1), n(m-d+1))$.

Prouvons dans un premier temps que $\min(m(n-d+1), n(m-d+1)) = m(n-d+1)$. On a $n \leqslant m$ et $d > 1$, donc $(1-d) < 0$.

On a les équivalences suivantes :

$$\begin{aligned} m \geqslant n &\Leftrightarrow m(1-d) \leqslant n(1-d) \\ &\Leftrightarrow mn + m(1-d) \leqslant mn + n(1-d) \\ &\Leftrightarrow m(n-d+1) \leqslant n(m-d+1) \\ &\Leftrightarrow min(m(n-d+1), n(m-d+1)) = m(n-d+1). \end{aligned}$$

On a donc $km \leqslant m(n-d+1)$, par suite $k \leqslant n-d+1$, d'où $d+k \leqslant n+1$. □

Définition 18 (Code MRD)**.** *Un code atteignant la borne de singleton en métrique rang, i.e. vérifiant $k = n-d+1$ est appelé MRD (Maximum Rank Distance separable).*

Définition 19 (Coefficient binomial de Gauss)**.** *Le coefficient binomial de Gauss, noté $\binom{n}{k}_q$ avec q une puissance d'un nombre premier, correspond au nombre de sous-espaces vectoriels de dimension k dans un espace vectoriel de dimension n à coefficient dans le corps fini $\mathbb{F}_q$ à q éléments.*

Proposition 7.

$$\binom{n}{k}_q = \frac{(1-q^n)(1-q^{n-1})\dots(1-q^{n-k-1})}{(1-q)(1-q^2)\dots(1-q^k)}.$$

Notation 1.

◇ $\mathcal{S}_w(\mathcal{V}) = \{\mathbf{x} \in \mathbf{v} \mid \|\mathbf{x}\| = w\}$ *(Sphère centrée en l'origine de rayon w dans $\mathcal{V}$)*

◇ $\mathcal{B}_w(\mathcal{V}) = \bigcup_{i=0}^{w} \mathcal{S}_w(\mathcal{V})$ *(Boule centrée en l'origine de rayon w dans $\mathcal{V}$).*

Proposition 8. *Le nombre d'éléments dans une sphère ou dans une boule est donné respectivement par :*

◇ $\mid \mathcal{S}_w(\mathcal{V}) \mid = \left(\prod_{i=0}^{w-1}(q^n - q^i)\right) \cdot \binom{m}{w}_q$

◇ $\mid \mathcal{B}_w(\mathcal{V}) \mid = \sum_{i=0}^{w} \mid \mathcal{S}_w(\mathcal{V}) \mid .$

Proposition 9 (Borne d'empilement des sphères)**.** *Soit $\mathcal{C}$ un code (n, M, d) sur $\mathcal{V}$. Soit $t \triangleq \lfloor \frac{n-k}{2} \rfloor$. Alors nécessairement $M \times \mid \mathcal{B}_w(\mathcal{V}) \mid \leqslant q^{mn}$.*

La proposition précédente traduit simplement le fait que l'union de toutes les boules dont les centres sont les mots de code (à noter que le code n'a pas besoin d'être linéaire) et de rayon strictement inférieur à la distance minimal du code ne peut pas contenir davantage d'éléments que l'espace $\mathcal{V}$ n'en contient.

La question de l'existence de codes de paramètres fixés est alors donnée par un équivalent de la borne de Gilbert-Varshamov pour la métrique rang. Cette borne est très importante en théorie des codes, c'est notamment cette borne qui détermine la difficulté des problèmes de décodage par syndrome.

Proposition 10 (Borne de Gilbert-Varshamov en métrique rang)**.** *Si $M \times \mid \mathcal{B}_w(\mathcal{V}) \mid < q^{mn}$, alors il existe un code $(n, M+1, d)$ sur $\mathbb{F}_{q^m}$.*

Pour les preuves de ces propositions, le lecteur pourra se référer à [Loi07].

Enfin, introduisons une dernière notation utile pour décrire des vecteurs partageant le même support :

$$\mathfrak{S}_w^n(\mathcal{V}) = \left\{ (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{V}^n \;\middle|\; \begin{array}{l} \exists E \subset \mathcal{V} \text{ avec } \dim(E) = w \\ \text{tel que } \forall i \in [\![1, n]\!],\ \mathsf{Supp}(\mathbf{x}_i) = E \end{array} \right\}.$$

2.4 Codes correcteurs en métrique rang

Les codes de Gabidulin sont les premiers codes correcteurs basés sur la métrique rang et sont apparus en 1985 [Gab85]. Ils exploitent une classe particulière de polynômes appelés q-polynômes introduits par Ore en 1933 [Ore33]. La structure des q-polynômes est intrinsèquement liée à la métrique rang et le décodage des codes de Gabidulin se ramène au problème d'interpolation des q-polynômes. Nous n'expliciterons pas toutes ces notions ici, le lecteur curieux pourra se référer à [Loi07, ALR18].

2.4.1 Codes de Gabidulin

Notation 2. $[i] \triangleq q^i$.

Définition 20 (q-polynômes)**.** *Un q-polynôme de q-degré r sur $\mathbb{F}_{q^m}$ est un polynôme de la forme $P(X) = \sum_{i=0}^{r} p_i X^{[i]}$ avec, pour tout $i \in [\![0, r]\!]$, $p_i \in \mathbb{F}_{q^m}$ et $p_r \neq 0$.*

Définition 21 (Codes de Gabidulin). *Soit $k, n, m \in \mathbb{N}$ tels que $k \leqslant n \leqslant m$. Soit $\mathbf{g} = (g_1, \ldots, g_n) \in \mathcal{V}$ une famille linéairement indépendante de vecteurs sur $\mathbb{F}_q^m$ à coefficients dans $\mathbb{F}_q$. Un code de Gabidulin est un code $[n,k]_{q^m}$ de matrice génératrice :*

$$Gab_{\mathbf{g}} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}.$$

Le décodage des codes de Gabidulin est déterministe, et un code de Gabidulin est toujours MRD, ce qui signifie que la distance minimale du code est toujours égale à $n-k+1$ et par suite que sa capacité de correction est de $\lfloor \frac{n-k}{2} \rfloor$.

2.4.2 Codes quasi-cycliques

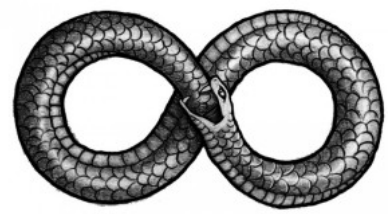

FIGURE 2.1 – Le symbole Ouroboros

L'Ouroboros est un symbole très ancien d'un serpent ou d'un dragon qui se mord la queue. On le retrouve dans de nombreuses cultures et il a de multiples interprétations. Sur cette image, il représente bien l'idée d'un cycle, d'un éternel recommencement. D'ailleurs sa forme n'est pas sans rappeler le symbole mathématique de l'infini.

Dans cette partie, on considérera le polynôme $P = X^n - 1$, ainsi $\mathcal{V}$ sera identifié avec l'anneau quotient $\mathbb{F}_{q^m}[X]/\langle X^n - 1 \rangle$. On définit alors le produit dans $\mathcal{V}$ de façon à ce qu'il soit isomorphe au produit défini dans l'anneau quotient des polynômes.

Définition 22 (Produit dans $\mathcal{V}$)**.**

$$\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}, \mathbf{u} \cdot \mathbf{v} \triangleq \mathbf{w} \text{ tel que } \forall k \in [\![0, n-1]\!],\ w_k = \sum_{\substack{i+j \equiv k \mod n \\ i,j \in [\![0,n-1]\!]}} u_i v_j.$$

Proposition 11. *$\mathcal{V}$ muni de l'addition et du produit défini précédemment est isomorphe à l'anneau quotient $(\mathbb{F}_{q^m}[X]/\langle X^n - 1 \rangle, +, \times)$.*

Le produit entre deux vecteurs peut alors être assimilé à un produit vecteur-matrice.

Proposition 12.

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{V},\ \mathbf{u} \cdot \mathbf{v} = (u_0, \ldots, u_{n-1}) \cdot \begin{pmatrix} v_0 & v_1 & \ldots & v_{n-1} \\ v_{n-1} & v_0 & \ldots & v_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ v_1 & v_2 & \ldots & v_0 \end{pmatrix}.$$

Définition 23 (Matrice circulante)**.** *Soit* $\mathbf{v} \in \mathcal{V}$*. La matrice circulante induite par* $\mathbf{v}$ *est la matrice, notée* $\mathsf{rot}(\mathbf{v})$ (rot *comme rotation), définie par :*

$$\mathsf{rot}(\mathbf{v}) = \begin{pmatrix} v_0 & v_1 & \ldots & v_{n-1} \\ v_{n-1} & v_0 & \ldots & v_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ v_1 & v_2 & \ldots & v_0 \end{pmatrix}.$$

Corollaire 1. *Pour tout* $\mathbf{u}, \mathbf{v} \in \mathcal{V},\ \mathbf{u} \cdot \mathbf{v} = \mathbf{u} \cdot \mathsf{rot}(\mathbf{v}) = \mathbf{v} \cdot \mathsf{rot}(\mathbf{u}) = \mathbf{v} \cdot \mathbf{u}$*.*

Définition 24 (Code quasi-cyclique)**.** *Soit* $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_s) \in \mathbb{F}_{q^m}^{sn}$*, vu comme s blocs successifs de taille n où chaque* c_i *est considéré comme un polynôme de* $\mathbb{F}_{q^m}[X]/\langle X^n - 1\rangle$*. Un code* $[sn, k]_{q^m}$ *linéaire* $\mathcal{C}$ *est dit quasi-cyclique d'index s si, pour tout* $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_s) \in \mathcal{C},\ (X \cdot \mathbf{c}_1, \ldots, X \cdot \mathbf{c}_s) \in \mathcal{C}$*.*

Proposition 13 (Matrice génératrice d'un code quasi-cyclique)**.** *Toute matrice génératrice d'un code* $[sn, n]$ *quasi-cyclique d'index s et de taux* $\frac{1}{s}$ *est de la forme :*

$$\mathbf{G} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \ldots & \mathbf{A}_s \end{pmatrix}$$

où, pour tout $i \in [\![1, s]\!]$*,* $\mathbf{A}_i$ *est une matrice circulante de taille* $n \times n$*.*

Définition 25 (Code quasi-cyclique sous forme systématique)**.** *Un code* $[sn, n]$ *est dit quasi-cyclique sous forme systématique d'index s et de taux* $\frac{1}{s}$ *si sa matrice de*

parité peut s'écrire sous la forme :

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & 0 & \dots & 0 & \mathbf{A}_1 \\ 0 & \mathbf{I}_n & \ddots & \vdots & \mathbf{A}_2 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & \mathbf{I}_n & \mathbf{A}_{s-1} \end{pmatrix}$$

où $\mathbf{A}_1, \dots, \mathbf{A}_{s-1}$ *sont des matrices circulantes de taille* $n \times n$.

Notation 3. *On notera* $\mathcal{QC}_s^{syst}(\mathbb{F}_{q^m})$ *l'ensemble des matrices de parité sous forme systématique de codes* $[sn, n]$ *quasi-cycliques d'index* s *et de taux* $\frac{1}{s}$ *à coefficients dans* $\mathbb{F}_{q^m}$.

2.4.3 Codes idéaux

Les codes idéaux peuvent être vus comme une généralisation des codes quasi-cycliques au cas où P est choisi quelconque de degré n dans $\mathbb{F}_q[X]$. $\mathcal{V}$ sera désormais identifié avec l'anneau quotient $\mathbb{F}_{q^m}[X]/\langle P \rangle$. Nous définissons à nouveau le produit dans $\mathcal{V}$ de façon à ce qu'il soit isomorphe au produit défini dans l'anneau quotient des polynômes.

Définition 26 (Produit dans $\mathcal{V}$)**.** *Pour tout* $\mathbf{u}, \mathbf{v} \in \mathcal{V}$, $\mathbf{u} \cdot \mathbf{v} \triangleq u(X) \cdot v(X) \mod P$.

Proposition 14. *Pour tout* $\mathbf{u}, \mathbf{v} \in \mathcal{V}$,

$$\begin{aligned} \mathbf{u} \cdot \mathbf{v} &= u(X) \cdot v(X) \qquad \mod P \\ &= \sum_{i=0}^{n-1} u_i X^i \cdot v(X) \quad \mod P \\ &= \sum_{i=0}^{n-1} u_i [X^i v(X)] \quad \mod P \\ &= \mathbf{u} \cdot \begin{pmatrix} v(X) & \mod P \\ X \cdot v(X) & \mod P \\ \vdots & \\ X^{n-1} \cdot v(X) & \mod P \end{pmatrix}. \end{aligned}$$

La proposition précédente montre que l'on peut substituer un vecteur par une matrice carrée $n \times n$ ayant une forme particulière. Une telle matrice est appelée la matrice idéale induite par $\mathbf{v}$.

Définition 27 (Matrice idéale)**.** *Soit* $\mathbf{v} \in \mathcal{V}$*, la matrice idéale induite par* $\mathbf{v}$*, notée* $\mathcal{I}(\mathbf{v})$*, est la matrice :*

$$\mathcal{I}(\mathbf{v}) = \begin{pmatrix} v(X) & \mod P \\ X \cdot v(X) & \mod P \\ \vdots & \\ X^{n-1} \cdot v(X) & \mod P \end{pmatrix}.$$

Définition 28 (Code idéal)**.** *Soit* $\mathbf{c} = (c_1, \ldots, c_s) \in \mathbb{F}_{q^m}^{sn}$*, vu comme s blocs successifs de taille n où chaque* c_i *est considéré comme un polynôme de* $\mathbb{F}_{q^m}[X]/\langle P \rangle$*. Un code* $[sn, k]_{q^m}$ $\mathcal{C}$ *est dit idéal d'index s si, pour tout* $\mathbf{c} = (c_1, \ldots, c_s) \in \mathcal{C}$*,* $(X \cdot c_1 \mod P, \ldots, X \cdot c_s \mod P) \in \mathcal{C}$*.*

Proposition 15 (Matrice génératrice d'un code idéal)**.** *Toute matrice génératrice d'un code* $[sn, n]_{q^m}$ *idéal d'index s est de la forme :*

$$\mathbf{G} = \begin{pmatrix} \mathcal{I}(\mathbf{v}_1) & \mathcal{I}(\mathbf{v}_2) & \ldots & \mathcal{I}(\mathbf{v}_s) \end{pmatrix}.$$

Proposition 16 (Code idéal sous forme systématique)**.** *Un code* $[sn, n]_{q^m}$ *idéal d'index s sous forme systématique est un code idéal dont la matrice de parité peut s'écrire sous la forme :*

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & 0 & \ldots & 0 & \mathcal{I}(\mathbf{a}_1) \\ 0 & \mathbf{I}_n & \ddots & \vdots & \mathcal{I}(\mathbf{a}_2) \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \ldots & 0 & \mathbf{I}_n & \mathcal{I}(\mathbf{a}_{s-1}) \end{pmatrix}.$$

où, pour tout $i \in [\![1, s-1]\!]$*,* $\mathbf{a}_i \in \mathcal{V}$*.*

Notation 4. *On notera* $\mathcal{I}_s^{syst}(\mathbb{F}_{q^m})$ *l'ensemble des matrices de parité sous forme systématique de codes* $[sn, n]$ *idéaux d'index s et de taux* $1/s$ *à coefficients dans* $\mathbb{F}_{q^m}$*.*

2.4.4 Codes LRPC

Les codes LRPC sont quant à eux apparus en 2013 [GMRZ13]. A la différence des codes de Gabidulin, le décodage est probabiliste, mais la probabilité d'erreur dans le décodage peut être rendue aussi petite que nécessaire.

Définition 29 (Codes LRPC). *Un code LRPC (Low Rank Parity Check codes) de rang d, longueur n et de dimension k sur $\mathbb{F}_{q^m}$ est un code dont la matrice de parité $\mathbf{H} = (h_{i,j}) \in \mathbb{F}_{q^m}^{(n-k)\times n}$ est telle que le sous-espace vectoriel engendré par ses coordonnées a une dimension majorée par d.*

2.5 Les problèmes difficiles basés sur la métrique rang

En métrique rang, la sécurité repose généralement sur le problème RSD, adaptation du problème SD à la métrique rang, ainsi que sur ses dérivés. Comme mentionné précédemment, le problème SD est prouvé $\mathcal{NP}$-complet car équivalent au problème de décodage d'un code linéaire aléatoire [BMVT78]. Le problème RSD quant à lui n'est pas prouvé $\mathcal{NP}$-complet, mais une réduction probabiliste du problème RSD vers le problème SD a été démontré dans [GZ16].

Définition 30 (Distribution RSD). *Etant donné des entiers naturels n, k et w, la distribution* $\mathsf{RSD}(n,k,w)$ *est la distribution obtenue en choisissant $\mathbf{H} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n}$ et $\mathbf{x} \xleftarrow{\$} \mathcal{S}_w(\mathcal{V})$, et renvoyant le doublet $(\mathbf{H}, (\mathbf{H}\mathbf{x}^\top)^\top)$.*

Définition 31 (Le problème RSD). *Etant donné $(\mathbf{H}, \mathbf{y}) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n-k}$ issu de la distribution* RSD*, le problème* $\mathsf{RSD}(n,k,w)$ *(Rank Syndrome Decoding problem) consiste à trouver $\mathbf{x} \in \mathcal{S}_w(\mathcal{V})$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$.*

Définition 32 (Version décisionnelle du problème RSD). *Etant donné $(\mathbf{H}, \mathbf{y}) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n-k}$, la version décisionnelle du problème* RSD *consiste à pouvoir déterminer, avec un avantage non négligeable, si $(\mathbf{H}, \mathbf{y})$ a été généré à partir de la distribution* $\mathsf{RSD}(n,k,w)$ *ou de la distribution uniforme sur $\mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n-k}$.*

Ces problèmes existent en version quasi-cyclique. Ceux-ci permettent la construction de protocoles avec des tailles de clés plus petites. En effet, en utilisant des matrices de parité sous forme systématiques de codes quasi-cycliques, le nombre de coefficients à stocker est considérablement réduit par rapport à une matrice de parité sous forme systématique d'un code linéaire aléatoire.

Définition 33 (Distribution s-RQCSD). *Etant donné des entiers naturels n, w, s, la distribution* s-$\mathsf{RQCSD}(n,w)$ *est la distribution obtenue en choisissant $\mathbf{H} \xleftarrow{\$} \mathcal{QC}_s^{syst}(\mathbb{F}_{q^m})$ et $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \xleftarrow{\$} \mathfrak{S}_w^s(\mathcal{V})$ et renvoyant le doublet $(\mathbf{H}, (\mathbf{H}\mathbf{x}^\top)^\top)$.*

Définition 34 (Problème s-RQCSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{QC}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$ *issu de la distribution* s-RQCSD*, le problème* s-RQCSD(n, w) *consiste à trouver* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathfrak{S}_w^s(\mathcal{V})$ *tels que* $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$*.*

Définition 35 (Version décisionnelle du problème s-RQCSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{QC}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$*, la version décisionnelle du problème* s-RQCSD *consiste à pouvoir déterminer, avec une probabilité non négligeable, si* $(\mathbf{H}, \mathbf{y})$ *a été généré à partir de la distribution* s-RQCSD(n, w) *ou de la distribution uniforme sur* $\mathcal{QC}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$*.*

Enfin, ceux-ci peuvent être généralisés au cas de matrices de parité de code idéaux.

Définition 36 (Distribution s-IRSD)**.** *Etant donné des entiers naturels* n*,* w*,* s *et un polynôme* $P \in \mathbb{F}_q[X]$ *de degré* n*, la distribution* s-IRSD(n, w) *est la distribution obtenue en choisissant* $\mathbf{H} \xleftarrow{\$} \mathcal{I}_s^{syst}(\mathbb{F}_{q^m})$ *et* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \xleftarrow{\$} \mathfrak{S}_w^s(\mathcal{V})$ *et renvoyant le doublet* $(\mathbf{H}, (\mathbf{H}\mathbf{x}^\top)^\top)$*.*

Définition 37 (Problème s-IRSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{I}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$ *issu de la distribution* s-IRSD*, le problème* s-IRSD(n, w) *consiste à trouver* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathfrak{S}_w^s(\mathcal{V})$ *tels que* $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$*.*

Définition 38 (Version décisionnelle du problème s-IRSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{I}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$*, la version décisionnelle du problème* s-IRSD *consiste à pouvoir déterminer, avec une probabilité non négligeable, si* $(\mathbf{H}, \mathbf{y})$ *a été généré à partir de la distribution* s-IRSD(n, w) *ou de la distribution uniforme sur* $\mathcal{I}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$*.*

Bien qu'il n'existe pas de résultat connu sur la difficulté de ces variantes, aucune attaque connue n'a encore su tirer profit de la structure quasi-cyclique ou idéale utilisée tant que le polynôme P est choisi irréductible.

Définition 39 (Problème s-IRSR)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{I}_s^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{sn}$ *issu de la distribution* s-IRSD*, le problème* s-IRSR *consiste à déterminer un espace* E *de dimension* w *tel que* $\mathbf{y}^\top = \mathbf{H}\mathbf{x}^\top$ *où* $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathfrak{S}_w^s(\mathcal{V})$ *et* E *est le support commun à* $\mathbf{x}_1, \ldots, \mathbf{x}_n$*.*

Proposition 17. *Les problèmes* s-IRSD *et* s-IRSR *sont équivalents.*

Démonstration. Supposons que l'on soit capable de résoudre le problème s-IRSD et soit $(\mathbf{H}, \mathbf{y})$ une instance du problème s-IRSR. Nous sommes en mesure de déterminer $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathfrak{S}_w^s(\mathcal{V})$ tel que $\mathbf{y}^\top = \mathbf{H}\mathbf{x}^\top$. Il suffit donc de calculer

le support commun E de $\mathbf{x}_1, \ldots, \mathbf{x}_s$ pour résoudre l'instance du problème s-IRSR considérée.

Réciproquement, supposons que l'on soit capable de résoudre le problème s-IRSR et soit $(\mathbf{H}, \mathbf{y})$ une instance du problème s-IRSD. Nous sommes en mesure de déterminer un espace E de dimension w tel que $\mathbf{y}^\top = \mathbf{H}\mathbf{x}^\top$ où $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \in \mathfrak{S}_w^s(\mathcal{V})$. On a :

$$\begin{pmatrix} \mathbf{I}_n & 0 & \ldots & 0 & \mathcal{I}(\mathbf{a}_1) \\ 0 & \mathbf{I}_n & \ddots & \vdots & \mathcal{I}(\mathbf{a}_2) \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \ldots & 0 & \mathbf{I}_n & \mathcal{I}(\mathbf{a}_{s-1}) \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_s \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_s \end{pmatrix}$$

Chaque $\mathbf{x}_1, \ldots, \mathbf{x}_s$ est un vecteur de $\mathcal{V}$, on peut donc exprimer ces vecteurs sous la forme $\mathbf{x}_1 = (x_{11}, \ldots, x_{1n}), \ldots, \mathbf{x}_s = (x_{s1}, \ldots, x_{sn})$ avec, pour tout $(i, j) \in [\![1, s]\!] \times [\![1, n]\!]$, $x_{ij} \in \mathbb{F}_{q^m}$. Soit $\{\mathbf{e}_1, \ldots, \mathbf{e}_w\}$ une base de E. On peut exprimer chaque x_{ij} dans la base E.

$$\forall (i, j) \in [\![1, s]\!] \times [\![1, n]\!],\ x_{ij} = \sum_{k=1}^{w} \lambda_{ijk} \cdot \mathbf{e}_k \text{ avec } \lambda_{ijk} \in \mathbb{F}_q$$

On peut alors se ramener à la résolution d'un système à sn équations sur $\mathbb{F}_{q^m}$, soit snm équations dans $\mathbb{F}_q$ à snw inconnues dans $\mathbb{F}_q$. Le système a ainsi plus d'équations linéairement indépendantes que d'inconnues et possède au moins une solution car $(\mathbf{H}, \mathbf{y})$ est issu de la distribution s-IRSD. On peut ainsi retrouver $\mathbf{x}$ en résolvant le système précédent. □

Les deux approches principales utilisées pour résoudre les problèmes RSD sont de type combinatoire ou algébrique. Les attaques combinatoires ont longtemps été les plus efficaces [GRS15, AGHT18], mais de récentes améliorations ont été découvertes au niveau des attaques algébriques [BBC$^+$20a, BBC$^+$20b]. Ces nouvelles attaques ont forcé le schéma RQC à définir de nouvelles versions du problème IRSD, appelées non-homogènes (NH), afin de ne pas trop augmenter la taille des paramètres pour les niveaux de sécurité visés. Nous ne présenterons que la version des problèmes correspondant aux codes $[3n, n]_{q^m}$ comme cela a été fait pour RQC (spécification d'avril 2020 [MAB$^+$20b]).

Définition 40 (Distribution 3-NHIRSD)**.** *Etant donné des entiers naturels n, w_1, w_2 et un polynôme $P \in \mathbb{F}_q[X]$ de degré n, la distribution* $\mathsf{NHIRSD}(n, w_1, w_2)$ *est la distribution obtenue en choisissant* $\mathbf{H} \xleftarrow{\$} \mathcal{I}_3^{syst}(\mathbb{F}_{q^m})$ *et* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \xleftarrow{\$} \mathcal{V}^3$ *tels que* $\|\mathbf{x}_1\| = \|\mathbf{x}_3\| = w_1$, $\|\mathbf{x}_2\| = w_1 + w_2$, $\mathsf{Supp}(\mathbf{x}_1) = \mathsf{Supp}(\mathbf{x}_3) \subset \mathsf{Supp}(\mathbf{x}_2)$ *et renvoyant le doublet* $(\mathbf{H}, (\mathbf{H}\mathbf{x}^\top)^\top)$*.*

Définition 41 (Problème 3-NHIRSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{I}_3^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{3n}$ *issu de la distribution* 3-NHIRSD*, le problème* $3\text{-}\mathsf{NHIRSD}(n, w_1, w_2)$ *consiste à trouver* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{V}^3$ *tels que* $\|\mathbf{x}_1\| = \|\mathbf{x}_3\| = w_1$, $\|\mathbf{x}_2\| = w_1 + w_2$, $\mathsf{Supp}(\mathbf{x}_1) = \mathsf{Supp}(\mathbf{x}_3) \subset \mathsf{Supp}(\mathbf{x}_2)$ *et* $\mathbf{H}\mathbf{x}^\top = \mathbf{y}^\top$*.*

Proposition 18. *Etant donné des entiers naturels n, w_1, w_2, et $P \in \mathbb{F}_q[X]$ un polynôme irréductible de degré n. Le problème* $3\text{-}\mathsf{IRSD}(n, w_1)$ *se réduit au problème* $3\text{-}\mathsf{NHIRSD}(n, w_1, w_2)$*.*

Démonstration. Supposons que l'on dispose d'un algorithme $\mathcal{A}(n, w_1, w_2)$ capable de résoudre en temps polynomial n'importe quelle instance de paramètre n, w_1, w_2 du problème 3-NHIRSD. Montrons que l'on peut alors résoudre n'importe quelle instance du problème 3-IRSD en temps polynomial. Soit n', w_1' des entiers naturels et considérons l'instance 3-IRSD correspondante à ces paramètres. Alors, en utilisant l'algorithme $\mathcal{A}$ avec les paramètres $(n', w_1', 0)$, on peut résoudre en temps polynomial l'instance du problème 3-IRSD donné. □

Définition 42 (Version décisionnelle du problème 3-NHIRSD)**.** *Etant donné* $(\mathbf{H}, \mathbf{y}) \in \mathcal{I}_3^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{3n}$*, la version décisionnelle du problème* 3-NHIRSD *consiste à pouvoir déterminer, avec une probabilité non négligeable, si* $(\mathbf{H}, \mathbf{y})$ *a été généré à partir de la distribution* $3\text{-}\mathsf{NHIRSD}(n, w_1, w_2)$ *ou de la distribution uniforme sur* $\mathcal{I}_3^{syst}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{3n}$*.*

2.6 Les schémas RQC et ROLLO

Le schéma RQC

RQC (Rank Quasi-Cyclic) est un schéma de chiffrement à clé publique soumis à l'appel à standardisation du NIST. Comme son nom l'indique, sa sécurité était basée initialement sur les versions quasi-cycliques du problème RSD, et plus précisément sur la version décisionnelle du problème RQCSD. Il a ensuite été généralisé au cas des codes idéaux. Puis, avec l'apparition récente de nouvelles

attaques sur la métrique rang [BBC+20b], il utilise désormais les versions non-homogènes de ces problèmes afin de conserver des jeux de paramètres attractifs.

RQC est un schéma de chiffrement sécurisé contre les attaques à chiffrés choisis (IND-CCA2), il se base sur une version IND-CPA noté RQC.PKE (voir figure 2.2) et utilise la transformation HHK [HHK17] pour rendre le schéma IND-CCA2. Il exploite une approche novatrice introduite par Alekhnovitch [Ale03] permettant ainsi de faire en sorte que la sécurité du schéma soit réduite au problème du décodage d'un code linéaire aléatoire. Cette approche permet de ne plus faire reposer la sécurité sur des hypothèses d'indistinguabilité de la famille de codes utilisée par rapport à un code aléatoire [AMBD+18].

Nous décrirons ici RQC en version idéale, sans utiliser les versions non-homogènes des problèmes sous-jacents afin de ne pas trop alourdir la description du schéma et les descriptions et preuves du Hash Proof System (HPS) basé sur RQC que nous présenterons en section 3.7. RQC utilise une matrice génératrice d'un code de Gabidulin $\mathsf{Gab}_{\mathbf{g}}$ et un code aléatoire $[2n, n]_{q^m}$ de matrice de parité $(\mathbf{I_n} \ \ \mathcal{I}(\mathbf{h}))$ avec $\mathbf{h}$ un vecteur aléatoire de $\mathcal{V}$.

- $\diamond$ $\mathsf{Setup}(1^{\mathfrak{K}})$:
  Etant donné un paramètre de sécurité $\mathfrak{K}$, générer et renvoyer les paramètres globaux $\mathsf{param} = (n, k, \delta, w, w_r, P)$ où $P \in \mathbb{F}_q[X]$ est un polynôme irréductible de degré n.
- $\diamond$ $\mathsf{KeyGen}(\mathsf{param})$:
  Générer $\mathbf{h} \xleftarrow{\$} \mathcal{V}$, $\mathbf{g} \in \mathcal{S}_n(\mathcal{V})$, $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathfrak{S}^2_w(\mathcal{V})$. Calculer $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$ et renvoyer $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$ et $\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{s})$.
- $\diamond$ $\mathsf{Encrypt}(\mathsf{pk}, \mathbf{m}, \theta)$:
  Utiliser la graine θ pour générer $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \xleftarrow{\$} \mathfrak{S}^3_{w_r}(\mathcal{V})$. Générer $\mathbf{c}_1 = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ et $\mathbf{c}_2 = \mathbf{m}\mathsf{Gab}_{\mathbf{g}} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{r}_3$ et renvoyer $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$.
- $\diamond$ $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$:
  Renvoyer $\mathcal{C}.\mathsf{Decode}(\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1)$.

FIGURE 2.2 – Description de RQC.PKE [AMBD+18]

Au niveau du déchiffrement, la connaissance de la clé secrète sk permet de cal-

culer l'expression $\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1$. On a alors :

$$\begin{aligned}\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1 &= (\mathbf{m}\mathrm{Gab}_{\mathbf{g}} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{r}_3) - \mathbf{y}(\mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2) \\ &= (\mathbf{m}\mathrm{Gab}_{\mathbf{g}} + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{h} \cdot \mathbf{y} \cdot \mathbf{r}_2 + \mathbf{r}_3) - \mathbf{y}(\mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2) \\ &= \mathbf{m}\mathrm{Gab}_{\mathbf{g}} + \mathbf{x}\mathbf{r}_2 + \mathbf{r}_3 - \mathbf{y}\mathbf{r}_1\end{aligned}$$

Le jeu de paramètres du chiffrement est alors choisi de sorte que le rang de l'expression $\mathbf{x}\mathbf{r}_2 + \mathbf{r}_3 - \mathbf{y}\mathbf{r}_1$ soit inférieur à la capacité de correction du code de Gabidulin, à savoir $\lfloor \frac{n-k}{2} \rfloor$, permettant ainsi de retrouver le message initial $\mathbf{m}$.

Les schémas ROLLO

ROLLO [MAB+20a] quant à lui est la réunion de trois cryptosystèmes, à savoir LAKE [ABD+17a], LOCKER [ABD+17b] et Ouroboros-R [MAB+17]. Ces trois schémas sont tous basés sur la métrique rang, partagent le même algorithme de déchiffrement pour le décodage des codes LRPC et sont très similaires. L'idée principale derrière ces trois schémas est d'utiliser des codes LRPC à la place des codes de Gabidulin, permettant d'aboutir à un déchiffrement plus rapide ainsi qu'à des tailles de clés réduites comparé au schéma RQC. L'algorithme de déchiffrement est par conséquent différent de celui utilisé dans RQC, il s'agit de l'algorithme RSR (Rank Support Recovery). Comme son nom l'indique, cet algorithme retrouve, dans un premier temps, le support de l'erreur. Une fois que le support est connu, les coordonnées de l'erreur peuvent être retrouvées en résolvant un système d'équations linéaires. Notons qu'à la différence de RQC, le déchiffrement n'est plus déterministe, le taux d'erreur de déchiffrement (DFR, Decryption Failure Rate) peut être majoré et ainsi rendu aussi bas que souhaité en fonction du choix du jeu de paramètres.

Le schéma LAKE

LAKE, correspondant à ROLLO-I, est un protocole d'échange de clés (KEM, Key Encapsulation Mechanism) sécurisé contre les attaques à clairs choisis (IND-CPA) soumis durant le processus de standardisation du NIST. Ce schéma permet à deux personnes de partager une clé commune à la fin du protocole, clé éphémère permettant de communiquer par la suite de manière sécurisée en utilisant un chiffrement symétrique. La description du schéma est donnée sur la figure 2.3.

- ◇ $\mathsf{Setup}(1^{\mathfrak{K}})$: Etant donné $\mathfrak{K}$, générer et renvoyer les paramètres globaux $\mathsf{param} = (n, m, d, r, P)$ où $P \in \mathbb{F}_q[X]$ désigne un polynôme irréductible de degré n.
- ◇ $\mathsf{KeyGen}(\mathsf{param})$: Générer $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathfrak{S}_d^2(\mathcal{V})$. Calculer $\mathbf{h} = \mathbf{x}^{-1}\mathbf{y} \mod P$ et renvoyer $\mathsf{pk} = \mathbf{h}$ et $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- ◇ $\mathsf{Encaps}(\mathsf{pk})$: Générer $(\mathbf{e}_1, \mathbf{e}_2) \xleftarrow{\$} \mathfrak{S}_r^2(\mathcal{V})$, $E = \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$, calculer $\mathbf{c} = \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{h} \mod P$. Calculer $K = \mathsf{Hash}(E)$ et renvoyer $\mathbf{c}$.
- ◇ $\mathsf{Decaps}(\mathsf{sk}, \mathbf{c})$: Calculer $\mathbf{s} = \mathbf{x} \cdot \mathbf{c} \mod P$, $F = \mathsf{Supp}(\mathbf{x}, \mathbf{y})$ et $E = RSR(F, \mathbf{s}, r)$.
  Retrouver $K = \mathsf{Hash}(E)$.

FIGURE 2.3 – Description du protocole LAKE (ROLLO I) [MAB+20a]

Au niveau du Decaps, connaissant la clé secrète sk, on peut calculer $\mathbf{x} \cdot \mathbf{c}$, on a alors :

$$\begin{aligned} \mathbf{x} \cdot \mathbf{c} &= \mathbf{x}(\mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{h}) \\ &= \mathbf{x}\mathbf{e}_1 + \mathbf{x}\mathbf{e}_2\mathbf{x}^{-1}\mathbf{y} \\ &= \mathbf{x}\mathbf{e}_1 + \mathbf{y}\mathbf{e}_2 \end{aligned}$$

On obtient alors un vecteur de l'espace produit $\langle E, F \rangle$. Connaissant le support F, algorithme RSR permet de récupérer le support de l'erreur E. On peut alors récupérer la clé partagée en calculant $K = \mathsf{Hash}(E)$

Le schéma LOCKER

LOCKER, correspondant à ROLLO-II, est presque identique au schéma LAKE et peut être vu comme une version chiffrement à clé publique de celui-ci. Il s'agit donc d'un schéma de chiffrement à clé publique sécurisé contre les attaques à chiffrés choisis (IND-CC2) soumis durant le processus de standardisation du NIST. A la différence de LAKE, un message μ est XOR au hash du support E, qui correspondait sur le schéma précédent à la clé commune partagée. La description du schéma est donnée sur la figure 2.4.

Au niveau du déchiffrement, on récupère la valeur de $\mathsf{Hash}(E)$ comme dans le schéma précédent que l'on XOR avec l'expression *cipher* pour retrouver le message initial $\mathbf{m}$.

- ◇ $\mathsf{Setup}(1^{\mathfrak{K}})$: Etant donné $\mathfrak{K}$, générer et renvoyer les paramètres globaux $\mathsf{param} = (n, m, d, r, P)$ où $P \in \mathbb{F}_q[X]$ désigne un polynôme irréductible de degré n.
- ◇ $\mathsf{KeyGen}(\mathsf{param})$: Générer $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathfrak{S}_d^2(\mathcal{V})$. Calculer $\mathbf{h} = \mathbf{x}^{-1}\mathbf{y} \mod P$ et renvoyer $\mathsf{pk} = \mathbf{h}$ et $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- ◇ $\mathsf{Encrypt}(\mathbf{m}, \mathsf{pk})$: Générer $(\mathbf{e}_1, \mathbf{e}_2) \xleftarrow{\$} \mathfrak{S}_r^2(\mathcal{V})$, $E = \mathsf{Supp}(\mathbf{e}_1, \mathbf{e}_2)$, calculer $\mathbf{c} = \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{h} \mod P$. Calculer $cipher = \mathbf{m} \oplus \mathsf{Hash}(E)$ et renvoyer le chiffré $C = (\mathbf{c}, cipher)$.
- ◇ $\mathsf{Decrypt}(C, \mathsf{sk})$: Calculer $\mathbf{s} = \mathbf{x} \cdot \mathbf{c} \mod P$, $F = \mathsf{Supp}(\mathbf{x}, \mathbf{y})$ et $E = RSR(F, \mathbf{s}, r)$. Renvoyer $\mathbf{m} = cipher \oplus \mathsf{Hash}(E)$.

FIGURE 2.4 – Description du schéma LOCKER [MAB+20a]

Le schéma Ouroboros-R

Le schéma Ouroboros-R, correspondant à ROLLO-III (cette version à été officiellement supprimée de ROLLO lors de la mise à jour d'avril 2020), peut-être vu comme une version d'échange de clés de RQC à laquelle on aurait oté les codes de Gabidulin 2.5.

- ◇ $\mathsf{Setup}(1^{\mathfrak{K}})$: Etant donné $\mathfrak{K}$, générer et renvoyer les paramètres globaux $\mathsf{param} = (n, m, d, r, P)$ où $P \in \mathbb{F}_q[X]$ désigne un polynôme irréductible de degré n.
- ◇ $\mathsf{KeyGen}(\mathsf{param})$: Générer $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathfrak{S}_d^2(\mathcal{V})$, $\mathbf{h} \xleftarrow{\$} \mathcal{V}$. Calculer $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$. Renvoyer $\mathsf{pk} = (\mathbf{h}, \mathbf{s})$ et $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$.
- ◇ $\mathsf{Encaps}(\mathsf{pk})$: Générer $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \xleftarrow{\$} \mathfrak{S}_r^3(\mathcal{V})$, $E = \mathsf{Supp}(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$. Calculer $\mathbf{c}_1 = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2 \mod P$ et $\mathbf{c}_2 = \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{r}_3$. Calculer $K = \mathsf{Hash}(E)$ et renvoyer $\mathbf{c}_1$ et $\mathbf{c}_2$.
- ◇ $\mathsf{Decaps}(\mathsf{sk}, \mathbf{c}_1, \mathbf{c}_2)$: Calculer $\mathbf{e} = \mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1 \mod P$, $F = \mathsf{Supp}(\mathbf{x}, \mathbf{y})$ et $E = RSR(F, \mathbf{e}, r)$. Retrouver $K = \mathsf{Hash}(E)$.

FIGURE 2.5 – Description du schéma Ouroboros-R [MAB+17]

Au niveau du Decaps, la clé secrète permet le calcul de $\mathbf{e}$, qui comme dans RQC est un vecteur de l'espace produit $\langle E, F \rangle$. Connaissant F grâce à la clé secrète, on peut récupérer le support E et donc la valeur de $K = \mathsf{Hash}(E)$.

Chapitre 3

Un HPS reposant sur le chiffrement RQC

La notion de Hash Proof System (HPS), également désignée sous le nom de Smooth Projective Hash Functions, a été introduite pour la première fois à Eurocrypt'02 par Cramer et Shoup. Cette primitive cryptographique peut être vue comme un système de preuve à divulgation nulle de connaissance non interactif sur un langage $\mathcal{NP}$. Elle a donné lieu à l'élaboration des premiers chiffrements à clé publique résistants contre les attaques à chiffrés choisis [CS02b], et a été exploitée par la suite pour d'autres applications comme par exemple l'élaboration de protocoles d'échange de clés authentifiées (AKE), notamment basés sur l'utilisation de mots de passe (PAKE) [GL03], des protocoles de type Oblivious Transfert (OB) [Kal05] ou des chiffrements à témoin.

En section 3.1, nous décrirons la notion de Hash Proof System ainsi que les deux propriétés généralement attendues d'un HPS, à savoir la *correctness* et la *smoothness*. La section 3.2 présentera un état de l'art sur les HPS post-quantiques existants. La section 3.3 décrira un problème récurrent concernant les HPS post-quantiques. En effet, ceux-ci sont généralement construits à partir du langage des chiffrés d'un message μ. Tous les éléments correspondants à des chiffrés valides du message μ doivent vérifier la propriété de *correctness* tandis que tous les éléments qui ne sont pas des chiffrés valides du message μ doivent eux vérifier la propriété de *smoothness*. Or, il existe généralement des éléments indistinguables des chiffrés valides de μ qui ne sont pas des chiffrés valides de μ. Le traitement à apporter à des derniers posent alors problème dans les preuves

de sécurité. La section 3.4 décrira le HPS basé sur le schéma de chiffrement RQC que nous avons élaboré ainsi que les preuves liées à sa sécurité. La section 3.5 décrira les preuves de validité à la Stern nécéssaire pour gérer la zone problématique évoquée en section 3.3. La section 3.6 décrira deux applications de HPS, à savoir un schéma de chiffrement à témoin et un schéma d'échange de clés authentifié par mot de passe. Enfin, la section 3.7 explicitera des jeux de paramètres viables pour les protocoles en question.

3.1 Définition d'un HPS

Un HPS est défini à partir d'un langage $\mathcal{NP}$, généralement noté $\mathcal{L}$, qui est un sous-espace d'un espace ambiant que l'on notera $\mathcal{X}$. Il utilise deux clés, hk et hp, qui représentent respectivement la clé de hachage et la clé de projection. La clé de hachage hk est nécessaire pour associer à tout élément $x \in \mathcal{X}$ un haché noté $\mathcal{H}_{\mathsf{hk}}$. La clé de projection hp quant à elle associera à tout mot $W \in \mathcal{L}$ un haché projeté $\mathcal{H}_{\mathsf{hp}}$, un témoin w de l'appartenance de W au langage $\mathcal{L}$ est nécessaire dans ce cas de figure. Cette clé de projection est calculée à partir de la clé hk et peut dépendre ou non d'un mot W choisi dans le langage $\mathcal{L}$. Dans le cas où cette clé est générée de manière indépendante des mots du langage, le HPS est dit KV, des initiales des auteurs Katz et Vaikuntanathan qui ont été les premiers à introduire cette notion [KV11], permettant ainsi de se protéger des attaques où l'adversaire choisirait le mot W après avoir eu connaissance de la valeur de hp.

Pour tout mot $W \in \mathcal{L}$, les valeurs des hachés $\mathcal{H}_{\mathsf{hk}}$ et $\mathcal{H}_{\mathsf{hp}}$ doivent être égales. Cette propriété est appelée la *correctness* dudit HPS. Si $x \notin \mathcal{L}$, la valeur du haché $\mathcal{H}_{\mathsf{hk}}$ doit être indistinguable d'une valeur générée aléatoirement. Cette deuxième propriété est appelée la *smoothness*.

Ainsi, le fait d'être capable de calculer le haché projeté $\mathcal{H}_{\mathsf{hp}}$ correspondant à un mot du langage, étant seulement donné la clé de projection hp, peut être interprété comme une preuve de connaissance implicite de l'appartenance du mot au langage en question. La figure 3.1 ci-dessous donne une vue globale du fonctionnement d'un Hash Proof System.

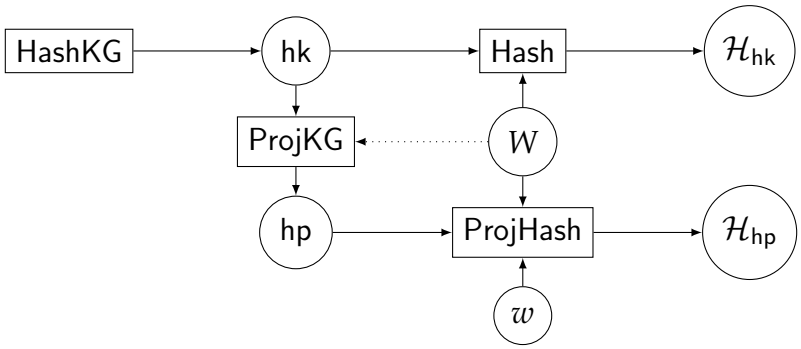

FIGURE 3.1 – Schématisation d'un Hash Proof System [BHG+16]

Lorsqu'il est difficile de distinguer entre les mots du langage et ceux de l'espace ambiant, difficile au sens calculatoire du terme, on parle alors de langage difficilement décelable (*hard-subset-membership* language). Formellement :

Définition 43 (Langage difficilement décelable)**.** *Un langage est dit difficilement décelable s'il possède les trois propriétés suivantes :*

- ◇ *$\mathcal{L}$-samplability : Il existe un algorithme probabiliste en temps polynomial prenant en entrée un paramètre de sécurité $\mathfrak{K}$ et permettant d'obtenir un mot W choisi aléatoirement dans le langage $\mathcal{L}$ ainsi qu'un témoin valide w de l'appartenance de ce mot au langage selon une distribution donnée (pas nécessairement la distribution uniforme) ;*
- ◇ *$\mathcal{X}$-samplability : Il existe un algorithme probabiliste en temps polynomial prenant en entrée un paramètre de sécurité $\mathfrak{K}$ et permettant d'obtenir un élément aléatoire x de $\mathcal{X}$ selon une distribution donnée (pas nécessairement la distribution uniforme) ;*
- ◇ *Hard-subset-membership : Soit $\mathcal{U}_{\mathcal{X}}$ la distribution aléatoire uniforme sur $\mathcal{X}$ et $\mathcal{U}_{\mathcal{L}}$ la distribution aléatoire uniforme sur $\mathcal{L}$. Le langage à la propriété d'être hard-subset-membership si les deux distributions $\mathcal{U}_{\mathcal{X}}$ et $\mathcal{U}_{\mathcal{L}}$ sont calculatoirement indistinguables ;*

Décrivons maintenant formellement la notion d'Hash Proof System ainsi que ses deux propriétés principales, à savoir la *correctness* et *smoothness*.

Définition 44 (Hash Proof System)**.** *Un hash proof system défini sur $\mathcal{L} \subset \mathcal{X}$ à valeur dans $\mathcal{V}$ est défini par la donnée de cinq algorithmes* (Setup, HashKG, ProjKG, Hash, ProjHash) *:*

- ◇ $\mathsf{Setup}(1^{\mathfrak{K}})$ *génère à partir de $\mathfrak{K}$ les paramètres globaux* param *du schéma ainsi*

que la description du langage $\mathcal{NP}$ utilisé noté $\mathcal{L}$;

- ◇ $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$ *renvoie la clé de hachage* hk *associée au langage* $\mathcal{L}$ *;*
- ◇ $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \mathsf{param}, W)$ *dérive la clé de projection* hp*, possiblement dépendante du mot* W *utilisé pour générer la clé* hk *;*
- ◇ $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W)$ *renvoie le haché* $\mathcal{H}_{\mathsf{hk}} \in \mathcal{V}$ *en utilisant la clé de hachage* hk *et le mot* W *;*
- ◇ $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, W, w)$ *renvoie le haché* $\mathcal{H}_{\mathsf{hp}} \in \mathcal{V}$ *en utilisant la clé de projection* hp *et le témoin* w *d'appartenance du mot* W *au langage* $\mathcal{L}$ *;*

Définition 45 (Correctness)**.** *Soit $W \in \mathcal{L}$ et w un témoin de son appartenance au langage. Un Hash Proof System satisfait la propriété de correctness si, pour toute clé de hashage* hk *et toute clé de projection associée* hp*, on a $\mathcal{H}_{\mathsf{hk}} = \mathcal{H}_{\mathsf{hp}}$.*

Pour parvenir à construire des HPS post-quantiques, la notion d'*approximate correctness* fut introduite dans [KV09]. Dans un tel HPS, la propriété de correctness est assouplie de façon à ce que les deux valeurs des hachés n'aient plus à être égales mais seulement suffisamment proches l'une de l'autre selon une certaine distance.

Définition 46 (ϵ-correctness)**.** *Soit $W \in \mathcal{L}$ et w un témoin de son appartenance au langage. Soit d une distance sur $\mathcal{V}$. Un Hash Proof System satisfait la propriété d'ϵ-correctness si, pour toute clé de hachage* hk *et toute clé de projection associée* hp*, on a $d(\mathcal{H}_{\mathsf{hk}}, \mathcal{H}_{\mathsf{hp}}) \leqslant \epsilon$.*

En ce qui concerne la propriété de *smoothness*, plusieurs définitions différentes ont été proposées menant à différentes familles de HPS dénommées CS-HPS, GL-HPS et KV-HPS des initiales de leurs auteurs respectifs Cramer et Shoup, Gennaro et Lindell, Katz et Vaikuntanathan [CS02b, GL03, KV09]. La propriété de *smoothness* s'assure qu'étant donné une clé projeté hp, le hashé $\mathcal{H}_{\mathsf{hk}}$ d'un élément $x \notin \mathcal{L}$ soit indistinguable d'une valeur aléatoire choisie de manière uniforme dans $\mathcal{V}$.

Les premières définitions de la smoothness furent celles de Cramer et Shoup, puis de Gennaro et Lindell. La seule différence entre ces deux définitions est que la clé projetée hp peut dépendre du choix du mot du langage W dans la deuxième, ce qui n'était pas envisagé dans la première définition. La troisième définition, celle de KV-smoothness, fut introduite pour prendre en compte le fait qu'un attaquant puisse malicieusement générer le mot W après avoir pris

connaissance de la valeur de hp. Pour pallier à ce problème, la notion de KV-smoothness calculatoire fut introduite [BBC+13], c'est cette définition que nous adopterons (voir figure 3.2).

$\mathsf{Exp}^{\mathsf{smooth-b}}_{\mathcal{A}}(\mathfrak{K})$:
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$
3. $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, \mathsf{param})$
4. $\mathbf{x} \notin \mathcal{L} \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}, \mathsf{hp})$
5. If $b = 0$, $\mathcal{H}_{\mathsf{hk}} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, \mathbf{x})$
If $b = 1$, $\mathcal{H}_{\mathsf{hk}} \xleftarrow{\$} \mathcal{V}$
6. Return $b' \leftarrow \mathcal{A}.\mathsf{guess}(\mathcal{L}, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, \mathbf{x})$

FIGURE 3.2 – Jeu $\mathsf{Exp}^{\mathsf{smooth-b}}_{\mathcal{A}}(\mathfrak{K})$ définissant la KV-smoothness calculatoire

3.2 Etat de l'art sur les HPS post-quantiques

Il n'existe que peu de Hash Proof Systems dans le domaine post-quantique à ce jour à ma connaissance.

Le premier fut proposé par Katz et Vaikuntanathan en 2009 [KV09]. Comme mentionné à la section précédente, cette publication a introduit la notion d'approximate correctness et par suite d'approximate HPS. Leur construction à permis l'élaboration du premier schéma de chiffrement à clé publique sécurisé contre les attaques à chiffré choisi et dont la sécurité repose sur la difficulté du problème *learning with errors* (LWE) introduit par Regev [Reg09], qui peut être vu comme une généralisation du problème *learning parity with noise* (LPN) qui sera introduit en détail à la section 4.1. Cependant, le langage utilisé n'est pas exactement défini comme étant le langage standard des chiffrés valides de LWE, ce qui a engendré un déchiffrement couteux, comme explicité dans [BBDQ18]. Ils ont également élaboré une construction générique permettant d'obtenir un PAKE en trois passes à partir d'un chiffrement à clé publique CCA2 associé à un HPS et ont été capables de concevoir le premier PAKE dont la sécurité repose sur l'utilisation des réseaux euclidiens. Celui-ci est prouvé dans le modèle standard, par conséquent ne dépend pas de l'existence théorique d'un oracle aléatoire mais requiert l'utilisation d'une chaîne de caractères commune (CRS) partagée par le client et le serveur au début du protocole.

Un second fut proposé par Persichetti en 2013 [Per13]. Son but était de concevoir un Hash Proof System reposant sur l'utilisation des codes correcteurs d'erreurs et de construire un schéma de chiffrement à clé publique résistant à un certain type d'attaque par canaux cachés exploitant la mémoire de l'appareil utilisé, appelée « cold boot », à partir dudit HPS. Dans cette publication, une propriété d'universalité est introduite qui se substitue à la propriété de smoothness, et des transformations génériques existent permettant d'aboutir à la construction d'un HPS de type GL/CS (dont la valeur des hachés dépend du mot du langage considéré) à partir d'un HPS ayant cette propriété. Cependant, bien que la propriété d'universalité ait été bien définie, la preuve quant à elle considère des clés choisies aléatoirement au lieu de prouver le résultat pour l'ensemble de toutes les clés possibles, ce qui fait que des clés choisies spécifiquement par un adversaire pourraient, et dans ce cas feraient, échouer la preuve, compromettant ainsi la sécurité de la construction.

En 2017, Zhang and Yu [ZY17] proposèrent une autre construction générique permettant d'obtenir un PAKE en deux passes à partir d'un chiffrement à clé publique CCA2 et d'un HPS associé au lieu des trois requis dans [KV09]. Le chiffrement utilisé devait avoir la propriété d'être formé en deux parties indépendantes l'une de l'autre, la première partie ayant pour but d'accomplir la « fonctionnalité » du chiffrement tandis que la deuxième devait permettre d'atteindre la sécurité CCA. Le Hash Proof System utilisé devait quant à lui être non-adaptatif, autrement dit la fonction de projection ProjKG devait uniquement dépendre de la valeur hk et donc être indépendante du mot du langage choisi (KV-HPS). Ils proposèrent une construction concrète d'un tel chiffrement et du HPS associé à partir du problème LWE. Leur construction faisait appel à une preuve non-interactive à divulgation nulle de connaissance dans le modèle de l'oracle aléatoire en supplément du HPS.

Enfin, en 2018, Ben Hamouda et al. [BBDQ18] proposèrent un Hash Proof System basé sur le langage standard des chiffrés LWE permettant de pallier aux problèmes de décodage rencontrés dans [KV09]. Deux variantes ont été proposées, l'une basée sur un chiffrement IND-CCA2 à la Miccianco Peikert [MP12] et l'autre sur une restriction IND-CPA du même schéma. Cette dernière version leur ont permis d'obtenir le premier KV-HPS (non approximé) basé sur les réseaux euclidiens ainsi que le premier PAKE en deux passes basé sur les réseaux

euclidiens dans le modèle standard. L'ajout d'une preuve de connaissance non-interactive au HPS leur a également permis d'atteindre un PAKE en une seule passe.

3.3 Une zone problématique

Bien que le HPS conçu par Katz et Vaikuntanathan [Kal05] ait montré qu'un langage basé sur des chiffrés s'avère efficace pour concevoir des HPS post-quantiques, l'utilisation d'un tel langage semble introduire de façon inhérente une zone problématique pour la preuve de smoothness du HPS sous-jacent. En effet, si l'on note $\mathcal{L}_\mu$ l'ensemble des chiffrés d'un message μ donné et par $\mathcal{L}_\mu^*$ l'ensemble des éléments qui se déchiffrent en μ, alors on peut remarquer que ces ensembles sont distincts et que $\mathcal{L}_\mu \subseteq \mathcal{L}_\mu^*$ étant donné que tout chiffré valide de μ se déchiffre en μ (on considère ici un chiffrement déterministe).

Cependant, $\mathcal{L}_\mu^* \subseteq \mathcal{L}_\mu$ n'est pas vrai en général pour les langages basés sur des chiffrés, puisqu'il existe des éléments n'ayant pas la forme d'un chiffré mais qui pourtant se déchiffrent également en μ. L'existence de tels éléments jette un doute sur la sécurité générale des schémas construits autour de ces HPS car un adversaire pourrait volontairement générer des éléments se déchiffrant en μ sans appartenir au langage du HPS et éventuellement s'en servir pour concevoir des attaques et obtenir des informations auxquelles il n'est pas censé accéder.

L'existence d'une telle zone grise s'avère particulièrement problématique si l'on souhaite obtenir un HPS sécurisé dans le modèle UC (Universaly Composable). Imaginons un PAKE (Password Authenticated Key Exchange protocol) où chaque utilisateur dérive une clé partagée d'un HPS à partir d'un commitment issu de leur mot de passe commun et où le HPS en question ne gèrerait pas ladite zone problématique. Dans la preuve de composition universelle (UC proof), il est nécessaire de construire un simulateur capable de reproduire de façon abstraite la fonctionnalité idéale du protocole qu'il simule.

Jusqu'à présent, les HPS post-quantiques ont toujours eu besoin d'affaiblir cette fonctionnalité, le simulateur étant incapable de détecter si l'adversaire lui a envoyé un chiffré valide du mot de passe ou un élément qui, pour une clé secrète donnée, se déchiffre en le mot de passe.

Dans le premier cas de figure, le protocole doit toujours aboutir à un succès, assurant ainsi une correctness parfaite, mais dans le second cas le comportement que doit adopter le protocole n'est pas clair. Soit il considère ce comportement comme admissible et le protocole aboutira toujours à un succès, ce qui n'est pas le comportement considéré dans les HPS post-quantiques existants, soit ce comportement est prohibé et le protocole doit alors aboutir à un échec, mais le protocole étant incapable de discerner entre les chiffrés valides et les éléments se déchiffrant en $\boldsymbol{\mu}$ ne pourra pas assurer correctement sa fonctionnalité.

Ainsi, supprimer cette zone grise permettrait de concevoir des HPS post-quantiques dans le modèle UC, ce qui n'a jamais été réalisé jusqu'à présent.

3.4 Elaboration d'un HPS basé sur RQC

3.4.1 Langage

Comme les autres HPS post-quantiques cités précédemment, le langage que nous avons utilisé est également basé sur des chiffrés, et plus précisement sur les chiffrés du schéma de chiffrement à clé publique RQC, candidat à l'appel à standardisation du NIST.

L'espace ambiant sera constitué de l'ensemble des chiffrés possibles de RQC et le langage $\mathcal{L}_\mu$ sera défini comme étant l'ensemble des chiffrés d'un message donné $\boldsymbol{\mu}$.

Soit n, m, k, q, w_r des entiers naturels dépendant d'un paramètre de sécurité λ. Le langage $\mathcal{L}_\mu$ est alors constitué de l'ensemble des chiffrés $\mathbf{c} \in \mathcal{CT}$ d'un message $\boldsymbol{\mu} \in \mathcal{PT}$ tel que défini sur la figure 2.2. Ainsi, étant donné une clé publique de RQC $\mathsf{pk} = (\mathbf{g}, \mathbf{H}, \mathbf{s})$, nous avons :

$$
\begin{aligned}
\mathcal{X} &= \left\{ \mathbf{c} \in \mathcal{CT} \;\middle|\; \exists \boldsymbol{\mu} \in \mathcal{PT}, \exists \theta,\ \mathbf{c} = \mathsf{RQC.Encrypt}(\mathsf{pk}, \boldsymbol{\mu}, \theta) \right\}. \\
\mathcal{L}_\mu &= \left\{ \mathbf{c} \in \mathcal{CT} \;\middle|\; \exists \theta,\ \mathbf{c} = \mathsf{RQC.Encrypt}(\mathsf{pk}, \boldsymbol{\mu}, \theta) \right\} \\
&= \left\{ \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 + \mathbf{H} \cdot \mathbf{r}_2 \\ \boldsymbol{\mu} \cdot \mathrm{Gab}_{\mathbf{g}} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{r}_3 \end{pmatrix} \;\middle|\; (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \xleftarrow{\$} \mathfrak{S}^3_{w_r}(\mathcal{V}) \right\}.
\end{aligned}
$$

Le fait de restreindre l'espace ambiant aux seuls chiffrés de RQC permet de

totalement supprimer la zone problématique évoquée précédemment. Dans le cas contraire, si l'espace ambiant eut été $\mathcal{V}$ tout entier, nous aurions alors eu un ensemble important d'éléments se déchiffrant en $\boldsymbol{\mu}$ sans être pour autant des chiffrés de $\boldsymbol{\mu}$. Prenons par exemple un élément de la forme $e = (\mathbf{c}_1, \mathbf{c}_2)$ avec $\mathbf{c}_1 = \mathbf{H} \cdot \mathbf{r}_2$, $\mathbf{c}_2 = \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}} + \mathbf{s} \cdot \mathbf{r}_2$ et $\mathbf{r}_2$ de rang w_r (un chiffré tel que dans RQC mais avec $\mathbf{r}_1$ et $\mathbf{r}_3$ égaux à zero).

Dans ce cas de figure, on aurait $\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1 = \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}} + \mathbf{x} \cdot \mathbf{r}_2$. Cette expression se déchiffre bien en $\boldsymbol{\mu}$ car le rang de $\mathbf{x} \cdot \mathbf{r}_2$ sera bel et bien en-dessous de la capacité de correction du code de Gabidulin utilisé, pour autant cet élément n'est clairement pas un chiffré valide de RQC car $\mathbf{r}_1$ et $\mathbf{r}_3$ devraient être deux vecteurs de même support et de rang w_r.

En restreignant l'espace ambiant aux seuls chiffrés de RQC, notre construction ne souffre plus du problème récurrent qui semble inhérent aux langages basés sur des chiffrés pour des chiffrements post-quantiques, et permet donc d'ouvrir la voie à la conception d'HPS post-quantiques sécurisés dans le modèle UC.

Cette restriction a cependant une contrepartie car elle implique de devoir vérifier que les éléments que nous manipulons sont bien des chiffrés de RQC, ce qui peut-être fait soit *à fortiori* si le mot W est généré par un utilisateur honnête, soit *à priori* à l'aide d'une preuve de validité du chiffré. Ces stratégies ont respectivement été utilisées pour construire les applications proposées dans la section 3.6.1.

3.4.2 Construction

Notre construction, décrite sur la figure 3.3, n'étant pas un HPS exact mais approximatif, calcule deux valeurs $\mathcal{H}_{\mathsf{hk}}$ et $\mathcal{H}_{\mathsf{hp}}$ telles que $\|\mathcal{H}_{\mathsf{hk}} - \mathcal{H}_{\mathsf{hp}}\|$ est de rang faible. Pour atteindre cette condition, les hachés $\mathcal{H}_{\mathsf{hk}}$ et $\mathcal{H}_{\mathsf{hp}}$ ont été définis de façon à ce que chacun d'entre-eux contienne l'expression $\mathbf{s}\boldsymbol{\alpha}_1\mathbf{r}_2$ mais diffèrent de part l'ajout d'expressions appartenant à l'espace produit $\langle E_\alpha, E_r \rangle$ où E_r et E_α sont deux sous-espaces de faible dimension, représentant respectivement les supports partagés des valeurs $\boldsymbol{\alpha}_i$ et des valeurs $\mathbf{r}_i$.

De plus, il est possible de vérifier, à l'aide du code de Gabidulin utilisé, si le mot W utilisé est dans le langage $\mathcal{L}_\mu$ ou non en exploitant la clé secrète $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$ comme une trappe.

- ◇ $\mathsf{Setup}(1^{\mathfrak{K}})$: Etant donné le paramètre de sécurité $\mathfrak{K}$, générer les paramètres param du schéma, à savoir $(n, m, k, q, w_r, w_\alpha, w, \theta)$. Générer $\mathbf{h} \xleftarrow{\$} \mathcal{V}$, $\mathbf{g} \xleftarrow{\$} \mathcal{S}_n(\mathcal{V})$, $(\mathbf{x}, \mathbf{y}) \xleftarrow{\$} \mathfrak{S}^2_w(\mathcal{V})$, utiliser la graine θ pour générer $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \xleftarrow{\$} \mathfrak{S}^3_{w_r}(\mathcal{V})$. Calculer $(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{RQC.Encrypt}(\mathsf{pk}, \boldsymbol{\mu}, \theta)$ et $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$.
- ◇ $\mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$: Générer $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3) \xleftarrow{\$} \mathfrak{S}^3_{w_\alpha}(\mathcal{V})$.
  Retourner $\mathsf{hk} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3)$.
- ◇ $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_\mu, \mathsf{param})$: Retourner $\mathsf{hp} = \mathbf{s} \cdot \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2$.
- ◇ $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}_\mu, W)$: Retourner $\mathcal{H}_{\mathsf{hk}} = \boldsymbol{\alpha}_1 \cdot (\mathbf{c}_2 - \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}}) + \boldsymbol{\alpha}_3$.
- ◇ $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}_\mu, W, w)$: Retourner $\mathcal{H}_{\mathsf{hp}} = \mathsf{hp} \cdot \mathbf{r}_2$

FIGURE 3.3 – Un HPS basé sur les code-correcteurs d'erreurs et le schéma RQC

3.4.3 Correctness

Théorème 1. *Le* HPS *décrit dans la figure 3.3 satisfait la propriété d'ϵ-correctness pour $\epsilon = w_\alpha(w_r + 1)$.*

Démonstration. Soit $W \in \mathcal{L}_\mu$. Nous avons $\mathcal{H}_{\mathsf{hk}} - \mathcal{H}_{\mathsf{hp}} = \boldsymbol{\alpha}_1 \cdot \mathbf{r}_3 - \boldsymbol{\alpha}_2 \cdot \mathbf{r}_2 + \boldsymbol{\alpha}_3$. Comme $\mathsf{Supp}(\boldsymbol{\alpha}_1) = \mathsf{Supp}(\boldsymbol{\alpha}_2)$ et $\mathsf{Supp}(\mathbf{r}_3) = \mathsf{Supp}(\mathbf{r}_2)$, les espaces produits $\langle \boldsymbol{\alpha}_1 \mathbf{r}_3 \rangle_{\mathbb{F}_q}$ et $\langle \boldsymbol{\alpha}_2 \mathbf{r}_2 \rangle_{\mathbb{F}_q}$ sont égaux et ont pour dimension $w_\alpha w_r$. L'espace $\langle \boldsymbol{\alpha}_3 \rangle_{\mathbb{F}_q}$ quant à lui n'est pas inclu dans l'espace produit précédent et a dimension w_α. Au final, l'expression a donc un rang maximal de $w_\alpha \cdot w_r + w_\alpha = w_\alpha\ (w_r + 1)$.

□

3.4.4 Smoothness

La preuve de smoothness quant à elle est bien plus délicate et nous a amené à introduire une variante du problème décisionnel IRSD où l'adversaire peut partiellement manipuler la matrice de parité utilisée.

Etant donné $\mathbf{s}, \mathbf{t} \xleftarrow{\$} \mathcal{V}$ et $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{V}$, considérons le syndrome :

$$\begin{pmatrix} \mathcal{I}(\mathbf{s}) & \mathbf{I}_n & \mathcal{I}(0) \\ \mathcal{I}(\mathbf{t}) & \mathcal{I}(0) & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix}.$$

Trouver $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3) \in \mathfrak{S}^3_{w_\alpha}(\mathcal{V})$ satisfaisant cette équation correspond à résoudre

une instance d'un problème 3-IRSD. Supposons maintenant qu'à la place d'une valeur $\mathbf{t}$ choisie de façon aléatoire l'adversaire puisse choisir $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$ et $(\mathbf{r}_1, \mathbf{r}_2) \in \mathfrak{S}_{w_r}^2(\mathcal{V})$ tels que $\mathbf{t} = \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}} + \mathbf{s}\mathbf{r}_1 + \mathbf{r}_2$. La question à laquelle il nous faut répondre est de déterminer si, sous cette forme particulière de la matrice de parité, le problème reste difficile et plus particulièrement sa version décisionnelle.

Définition 47 (FIRSD distribution)**.** *Etant donné* $\mathbf{g} \in \mathcal{S}_n(\mathcal{V})$, $\mathbf{s} \xleftarrow{\$} \mathcal{V}$ *et* $w_\alpha, w_r \in \mathbb{N}$, *considérons un oracle noté* $\mathcal{O}_{\mathbf{s},\mathbf{g}}(\mathbf{t})$ *qui génère* $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3) \xleftarrow{\$} \mathfrak{S}_{w_\alpha}^3(\mathcal{V})$, *calcule et retourne* $\mathbf{y}_1 = \mathbf{s}\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2$ *en tant que valeur publique, puis vérifie à l'aide d'une preuve de validité* Π *que son entrée* $\mathbf{t}$ *est de la forme* $\mathbf{t} = \boldsymbol{\mu}Gab_{\mathbf{g}} + \mathbf{s}\mathbf{r}_1 + \mathbf{r}_2$ *avec* $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$ *et* $(\mathbf{r}_1, \mathbf{r}_2) \in \mathfrak{S}_{w_r}^2(\mathcal{V})$, *calcule* $\mathbf{y}_2 = \mathbf{t}\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_3$ *et retourne* $(\mathbf{y}_1, \mathbf{y}_2)$ *si* $\mathbf{t}$ *est valide et* $\perp$ *dans le cas contraire.*

Définition 48 (Version décisionnelle du problème FIRSD)**.** *Etant donné* $\mathbf{g} \in \mathcal{S}_n(\mathcal{V})$, $\mathbf{s} \xleftarrow{\$} \mathcal{V}$ *et* $w_\alpha, w_r \in \mathbb{N}$, *considérons un oracle noté* $\tilde{\mathcal{O}}_{\mathbf{s},\mathbf{g}}(\mathbf{t})$ *qui génère* $(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3) \xleftarrow{\$} \mathfrak{S}_{w_\alpha}^3(\mathcal{V})$ *et pièce* $\xleftarrow{\$} \{0,1\}$, *calcule et revoie* $\mathbf{y}_1 = \mathbf{s}\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2$ *en tant que valeur publique, puis vérifie à l'aide d'une preuve de validité* Π *que son entrée* $\mathbf{t}$ *est de la forme* $\mathbf{t} = \boldsymbol{\mu}Gab_{\mathbf{g}} + \mathbf{s}\mathbf{r}_1 + \mathbf{r}_2$ *avec* $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$ *et* $(\mathbf{r}_1, \mathbf{r}_2) \in \mathfrak{S}_{w_r}^2(\mathcal{V})$, *calcule* $\mathbf{y}_2 = \mathbf{t}\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_3$, *et renvoie, si* $\mathbf{t}$ *est valide, le couple* $(\mathbf{y}_1, \mathbf{y}_2)$ *si pièce = 0, le couple* $(\mathbf{y}_1, \mathbf{y}_2)$ *avec* $\mathbf{y}_2$ *une valeur générée aléatoirement dans* $\mathcal{V}$ *si pièce = 1, et* $\perp$ *si* $\mathbf{t}$ *n'est pas valide. La version décisionnelle du problème* FIRSD *consiste à déterminer, avec un avantage non négligeable, si* $\mathbf{y}_2$ *vient de la distribution* FIRSD *ou de la distribution uniforme dans* $\mathcal{V}$.

Conjecture. *La version décisionnelle du problème* FIRSD *est un problème difficile.*

Comme évoqué précédemment, ce problème est proche de la version décisionnelle d'un problème 3-IRSD, la différence étant que l'adversaire peut partiellement contrôler la matrice de parité $\mathbf{H}$ utilisée, plus spécifiquement la matrice idéale générée à partir de $\mathbf{t}$. La question est alors de déterminer l'impact que cette différence a sur la difficulté du problème initial.

Point de vue global. Soit $\mathbf{g} \in \mathcal{S}_n(\mathcal{V})$, $\mathbf{s} \xleftarrow{\$} \mathcal{V}$ et $w_\alpha, w_r \in \mathbb{N}$. Considérons un adversaire choisissant $(\mathbf{r}_1, \mathbf{r}_2) \in \mathfrak{S}_{w_r}^2(\mathcal{V})$ et $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$, et générant $\mathbf{t} = \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}} + \mathbf{s}\mathbf{r}_1 + \mathbf{r}_2$. La version décisionnelle du problème FIRSD consiste à distinguer une valeur $\mathbf{y}_2$ ayant la forme d'un syndrome comme décrit ci-dessous d'une valeur

générée de manière aléatoire.

$$\begin{pmatrix} \mathcal{I}(\mathbf{s}) & \mathbf{I}_n & 0 \\ \mathcal{I}(\mathbf{t}) & 0 & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix}.$$

Considérons deux cas extrêmes pour illustrer ce nouveau problème. Un premier cas où il n'y aurait aucune entropie et où l'adversaire n'aurait donc aucun impact sur la valeur finale du vecteur $\mathbf{t}$. Dans cette situation, $\mathbf{t}$ serait un vecteur généré aléatoirement dans $\mathcal{V}$ et le problème serait alors ramené à une instance de la version décisionnelle du problème 3-IRSD sous forme systématique. A l'opposé, considérons l'extrême inverse où l'adversaire aurait un contrôle total sur la valeur de $\mathbf{t}$. Dans ce cas, le problème ne serait plus difficile car l'adversaire pourrait alors choisir $\mathbf{t} = (1, 0, \ldots, 0)$ de façon à obtenir la matrice identité, lui permettant ainsi de retrouver facilement le support commun des $\boldsymbol{\alpha}_i$.

On peut ainsi voir que la difficulté du problème repose essentiellement sur la capacité que détient l'adversaire à manipuler la valeur de $\mathbf{t}$. Voyons maintenant plus en détails les possibilités que possède l'attaquant. Dans un premier temps, nous étudierons le problème par une approche entropique et nous montrerons qu'avec un choix judicieux de paramètres, l'adversaire est très restreint dans la génération de la valeur de $\mathbf{t}$. Dans un second temps, nous décrirons des cas où l'adversaire essaie de forger des valeurs de $\mathbf{t}$ ayant une forme très spécifique et nous montrerons qu'il est peu probable que de telles stratégies puissent révéler une quelconque information sur les $\boldsymbol{\alpha}_i$ ou leur support commun. Nous considérerons deux cas de figures, le premier consistant à générer un vecteur $\mathbf{t}$ ayant autant de zéros que possible et le second consistant à réduire le plus possible le rang de $\mathbf{t}$.

Point de vue entropique. Pour générer la valeur du vecteur $\mathbf{t}$, l'adversaire peut d'abord choisir le message $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$ de son choix, ce qui représente q^{km} possibilités. Ensuite, il lui faut choisir un sous-espace E de $\mathbb{F}_{q^m}$ de dimension w_r. Il y a jusqu'à $q^{w_r n}$ possibilités dans le choix de $\mathbf{r}_1$ et jusqu'à $q^{w_r n}$ possibilités pour $\mathbf{r}_2$ également. Au total, le nombre de bits que l'adversaire peut manipuler est majoré par $km + 2nw_r$. Choisir un vecteur dans l'espace ambiant tout entier $\mathcal{V}$ par contre représente q^{mn} possibilités, ce qui peut être vu comme une surface

rectangulaire de m bits par n bits. Au final, la proportion P de bits qu'un adversaire peut manipuler est donc égale à $\frac{km+2nw_r}{mn}$. Avec nos jeux de paramètres, comme on peut le voir dans le tableau qui suit, cette proportion n'excède jamais 13%.

Instance	q	n	m	k	w_r	P	Security
I	2	137	139	4	7	0.123	128
II	2	211	223	4	8	0.104	192
III	2	283	293	3	13	0.099	256

TABLE 3.1 – Proportion de bits manipulables.

La contrainte de la structure idéale.

Comme nous allons le voir, la structure cyclique inhérente aux matrices sous forme idéale est aussi un obstacle important pour l'élaboration d'une attaque à partir du vecteur $\mathbf{t}$. Considèrons le problème 2-IRSD suivant extrait du problème 3-IRSD évoqué précédemment :

$$\begin{pmatrix} \mathcal{I}(\mathbf{t}) & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_3 \end{pmatrix} = \mathbf{y}_2.$$

Première approche. Prenons le cas d'un adversaire tentant d'annuler un maximum de coordonnées du vecteur $\mathbf{t}$ dans le but de récupérer des informations sur les valeurs $\boldsymbol{\alpha}_1$ et $\boldsymbol{\alpha}_3$. Par la résolution linéaire d'un système d'équation, celui-ci pourra annuler approximativement w_r coordonnées du vecteur $\mathbf{t}$. La matrice idéale générée à partir de $\mathbf{t}$ viendra ensuite décaler ces zéros dans la matrice, et certains d'entre-eux seront même absorbés du fait de la réduction modulaire exercée par le polynôme P utilisé par la structure idéale.

Comme le nombre de zéros est limité à environ w_r coordonnées sur un total de n coordonnées, la proportion P_0 de zéros n'excédera pas 6% du nombre total de coordonnées de $\mathbf{t}$ avec nos jeux de paramètres. Par conséquent, il est peu probable qu'une telle stratégie puisse faire fuiter de l'information sur le syndrome, pas plus qu'elle ne modifiera la distribution de façon à pouvoir la distinguer, avec un avantage non négligeable, de la distribution aléatoire uniforme.

Instance	q	n	m	k	w_r	P_0	Security
I	2	137	139	4	7	0.051	128
II	2	211	223	4	10	0.047	192
III	2	283	293	3	13	0.046	256

TABLE 3.2 – Proportion de zéros possibles.

Seconde approche. Un adversaire pourrait également vouloir diminuer le rang de $\mathbf{t}$ au maximum. En procédant de la même manière que précédemment, par la résolution d'un système d'équations, un adversaire pourrait réduire le rang de $\mathbf{t}$ d'approximativement w_r, cependant cette réduction du rang de la matrice de parité ne serait pas suffisante pour permettre l'utilisation d'un algorithme de décodage pour les codes LRPC. De plus, réduire le rang de $\mathbf{t}$ d'approximativement w_r n'affectera pas le rang du syndrome associé $\mathbf{y}_2 = \mathbf{t} \cdot \boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_3$. En effet, nous aurions $\|\mathbf{t}\| = n - w_r$ et $\|\boldsymbol{\alpha}_1\| = \|\boldsymbol{\alpha}_3\| = w_\alpha$, et comme $(n - w_r + 1) \cdot w_\alpha \gg m$, le rang global du syndrome n'en serait pas affecté.

Le problème FIRSD ayant été introduit et sa difficulté mise en lumière, nous pouvons désormais passer à la preuve de smoothness à proprement parler.

Théorème 2. *Le* HPS *décrit en figure 3.3 satisfait la propriété de* KV*-smoothness calculatoire sous l'hypothèse que le problème 2-*IRSD *et la version décisionnelle du problème* FIRSD *soient des problèmes difficiles.*

Démonstration. La preuve est construite comme une succession de jeux, le premier jeu G_1 représentant la situation réelle où l'adversaire reçoit la valeur honnête du haché $\mathcal{H}_{\mathsf{hk}}$ jusqu'au jeu G_4 où l'adversaire reçoit une valeur générée aléatoirement dans $\mathcal{V}$ à la place de $\mathcal{H}_{\mathsf{hk}}$.

Game $\mathbf{G}_{1,\mathcal{A}}(\mathfrak{K})$:

1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$
3. $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_\mu, \mathsf{param})$
4. $W \in \mathcal{X} \setminus \mathcal{L}_\mu \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}_\mu, \mathsf{hp})$
5. $\mathcal{H}_{\mathsf{hk}} \leftarrow \mathsf{Hash}(\mathcal{L}_\mu, \mathsf{hk}, W)$
6. $b' \leftarrow \mathcal{A}.\mathsf{guess}(\mathcal{L}_\mu, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$

Game $\mathbf{G}_{2,\mathcal{A}}(\mathfrak{K})$:

1.a. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$

1.b. $\mathsf{param}.\mathbf{s} \xleftarrow{\$} \mathcal{V}$

2. $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$

3. $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_\mu, \mathsf{param})$

4. $W \in \mathcal{X} \setminus \mathcal{L}_\mu \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}_\mu, \mathsf{hp})$

5. $\mathcal{H}_{\mathsf{hk}} \leftarrow \mathsf{Hash}(\mathcal{L}_\mu, \mathsf{hk}, W)$

6. $b' \leftarrow \mathcal{A}.\mathsf{guess}(\mathcal{L}_\mu, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$

$\mathcal{D}^*_{\mathfrak{K}}((\mathbf{I}_n, \mathcal{I}(\mathbf{h})), \mathbf{s})$:

1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$

2. $\mathsf{param}.\mathbf{s} \leftarrow \mathbf{s}$

3. $\mathsf{param}.\mathbf{h} \leftarrow \mathbf{h}$

4. $b' \leftarrow \mathcal{D}_{\mathfrak{K}}(\mathcal{L}_\mu, \mathsf{param})$

5. If $b' == 1$, output $[2,1]$-IRSD

6. If $b' == 2$, output UNIFORM

Lors du passage au jeu G_2, la valeur secrète associée à $\mathbf{s}$ est oubliée et remplacée par une valeur aléatoire dans $\mathcal{V}$ puis le jeu se poursuit normalement. Supposons par l'absurde qu'il existe un algorithme $\mathcal{D}_{\mathfrak{K}}(\mathcal{L}_\mu, \mathsf{param})$ capable de distinguer entre les jeux G_1 et G_2 avec un advantage ϵ. Alors, il serait possible de construire un algorithme $\mathcal{D}^*_{\mathfrak{K}}$ capable de résoudre le problème 2-IRSD avec le même avantage ϵ ce qui est impossible par hypothèse de départ. Par l'absurde, il n'existe donc aucun algorithme capable de distinguer les jeux G_1 et G_2 sous l'hypothèse que le problème 2-IRSD soit un problème difficile.

Game $\mathbf{G}_{3,\mathcal{A}}(\mathfrak{K})$:

1.a. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$

1.b. $\mathsf{param}.\mathbf{s} \xleftarrow{\$} \mathcal{V}$

2. $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$

3. $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_\mu, \mathsf{param})$

4. $W \in \mathcal{X} \setminus \mathcal{L}_\mu \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}_\mu, \mathsf{hp})$

5.a. $\mathcal{H}_{\mathsf{hk}} \leftarrow \mathsf{Hash}(\mathcal{L}_\mu, \mathsf{hk}, W)$

5.b. $\mathcal{H}_{\mathsf{hk}} \xleftarrow{\$} \mathcal{V}$

6. $b' \leftarrow \mathcal{A}.\mathsf{guess}(\mathcal{L}_\mu, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$

$\mathcal{D}^*_{\mathfrak{K}}(\mathbf{s}, \mathbf{g})$:

1.a $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$

1.b $\mathsf{param}.\mathbf{s} \leftarrow \mathbf{s}$

2. $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$

3 $\mathsf{hp} \leftarrow \tilde{\mathcal{O}}_{\mathbf{s},\mathbf{g}}$ (public value)

4. $W \in \mathcal{X} \setminus \mathcal{L}_\mu \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}_\mu, \mathsf{hp})$

5. $\mathcal{H}_{\mathsf{hk}} \leftarrow \tilde{\mathcal{O}}_{\mathbf{s},\mathbf{g}}(\mathbf{c}_2 - \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}})$

6. $b' \leftarrow \mathcal{D}_{\mathfrak{K}}(\mathcal{L}_\mu, \mathsf{param}, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$

7. If $b' == 3$ output FIRSD

8. If $b' == 4$ output UNIFORM

Lors du passage du jeu G_3, la valeur du haché $\mathcal{H}_{\mathsf{hk}}$ est oubliée et remplacée par une valeur aléatoire dans $\mathcal{V}$, puis le jeu se poursuit normalement. Supposons par l'absurde qu'il existe un algorithme $\mathcal{D}_{\mathfrak{K}}(\mathcal{L}_\mu, \mathsf{param}, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$ capable de distinguer entre les jeux G_2 et G_3 avec un avantage ϵ. Alors, il serait possible de construire un algorithme $\mathcal{D}^*_{\mathfrak{K}}$ capable de résoudre la version décisionnelle du problème FIRSD avec le même avantage ϵ, ce qui est impossible par hypothèse de départ. Par l'absurde, il n'existe donc aucun algorithme capable de distinguer les jeux G_2 et G_3 sous l'hypothèse que la version décisionnelle du problème FIRSD soit un problème difficile.

Game $\mathbf{G}_{4,\mathcal{A}}(\mathfrak{K})$:

1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_\mu, \mathsf{param})$
3. $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}_\mu, \mathsf{param})$
4. $W \in \mathcal{X} \setminus \mathcal{L}_\mu \leftarrow \mathcal{A}.\mathsf{choose}(\mathcal{L}_\mu, \mathsf{hp})$
5. $\mathcal{H}_{\mathsf{hk}} \xleftarrow{\$} \mathcal{V}$
6. $b' \leftarrow \mathcal{A}.\mathsf{guess}(\mathcal{L}_\mu, \mathcal{H}_{\mathsf{hk}}, \mathsf{hp}, W)$

Lors du passage au jeu G_4, la valeur du vecteur $\mathbf{s}$ est reconstruite comme un syndrome puis le jeu se poursuit normalement. Les arguments pour ce passage sont les mêmes que pour la transition entre les jeux G_1 et G_2.

Nous avons donc réussi à construire une séquence de jeux transitionnant de l'expérience $\mathsf{Exp}^{\mathsf{smooth}-0}_{\mathfrak{K}}(\mathcal{A})$ à $\mathsf{Exp}^{\mathsf{smooth}-1}_{\mathfrak{K}}(\mathcal{A})$, par conséquent notre Hash Proof System satisfait la KV-smoothness calculatoire sous l'hypothèse que les versions décisionnelles des problèmes 2-IRSD et FIRSD soient des problèmes difficiles. De plus, l'avantage de l'adversaire contre l'expérience $\mathsf{Exp}^{\mathsf{smooth}-\mathsf{b}}_{\mathcal{A}}(\mathfrak{K})$ est majoré par :

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{A}}(\mathfrak{K}) \leqslant 2 \times \mathsf{Adv}^{\mathsf{2\text{-}IRSD}}(\mathfrak{K}) + \mathsf{Adv}^{\mathsf{FIRSD}}(\mathfrak{K}).$$

□

3.5 Preuves de validité à divulgation nulle de connaissance

3.5.1 Preuve de validité d'un chiffré RQC

Dans cette section, nous allons décrire une preuve à divulgation nulle de connaissance (zero-knowledge) de la validité d'un chiffré pour le schéma de chiffrement RQC. Celui-ci permet à un prouveur de convaincre un vérifieur qu'un couple de valeurs donné correspond bien à un chiffré de RQC. Par la suite, nous étendrons cette preuve de façon à pouvoir démontrer que deux couples de valeurs correspondent à deux chiffrés d'un même message $\boldsymbol{\mu}$. Cette preuve de connaissance est essentielle pour la construction du PAKE (Password Authenticated Key Exchange) décrit en section 3.6.2.

Commençons par rappeler brièvement quelques résultats concernant les preuves de connaissance en théorie des codes correcteurs. Le système de preuve de Stern [Ste96] est un protocole interactif en trois passes basé sur la difficulté du problème de décodage par syndrome. Soit $\mathbf{H}$ une matrice de parité, $\mathbf{x}$ un vecteur de poids faible w_x et $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$ le syndrome associé. Ce protocole permet à un prouveur P de prouver à un vérifieur $\mathbf{v}$ la connaissance d'un vecteur $\mathbf{x}$ de poids faible w_x tel que $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$. Ce schéma à une parfaite completeness, malheureusement un prouveur malhonnête peut duper le vérifieur avec une probabilité égale à $\frac{2}{3}$. Il est donc nécessaire de répéter le schéma de nombreuses fois afin d'obtenir une probabilité d'erreur de soundness négligeable.

En 1995, Chen [Che96] a adapté le protocole de Stern à la métrique rang, la sécurité du schéma reposant désormais sur le problème du décodage par syndrome en métrique rang. En 2011, Gaborit et al. [GSZ11] ont montré comment casser le protocole de Chen de deux façons différentes et ont proposé une nouvelle version de celui-ci permettant de résoudre les failles qu'ils avaient mis en évidence. En 2016, une variante plus flexible fut proposée, intitulée *Rank Concatenated Stern's Protocol* ($\mathcal{RCSP}$) [ABCG16]. Pour permettre la conception d'une preuve de connaissance d'un chiffré de RQC, nous prolongeons les travaux entrepris précédemment sur le protocole de Stern en métrique rang et proposons une version encore plus flexible que le $\mathcal{RCSP}$.

La variante *Rank Concatenated Stern's Protocol* ($\mathcal{RCSP}$) proposée dans [ABCG16] permet à un prouveur de convaincre un vérifieur de sa connaissance de deux vecteurs de poids faibles. Plus précisement, soit $\mathbf{Q} \xleftarrow{\$} \mathbb{F}_{q^m}^{k\times n}$, $\mathbf{r} \xleftarrow{\$} \mathbb{F}_{q^m}^{k\times n'}$ et $\mathbf{x}$, $\mathbf{y}$ deux vecteurs de poids faible w_x et w_y vérifiant l'équation suivante :

$$\begin{pmatrix}\mathbf{Q} & \mathbf{R}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = \mathbf{s}.$$

La variante $\mathcal{RCSP}$ permet alors de prouver la connaissance des deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ de poids faibles respectifs w_x et w_y vérifiant le syndrome décrit précédemment, et ce sans apport de connaissance (zero-knowledge proof).

Pour permettre une plus grande flexibilité du protocole, nous allons avoir besoin de contraindre les blocs de manière à pouvoir les forcer à être de même rang et de même support qu'un autre bloc ou de les rendre totalement indépendants des autres blocs selon le besoin. Considérons l'expression suivante :

$$\begin{pmatrix} \mathbf{I}_n & \mathcal{I}(\mathbf{h}) & 0 & 0 \\ 0 & \mathcal{I}(\mathbf{s}) & \mathbf{I}_n & \mathrm{Gab}_{\mathbf{g}} \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix}.$$

Le produit matriciel correspond au couple composant un chiffré de RQC. Le prouveur doit donc pouvoir être capable de démontrer que $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ vérifient bien l'égalité précédente, que $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$ et qu'il n'y ait aucune condition sur $\boldsymbol{\mu}$ car le support de $\boldsymbol{\mu}$ est indépendant du support commun à $\mathbf{r}_1, \mathbf{r}_2$ et $\mathbf{r}_3$.

Donnons maintenant quelques définitions et propositions afin d'atteindre cet objectif.

Définition 49. *Pour $\mathbf{x}, \mathbf{y} \in \mathcal{V}$, on dit que $\mathbf{x}$ et $\mathbf{y}$ sont équivalents, noté $\mathbf{x} \sim \mathbf{y}$, si $\mathsf{Supp}(\mathbf{x}) = \mathsf{Supp}(\mathbf{y})$.*

Comme expliqué au chapitre 2, à tout vecteur $\mathbf{x} \in \mathcal{V}$ on peut associer la matrice $\mathbf{M}_{\mathbf{x}} \in \mathbb{F}_q^{m \times n}$ pour une base $\mathcal{B} = \{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$ de $\mathbb{F}_{q^m}$ donnée. Rappelons que nous notons $\phi_{\mathcal{B}}$ l'application qui associe à tout vecteur de $\mathcal{V}$ sa matrice associée dans $\mathbb{F}_q^{m \times n}$.

$$\begin{array}{ccccc} \phi_{\mathcal{B}} & : & \mathbb{F}_{q^m}^n & \simeq & \mathbb{F}_q^{m \times n} \\ & & \mathbf{v} = (v_0, \ldots, v_{n-1}) & \mapsto & \mathbf{M}_{\mathbf{v}} = \begin{pmatrix} v_{1,0} & \cdots & v_{1,n-1} \\ v_{2,0} & \cdots & v_{2,n-1} \\ \vdots & & \vdots \\ v_{m,0} & \cdots & v_{m,n-1} \end{pmatrix} \begin{matrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_m. \end{matrix} \end{array}$$

Définition 50. *Soit $\mathbf{Q} \in \mathsf{GL}_m(q)$, $\mathbf{v} \in \mathcal{V}$ et $\mathcal{B}$ une base de $\mathbb{F}_{q^m}$. On définit le produit $\mathbf{Q} * \mathbf{v}$ comme étant $\mathbf{Q} * \mathbf{v} \triangleq \phi_{\mathcal{B}}^{-1}(\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{v}))$.*

Lemme 1. *Pour tout $\mathbf{v} \in \mathcal{V}$, $\mathbf{P} \in \mathsf{GL}_n(q)$ et $\mathbf{Q} \in \mathsf{GL}_m(q)$, on a :*

$$\phi_{\mathcal{B}}(\mathbf{v} \cdot \mathbf{P}) = \phi_{\mathcal{B}}(\mathbf{v}) \cdot \mathbf{P}.$$

Démonstration. Soit $\mathbf{v} \in \mathcal{V}$, $\mathbf{P} = (p_{ij}) \in \mathsf{GL}_n(q)$ et $\mathbf{Q} = (q_{ij}) \in \mathsf{GL}_m(q)$. Notons $\mathbf{v} = (v_1, \ldots, v_n)$ et, pour tout $i \in [\![1, n]\!]$, $v_i = \sum_{i=0}^{m} v_{ik} \boldsymbol{\beta}_k$ (décomposition de $\mathbf{v}$ dans la base $\mathcal{B}$).

$$
\begin{aligned}
\phi_{\mathcal{B}}(\mathbf{v}\cdot\mathbf{P}) &= \phi_{\mathcal{B}}(\sum_{i=1}^{n} v_i p_{i1}, \ldots, \sum_{i=1}^{n} v_i p_{in}) \\
&= \phi_{\mathcal{B}}(\sum_{i=1}^{n}\sum_{k=1}^{m} v_{ik}\beta_k p_{i1}, \ldots, \sum_{i=1}^{n}\sum_{k=1}^{m} v_{ik}\beta_k p_{in}) \\
&= \phi_{\mathcal{B}}(\sum_{k=1}^{m} \beta_k \sum_{i=1}^{n} v_{ik} p_{i1}, \ldots, \sum_{k=1}^{m} \beta_k \sum_{i=1}^{n} v_{ik}\ p_{in}) \\
&= \begin{pmatrix} \sum_{i=1}^{n} v_{i1} p_{i1} & \cdots & \sum_{i=1}^{n} v_{i1} p_{in} \\ \vdots & & \vdots \\ \sum_{i=1}^{n} v_{im} p_{i1} & \cdots & \sum_{i=1}^{n} v_{im} p_{in} \end{pmatrix} \\
&= \phi_{\mathcal{B}}(\mathbf{v})\cdot\mathbf{P}.
\end{aligned}
$$

□

Proposition 19. *Pour tout* $\mathbf{v} \in \mathcal{V}$, $\mathbf{P} \in \mathsf{GL}_n(q)$ *et* $\mathbf{Q} \in \mathsf{GL}_m(q)$, *on a :*

$$(\mathbf{Q} * \mathbf{v})\cdot\mathbf{P} = \mathbf{Q} * (\mathbf{v}\cdot\mathbf{P}).$$

Démonstration. Soit $\mathbf{v} \in \mathcal{V}$, $\mathbf{P} \in \mathsf{GL}_n(q)$ et $\mathbf{Q} \in \mathsf{GL}_m(q)$.

$$
\begin{aligned}
(\mathbf{Q} * \mathbf{v})\cdot\mathbf{P} &= (\mathbf{Q}\cdot\phi_{\mathcal{B}}(\mathbf{v}))\cdot\mathbf{P} \\
&= \mathbf{Q}\cdot(\phi_{\mathcal{B}}(\mathbf{v})\cdot\mathbf{P}) \\
&= \mathbf{Q}\cdot\phi_{\mathcal{B}}(\mathbf{v}\cdot\mathbf{P}) \\
&= \mathbf{Q} * (\mathbf{v}\cdot\mathbf{P}).
\end{aligned}
$$

□

Lemme 2. *Pour tout* $\mathbf{x} \in \mathcal{V}$ *et* $\mathbf{Q} \in \mathsf{GL}_m(q)$, $\|\mathbf{Q} * \mathbf{x}\| = \|\mathbf{x}\|$.

Démonstration. Soit $\mathbf{x} \in \mathcal{V}$ et $\mathbf{Q} \in \mathsf{GL}_m(q)$. Commençons par prouver que $\mathbf{Q} * \mathbf{x}$ ne peut pas avoir un rang supérieur à $\mathbf{x}$.

Multiplier $\mathbf{x}$ à gauche par $\mathbf{Q}$ revient à effectuer des combinaisons linéaires sur les lignes de $\phi_{\mathbf{x}}$. Supposons par l'absurde que le rang de $\mathbf{Q} * \mathbf{x}$ soit strictement supérieur au rang de $\mathbf{x}$. Cela signifie qu'au moins une des lignes de $\mathbf{Q} * \mathbf{x}$ n'est pas une combinaison linéaire des lignes de $\phi_{\mathbf{x}}$. Mais comme multiplier à gauche

par $\mathbf{Q}$ revient à faire des combinaisons linéaires des lignes de $\phi_{\mathbf{x}}$, cette situation est impossible. Contradiction. Ainsi le rang de $\mathbf{Q} * \mathbf{x}$ est inférieur ou égal au rang de $\mathbf{x}$.

Maintenant, montrons que le rang de $\mathbf{Q} * \mathbf{x}$ ne peut pas être strictement inférieur à $\mathbf{x}$ non plus. Notons w_x le rang de $\mathbf{x}$ ($w_x < \min(m, n)$). Supposons par l'absurde que $\|\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{x})\| < w_x$. Comme $\mathbf{Q} \in \mathsf{GL}_m(q)$, il existe une matrice inverse $\mathbf{Q}^{-1} \in \mathsf{GL}_m(q)$ telle que $\mathbf{Q}^{-1} \cdot \mathbf{Q} = \mathbf{I}_m$. On sait que $\|\mathbf{Q}^{-1} \cdot \mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{x})\| = \|\phi_{\mathcal{B}}(\mathbf{x})\|$. Comme $\|\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{x})\| < w_x$, cela signifie que multiplier à gauche par $\mathbf{Q}^{-1}$ a augmenté le rang de $\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{x})$. Contradiction. Ainsi, le rang de $\mathbf{Q} * \mathbf{x}$ est supérieur ou égal au rang de $\mathbf{x}$, et par suite $\|\mathbf{Q} * \mathbf{x}\| = \|\mathbf{x}\|$. □

Lemme 3. *Pour tout* $\mathbf{x} \in \mathcal{V}$ *et* $\mathbf{P} \in \mathsf{GL}_n(q)$, $\|\mathbf{x} \cdot \mathbf{P}\| = \|\mathbf{x}\|$.

Démonstration. Les démonstrations sont similaires aux précédentes, en raisonnant sur les colonnes de $\phi_{\mathcal{B}}(\mathbf{x})$ plutôt que sur les lignes. □

Corollaire 1. *Pour tout* $\mathbf{x} \in \mathcal{V}$, $\mathbf{Q} \in \mathsf{GL}_m(q)$ *et* $\mathbf{P} \in \mathsf{GL}_n(q)$, $\|\mathbf{Q} * \mathbf{x} \cdot \mathbf{P}\| = \|\mathbf{x}\|$.

Démonstration. Ce résultat est une conséquence directe des deux lemmes précédents. □

Proposition 20. *Pour tout* $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ *avec* $\|\mathbf{x}\| = \|\mathbf{y}\|$, *il existe* $\mathbf{Q} \in \mathsf{GL}_m(q)$ *et* $\mathbf{P} \in \mathsf{GL}_n(q)$ *telles que* $\mathbf{y} = \mathbf{Q} * \mathbf{x}\mathbf{P}$.

Démonstration. Soit $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ avec $\|\mathbf{x}\| = \|\mathbf{y}\| = r$ (avec nécessairement $r < \min(m, n)$). Il existe $\mathbf{P}_1 \in \mathsf{GL}_n(q)$ telle que $\phi_{\mathcal{B}}(\mathbf{x}) \cdot \mathbf{P}_1$ est sous forme systématique avec ses r premières colonnes non nulles. Il existe $\mathbf{P}_2 \in \mathsf{GL}_n(q)$ telle que $\phi_{\mathcal{B}}(\mathbf{y}) \cdot \mathbf{P}_2$ est sous forme systématique avec ses r premières colonnes non nulles. Les premières r colonnes de $\phi_{\mathcal{B}}(\mathbf{x}) \cdot \mathbf{P}_1$ forment une base $\mathcal{B}_1$ de $\mathbb{F}_{q^m}^r$. Les premières r colonnes de $\phi_{\mathcal{B}}(\mathbf{y}) \cdot \mathbf{P}_2$ forment une autre base $\mathcal{B}_2$ de $\mathbb{F}_{q^m}^r$. Ainsi, si l'on note $\mathbf{Q}$ la matrice de transition qui transforme la base $\mathcal{B}_1$ en la base $\mathcal{B}_2$, nous avons alors $\mathbf{Q} \cdot \mathbf{x} \cdot \mathbf{P}_1 = \mathbf{y} \cdot \mathbf{P}_2$. Finalement, $\mathbf{y} = \mathbf{Q} * \mathbf{x} \cdot (\mathbf{P}_1 \cdot \mathbf{P}_2^{-1})$. □

Dans la version $\mathcal{RCSP}$, un élément $\mathbf{v} \in \mathcal{V}$ est transformé en n'importe quel élément de même rang en évaluant l'expression $\mathbf{Q} * \mathbf{v}\mathbf{P}$ avec $\mathbf{Q} \in \mathsf{GL}_m(q)$ et $\mathbf{P} \in \mathsf{GL}_n(q)$. Pour chaque bloc, une matrice distincte $\mathbf{Q}$ et $\mathbf{P}$ était utilisée. Dans

notre cas, nous voudrions pouvoir conserver la relation qui lie le support des vecteurs de chacun des blocs entre-eux, pas seulement leur rang. Cette propriété est assurée par les propositions ci-dessous.

Lemme 4. *Pour tout* $\mathbf{x} \in \mathcal{V}$*, pour tout* $\mathbf{P} \in \mathsf{GL}_n(q)$*,* $\mathbf{x} \sim \mathbf{x} \cdot \mathbf{P}$*.*

Démonstration. Soit $\mathbf{x} \in \mathcal{V}$ et $\mathbf{P} \in \mathsf{GL}_n(q)$. Notons r le rang de $\mathbf{x}$, et $\mathcal{B}_\gamma = \{\gamma_1, \ldots, \gamma_r\}$ une base du support de $\mathbf{x}$. Comme multiplier à droite par $\mathbf{P}$ est équivalent à effectuer des combinaisons linéaires des colonnes de $\phi_{\mathcal{B}}(\mathbf{x})$, $\mathcal{B}_\gamma$ génère toutes les coordonnées de $\mathbf{x} \cdot \mathbf{P}$. Comme nous avons déjà prouvé précédemment que $\|\mathbf{x}\| = \|\mathbf{x} \cdot \mathbf{P}\|$, $\mathcal{B}_\gamma$ est également une base du support de $\mathbf{x} \cdot \mathbf{P}$. Par conséquent $\mathbf{x} \sim \mathbf{x} \cdot \mathbf{P}$. □

Proposition 21. *Pour tout* $\mathbf{x}, \mathbf{y} \in \mathcal{V}$*, pour tout* $\mathbf{Q} \in \mathsf{GL}_m(q)$*, pour tout* $\mathbf{P}_1, \mathbf{P}_2 \in \mathsf{GL}_n(q)$*, si* $\mathbf{x} \sim \mathbf{y}$ *alors* $(\mathbf{Q} * \mathbf{x}\mathbf{P}_1) \sim (\mathbf{Q} * \mathbf{y}\mathbf{P}_2)$*.*

Démonstration. Soit $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ tels que $\mathbf{x} \sim \mathbf{y}$. Soit $\mathbf{Q} \in \mathsf{GL}_m(q)$ et $\mathbf{P}_1, \mathbf{P}_2 \in \mathsf{GL}_n(q)$.

Prouvons dans un premier temps que $\mathbf{Q} * \mathbf{x} \sim \mathbf{Q} * \mathbf{y}$. Notons $\mathcal{B}_\gamma = \{\gamma_1, \ldots, \gamma_r\}$ une base du support commun à $\mathbf{x}$ et $\mathbf{y}$. Chaque colonne de $\phi_{\mathcal{B}}(\mathbf{x})$ et $\phi_{\mathcal{B}}(\mathbf{y})$ est une combinaison linéaire des vecteurs de $\mathcal{B}_\gamma$.

$$\phi_{\mathcal{B}}(\mathbf{x}) = \left(\sum_{k=0}^{r} x_{1k}\gamma_k \mid \ldots \mid \sum_{k=0}^{r} x_{nk}\gamma_k\right).$$

$$\phi_{\mathcal{B}}(\mathbf{y}) = \left(\sum_{k=0}^{r} y_{1k}\gamma_k \mid \ldots \mid \sum_{k=0}^{r} y_{nk}\gamma_k\right).$$

$$\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{x}) = \left(\sum_{k=0}^{r} x_{1k} \cdot \mathbf{Q} \cdot \gamma_k \mid \ldots \mid \sum_{k=0}^{r} x_{nk} \cdot \mathbf{Q} \cdot \gamma_k\right).$$

$$\mathbf{Q} \cdot \phi_{\mathcal{B}}(\mathbf{y}) = \left(\sum_{k=0}^{r} y_{1k} \cdot \mathbf{Q} \cdot \gamma_k \mid \ldots \mid \sum_{k=0}^{r} y_{nk} \cdot \mathbf{Q} \cdot \gamma_k\right).$$

Par conséquent, $\{\mathbf{Q} \cdot \gamma_1, \ldots, \mathbf{Q} \cdot \gamma_k\}$ est un support commun à $\mathbf{Q} * \mathbf{x}$ and $\mathbf{Q} * \mathbf{y}$, ce qui signifie que $\mathbf{Q} * \mathbf{x} \sim \mathbf{Q} * \mathbf{y}$.

Multiplier un vecteur à droite par une matrice de $\mathsf{GL}_n(q)$ ne changeant pas son support comme prouvé dans un lemme précédent, il vient $\mathbf{Q} * \mathbf{x}\mathbf{P}_1 \sim \mathbf{Q} * \mathbf{y}\mathbf{P}_2$. □

Proposition 22. *Pour tout* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2 \in \mathcal{V}$ *de rang* $r \in \mathbb{N}$ *tels que* $\mathbf{x}_1 \sim \mathbf{x}_2$ *et* $\mathbf{y}_1 \sim \mathbf{y}_2$, *il existe* $\mathbf{Q} \in \mathsf{GL}_m(q)$ *et* $\mathbf{P}_1, \mathbf{P}_2 \in \mathsf{GL}_n(q)$ *telles que* $\mathbf{y}_1 = \mathbf{Q} * \mathbf{x}_1\mathbf{P}_1$ *et* $\mathbf{y}_2 = \mathbf{Q} * \mathbf{x}_2\mathbf{P}_2$.

Démonstration. Soit $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2 \in \mathcal{V}$ de rang $r \in \mathbb{N}$ tels que $\mathbf{x}_1 \sim \mathbf{x}_2$ et $\mathbf{y}_1 \sim \mathbf{y}_2$.

Il existe $\mathbf{P}_1 \in \mathsf{GL}_n(q)$ telle que $\mathbf{x}_1\mathbf{P}_1$ est sous forme systématique avec ses r premières colonnes non nulles. Il existe $\mathbf{P}_2 \in \mathsf{GL}_n(q)$ telle que $\mathbf{x}_2\mathbf{P}_2$ est sous forme systématique, et telle que ses r premières colonnes soient égales aux r premières colonnes de $\mathbf{x}_1\mathbf{P}_1$, le reste étant des colonnes nulles.

De façon similaire, il existe $\mathbf{P}_3 \in \mathsf{GL}_n(q)$ telle que $\mathbf{y}_1\mathbf{P}_3$ est sous forme systématique avec ses r premières colonnes non nulles. Il existe $\mathbf{P}_4 \in \mathsf{GL}_n(q)$ telle que $\mathbf{y}_2\mathbf{P}_4$ est sous forme systématique, et telle que ses r premières colonnes soient égales aux r premières colonnes de $\mathbf{y}_1\mathbf{P}_3$, le reste étant des colonnes nulles.

Les r premières colonnes de $\mathbf{x}_1\mathbf{P}_1$ forment une base $\mathcal{B}_1$ de $\mathbb{F}_{q^m}^r$ et les r premières colonnes de $\mathbf{y}_1\mathbf{P}_3$ forment une base $\mathcal{B}_2$ de $\mathbb{F}_{q^m}^r$. Notons $\mathbf{Q}$ la matrice de passage de la base $\mathbf{v}_1$ à $\mathbf{v}_2$. On a alors $\mathbf{Q} * \mathbf{x}_1\mathbf{P}_1 = \mathbf{y}_1\mathbf{P}_3 = \mathbf{y}_2\mathbf{P}_4 = \mathbf{Q} * \mathbf{x}_2\mathbf{P}_2$. Au final, $\mathbf{y}_1 = \mathbf{Q} * \mathbf{x}_1(\mathbf{P}_1\mathbf{P}_3^{-1})$ et $\mathbf{y}_2 = \mathbf{Q} * \mathbf{x}_2(\mathbf{P}_2\mathbf{P}_4^{-1})$. □

En se basant sur les résultats précédents, nous pouvons désormais concevoir une preuve à divulgation nulle de connaissance pour un chiffré de RQC. Les chiffrés de RQC sont de la forme suivante :

$$\begin{pmatrix} \mathbf{I}_n & \mathcal{I}(\mathbf{H}) & 0 & 0 \\ 0 & \mathcal{I}(\mathbf{s}) & \mathbf{I}_n & \mathrm{Gab}_{\mathbf{g}} \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix} \tag{3.1}$$

où $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$ et $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$.

Décrivons le protocole correspond à ces expressions. La paire $(\mathbf{H}, \mathbf{c})$ est donnée en entrée au prouveur et au vérifieur, le prouveur dispose également du vecteur $\mathbf{r}$ tel que $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \boldsymbol{\mu})^\top$ et de $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$. $\mathbf{H}$ sera divisé en plusieurs sous-matrices :

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{H}_4 \end{pmatrix}$$

où : $\mathbf{H}_1 = \begin{pmatrix} \mathbf{I}_n \\ 0 \end{pmatrix}$ $\mathbf{H}_2 = \begin{pmatrix} \mathcal{I}(\mathbf{H}) \\ \mathcal{I}(\mathbf{s}) \end{pmatrix}$ $\mathbf{H}_3 = \begin{pmatrix} 0 \\ \mathbf{I}_n \end{pmatrix}$ $\mathbf{H}_4 = \begin{pmatrix} 0 \\ \mathrm{Gab}_{\mathbf{g}} \end{pmatrix}$.

En utilisant ces notations, l'équation (3.1) est équivalente à $\mathbf{H}\mathbf{r}^\top = \mathbf{c}^\top$. Le prouveur P et le vérifieur V intéragissent ensuite comme dans la figure 3.4.

Notations : Nous noterons $a \mid b$ la concaténation de deux éléments et par $\|_{i=1}^{n} a_i = a_1 \mid \ldots \mid a_n$ la concaténation d'une suite d'éléments.

1. [Commitment] P génère $\mathbf{Q} \xleftarrow{\$} \mathsf{GL}_m(q)$; $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3 \xleftarrow{\$} \mathsf{GL}_n(q)$; $\mathbf{P}_4 \xleftarrow{\$} \mathsf{GL}_k(q)$; $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \xleftarrow{\$} \mathcal{V}$; $\mathbf{v}_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k$; $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \xleftarrow{\$} 1^{\mathfrak{K}}$ et envoie le commitment CMT :=(cm_1, cm_2, cm_3), où
   - $cm_1 = \mathrm{Hash}(\mathbf{Q} \mid \|_{i=1}^{4} \mathbf{P}_i \mid \sum_{i=1}^{4} \mathbf{H}_i \mathbf{v}_i^\top \mid \mathbf{z}_1)$
   - $cm_2 = \mathrm{Hash}(\|_{i=1}^{4} \mathbf{Q} * \mathbf{v}_i \mathbf{P}_i \mid \mathbf{z}_2)$
   - $cm_3 = \mathrm{Hash}(\|_{i=1}^{3} \mathbf{Q} * (\mathbf{v}_i + \mathbf{r}_i)\mathbf{P}_i \mid \mathbf{Q} * (\mathbf{v}_4 + \boldsymbol{\mu})\mathbf{P}_4 \mid \mathbf{z}_3)$
2. [Challenge] V envoie un challenge aléatoire $ch \in \{0, 1, 2\}$ à P.
3. [Réponse] P répond de la façon suivante :
   - Si $ch = 0$, soit $\mathbf{E} = \mathbf{Q}$, $\forall i \in [\![1, 4]\!]$, soit $\mathbf{a}_i = \mathbf{v}_i$, $\mathbf{F}_i = \mathbf{P}_i$.
     Renvoyer RSP :=$(\mathbf{E} \mid \|_{i=1}^{4} \mathbf{F}_i \mid \|_{i=1}^{4} \mathbf{a}_i \mid \mathbf{z}_1 \mid \mathbf{z}_2)$
   - Si $ch = 1$, soit $\mathbf{I} = \mathbf{Q}$, $\mathbf{b}_4 = \mathbf{v}_4 + \boldsymbol{\mu}$, $\forall i \in [\![1, 3]\!]$, soit $\mathbf{b}_i = \mathbf{v}_i + \mathbf{r}_i$, $\mathbf{J}_i = \mathbf{P}_i$.
     Renvoyer RSP :=$(\mathbf{I} \mid \|_{i=1}^{4} \mathbf{J}_i \mid \|_{i=1}^{4} \mathbf{b}_i \mid \mathbf{z}_1 \mid \mathbf{z}_3)$
   - Si $ch = 2$, $\forall i \in [\![1, 4]\!]$, soit $\mathbf{c}_i = \mathbf{Q} * \mathbf{v}_i \mathbf{P}_i$, $\forall i \in [\![1, 3]\!]$, soit $\mathbf{d}_i = \mathbf{Q} * \mathbf{r}_i \mathbf{P}_i$ et soit $\mathbf{d}_4 = \mathbf{Q} * \boldsymbol{\mu} \mathbf{P}_4$.
     Renvoyer RSP :=$(\|_{i=1}^{4} \mathbf{c}_i \mid \|_{i=1}^{4} \mathbf{d}_i \mid \mathbf{z}_2 \mid \mathbf{z}_3)$
4. [Vérification] V effectue les vérifications suivantes :
   - Si $ch = 0$, vérifier que
     - $cm_1 = \mathrm{Hash}(\mathbf{E} \mid \|_{i=1}^{4} \mathbf{F}_i \mid \sum_{i=1}^{4} \mathbf{H}_i \mathbf{a}_i^\top \mid \mathbf{z}_1)$
     - $cm_2 = \mathrm{Hash}(\|_{i=1}^{4} \mathbf{E} * \mathbf{a}_i \mathbf{F}_i \mid \mathbf{z}_2)$
   - Si $ch = 1$, vérifier que
     - $cm_1 = \mathrm{Hash}(\mathbf{I} \mid \|_{i=1}^{4} \mathbf{J}_i \mid \sum_{i=1}^{4} \mathbf{H}_i \mathbf{b}_i^\top - \mathbf{c}^\top \mid \mathbf{z}_1)$
     - $cm_3 = \mathrm{Hash}(\|_{i=1}^{4} \mathbf{I} * \mathbf{b}_i \mathbf{J}_i \mid \mathbf{z}_3)$
   - Si $ch = 2$, vérifier que
     - $cm_2 = \mathrm{Hash}(\|_{i=1}^{4} \mathbf{c}_i \mid \mathbf{z}_2)$
     - $cm_3 = \mathrm{Hash}(\|_{i=1}^{4} \mathbf{c}_i + \mathbf{d}_i \mid \mathbf{z}_3)$
     - $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$

FIGURE 3.4 – Preuve de validité d'un chiffré de RQC

Définition 51. *Un protocole interactif entre deux machines probabilistes en temps polynomial* P *et* V *est dit statistiquement à divulgation nulle de connaissance (zero-*

knowledge) si, pour toute machine probabiliste en temps polynomial $\tilde{V}$, il existe une machine S qui renvoie, en temps polynomial, une valeur ayant une distribution statistiquement indistinguable des valeurs renvoyées lors des communications entre P *et* $\tilde{V}$.

Théorème 3. *Le protocole décrit en figure 3.4 est statistiquement à divulgation nulle de connaissance dans le modèle de l'oracle aléatoire.*

Démonstration. 3 La preuve utilise des techniques dans le même esprit que dans [Ste96, KTX08, LNSW13]. Nous allons donc construire un simulateur S qui, étant données les valeurs publiques du protocole en entrée et interagissant avec un vérifieur malhonnête $\widetilde{\mathsf{V}}$, retourne une retranscription simulée avec une probabilité de $\frac{2}{3}$ qui est statistiquement proche de la distribution obtenue avec des échanges réels. Les valeurs publiques en entrée sont $(\mathbf{H}, \mathbf{c})$, le simulateur S commence par choisir un challenge aléatoire $\bar{ch} \in \{0, 1, 2\}$, qui représente une prédiction du challenge que le vérifieur $\widetilde{\mathsf{V}}$ ne choisira pas.

$\diamondsuit$ **Premier cas** $\bar{ch} = 0$ **:**

S génère :

$$\mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q) \qquad \mathbf{P}'_1, \mathbf{P}'_2, \mathbf{P}'_3 \xleftarrow{\$} \mathsf{GL}_n(q)\,; \qquad \mathbf{P}'_4 \xleftarrow{\$} \mathsf{GL}_k(q)\,;$$
$$\mathbf{v}'_1, \mathbf{v}'_2, \mathbf{v}'_3 \xleftarrow{\$} \mathcal{V}\,; \qquad \mathbf{v}'_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,;$$
$$(\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3) \xleftarrow{\$} \mathfrak{S}_{w_r}^3(\mathcal{V})\,; \qquad \mathbf{r}'_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,; \qquad \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\mathfrak{K}}\,;$$

et envoie le commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

- $cm'_1 = \mathrm{Hash}(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \sum_{i=1}^4 \mathbf{H}_i(\mathbf{v}'^{\top}_i + \mathbf{r}'^{\top}_i) \mid \mathbf{z}'_1)$
- $cm'_2 = \mathrm{Hash}(\|_{i=1}^4 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{z}'_2)$
- $cm'_3 = \mathrm{Hash}(\|_{i=1}^4 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3)$.

Après avoir reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la manière suivante :

- Si $ch = 0$, retourner $\perp$ et interrompre.
- Si $ch = 1$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \|_{i=1}^4 \mathbf{v}'_i + \mathbf{r}'_i \mid \mathbf{z}'_1 \mid \mathbf{z}'_3)$
- Si $ch = 2$, retourner RSP :=$(\|_{i=1}^4 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \|_{i=1}^4 \mathbf{Q}' * \mathbf{r}'_i \mathbf{P}'_i \mid \mathbf{z}'_2 \mid \mathbf{z}'_3)$.

$\diamondsuit$ **Deuxième cas** $\bar{ch} = 1$ **:**

S génère :

$\mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q)$; $\quad \mathbf{P}'_1, \mathbf{P}'_2, \mathbf{P}'_3 \xleftarrow{\$} \mathsf{GL}_n(q)$; $\quad \mathbf{P}'_4 \xleftarrow{\$} \mathsf{GL}_k(q)$;
$\mathbf{v}'_1, \mathbf{v}'_2, \mathbf{v}'_3 \xleftarrow{\$} \mathcal{V}$; $\quad \mathbf{v}'_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k$;
$(\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3) \xleftarrow{\$} \mathfrak{S}^3_{w_r}(\mathcal{V})$; $\quad \mathbf{r}'_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k$; $\quad \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\mathfrak{K}}$;

et envoie le commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

$$cm'_1 = \text{Hash}(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \sum_{i=1}^4 \mathbf{H}_i \mathbf{v}_i'^{\top} \mid \mathbf{z}'_1)$$
$$cm'_2 = \text{Hash}(\|_{i=1}^4 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{z}'_2)$$
$$cm'_3 = \text{Hash}(\|_{i=1}^4 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3).$$

Après avoir reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la manière suivante :

- $\diamond$ Si $ch = 0$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \|_{i=1}^4 \mathbf{v}'_i \mid \mathbf{z}'_1 \mid \mathbf{z}'_2)$
- $\diamond$ Si $ch = 1$, retourner $\perp$ et interrompre.
- $\diamond$ Si $ch = 2$, retourner RSP :=$(\|_{i=1}^4 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \|_{i=1}^4 \mathbf{Q}' * \mathbf{r}'_i \mathbf{P}'_i \mid \mathbf{z}'_2 \mid \mathbf{z}'_3)$.

$\diamondsuit$ **Troisième cas** $\bar{ch} = 2$ **:**

En utilisant de l'algèbre linéaire, S calcule un vecteur $\mathbf{r}' = (\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3, \mathbf{r}'_4)$, tel que $\mathbf{H}\mathbf{r}'^{\top} = \mathbf{c}^{\top}$. Puis, il génère :

$\mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q)$; $\quad \mathbf{P}'_1, \mathbf{P}'_2, \mathbf{P}'_3 \xleftarrow{\$} \mathsf{GL}_n(q)$; $\quad \mathbf{P}'_4 \xleftarrow{\$} \mathsf{GL}_k(q)$;
$\mathbf{v}'_1, \mathbf{v}'_2, \mathbf{v}'_3 \xleftarrow{\$} \mathcal{V}$; $\quad \mathbf{v}'_4 \xleftarrow{\$} \mathbb{F}_{q^m}^k$; $\quad \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\mathfrak{K}}$;

et envoie le commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

$$cm'_1 = \text{Hash}(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \sum_{i=1}^4 \mathbf{H}_i \mathbf{v}_i'^{\top} \mid \mathbf{z}'_1)$$
$$cm'_2 = \text{Hash}(\|_{i=1}^4 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{P}'_4 \mid \mathbf{z}'_2)$$
$$cm'_3 = \text{Hash}(\|_{i=1}^4 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3).$$

Après avoir reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la manière suivante :

- $\diamond$ Si $ch = 0$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^4 \mathbf{P}'_i \mid \|_{i=1}^4 \mathbf{v}'_i \mid \mathbf{z}'_1 \mid \mathbf{z}'_2)$

- ◇ Si $ch = 1$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^{4}\mathbf{P}'_i \mid \|_{i=1}^{4}\mathbf{v}'_i + \mathbf{r}'_i \mid \mathbf{z}_1 \mid \mathbf{z}_3)$
- ◇ Si $ch = 2$, retourner $\perp$ et interrompre.

On peut remarquer que la probabilité que le simulateur renvoie $\perp$ est égale à $\frac{1}{3}$. De plus, lorsque le simulateur n'est pas interrompu, la distribution générée est statistiquement proche de celle obtenue lors des échanges réels lorsque la fonction de hachage est modélisée à l'aide d'un oracle aléatoire. Par conséquent, nous avons construit un simulateur qui parvient à simuler le protocole en temps polynomial avec une probabilité de $\frac{2}{3}$ sans avoir la connaissance des valeurs secrètes du protocole en question. □

Théorème 4. *S'il existe un prouveur malhonnête disposant d'un algorithme probabiliste en temps polynomial noté* $\widetilde{\mathsf{P}}$ *capable de convaincre un vérifieur avec une probabilité de* $\frac{2}{3} + \varepsilon$*, où* ε *est non négligeable, alors il existe un extracteur de connaissance probabiliste en temps polynomial qui renvoie, avec une écrasante probabilité,* $(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4)$ *tels que* $\sum_{i=1}^{4} \mathbf{H}_i \mathbf{y}_i^\top = \mathbf{c}^\top$ *et* $(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \in \mathfrak{S}_{w_r}^{3}(\mathcal{V})$.

Démonstration. Nous allons montrer comment construire un extracteur de connaissance $\mathcal{K}$. Soit $\widetilde{\mathsf{P}}$ le prouveur malhonnête capable de convaincre le vérifieur avec une probabilité de $\frac{2}{3} + \varepsilon$. En appliquant la technique de Véron [Vér97], qui rembobine $\widetilde{\mathsf{P}}$ un nombre de fois polynomial en $\frac{1}{\varepsilon}$, l'extracteur de connaissance pourra obtenir avec une probabilité écrasante un commitment avec lequel $\widetilde{\mathsf{P}}$ peut répondre de façon correcte aux trois challenges. Dans un premier temps, $\mathcal{K}$ obtient les équations suivantes :

- ◇ $cm_1 = \text{Hash}(\mathbf{E} \mid \|_{i=1}^{4}\mathbf{F}_i \mid \sum_{i=1}^{4} \mathbf{H}_i \mathbf{a}_i^\top) = \text{Hash}(\mathbf{I} \mid \|_{i=1}^{4}\mathbf{J}_i \mid \sum_{i=1}^{4} \mathbf{H}_i \mathbf{v}_i^\top - \mathbf{c}^\top)$
- ◇ $cm_2 = \text{Hash}(\|_{i=1}^{4}\mathbf{E} * \mathbf{a}_i\mathbf{F}_i) = \text{Hash}(\|_{i=1}^{4}\mathbf{c}_i)$
- ◇ $cm_3 = \text{Hash}(\|_{i=1}^{4}\mathbf{I} * \mathbf{v}_i\mathbf{J}_i) = \text{Hash}(\|_{i=1}^{4}\mathbf{c}_i + \mathbf{d}_i)$
- ◇ $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in \mathfrak{S}_{w_r}^{3}(\mathcal{V})$.

Comme la fonction Hash est modélisée par le modèle de l'oracle aléatoire (un adversaire est incapable de trouver une collision via cet oracle), on a alors :

- ◇ $\mathbf{E} = \mathbf{I}$ et $\forall i \in [\![1, 4]\!]$, $\mathbf{F}_i = \mathbf{J}_i$ et $\sum_{i=1}^{4} \mathbf{H}_i \mathbf{a}_i^\top = \sum_{i=1}^{4} \mathbf{H}_i \mathbf{v}_i^\top - \mathbf{c}^\top$
- ◇ $\forall i \in [\![1, 4]\!]$, $\mathbf{E} * \mathbf{a}_i\mathbf{F}_i = \mathbf{c}_i$, $\mathbf{I} * \mathbf{v}_i\mathbf{J}_i = \mathbf{c}_i + \mathbf{d}_i$

Soit $i \in [\![1,4]\!]$. On a $\mathbf{I} * \mathbf{v}_i\mathbf{J}_i = \mathbf{E} * \mathbf{v}_i\mathbf{F}_i = \mathbf{c}_i + \mathbf{d}_i$. Il s'ensuit $\mathbf{E} * (\mathbf{v}_i - \mathbf{a}_i)\mathbf{F}_i = \mathbf{d}_i$, ce qui implique que $(\mathbf{v}_i - \mathbf{a}_i) = \mathbf{E}^{-1} * \mathbf{d}_i\mathbf{F}_i^{-1}$.

$$\forall i \in [\![1,4]\!],\ (\mathbf{v}_i - \mathbf{a}_i) = \mathbf{E}^{-1} * \mathbf{d}_i\mathbf{F}_i^{-1}$$

Comme $\mathbf{E}^{-1} \in \mathsf{GL}_m(q)$ et $\mathbf{F}_i^{-1} \in \mathsf{GL}_m(q)$, on a $(\mathbf{v}_1 - \mathbf{a}_1, \mathbf{v}_2 - \mathbf{a}_2, \mathbf{v}_3 - \mathbf{a}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$.

Par conséquent, l'extracteur de connaissance $\mathcal{K}$ obtient les vecteurs $\mathbf{y}_i = \mathbf{v}_i - \mathbf{a}_i$, avec $i \in [\![1,4]\!]$, tels que : $\sum_{i=1}^{4} \mathbf{H}_i\mathbf{y}_i^\top = \mathbf{c}^\top$ et $(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$. □

3.5.2 Preuve de validité d'un duo de chiffrés

Pour la construction du PAKE en section 3.6.2, nous aurons besoin de dériver à partir du protocole en figure 3.4 une preuve que deux chiffrés générés selon deux clés différentes du cryptosystème RQC.PKE chiffrent le même message μ. Soit μ un message, soit $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$ et $\mathbf{c}' = (\mathbf{c}_3, \mathbf{c}_4)$ deux chiffrés de μ selon les paires de clés notées $(\mathsf{pk}, \mathsf{sk})$ et $(\mathsf{pk}', \mathsf{sk}')$ où $\mathsf{sk} = (\mathbf{x}, \mathbf{y})$, $\mathsf{pk} = (\mathbf{g}, \mathbf{h}, \mathbf{s})$ avec $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$, et $\mathsf{sk}' = (\mathbf{x}', \mathbf{y}')$, $\mathsf{pk}' = (\mathbf{g}, \mathbf{h}', \mathbf{s}')$ avec $\mathbf{s}' = \mathbf{x}' + \mathbf{h}' \cdot \mathbf{y}'$. On peut remarquer qu'une preuve que deux chiffrés $\mathbf{c}$ et $\mathbf{c}'$ chiffrent un même message μ revient à effectuer une preuve de connaissance sur les vecteurs $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$, $\mathbf{r}' = (\mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6)$ et μ telle que la relation suivante soit satisfaite :

$$\mathcal{R} = \left\{ \begin{array}{l|l} \mathbf{c_1} = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \\ \mathbf{c_2} = \mu\mathrm{Gab}_{\mathbf{g}} + \mathbf{s}\mathbf{r}_2 + \mathbf{r}_3 & (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V}) \\ \\ \mathbf{c_3} = \mathbf{r}_4 + \mathbf{h}'\mathbf{r}_5 \\ \mathbf{c_4} = \mu\mathrm{Gab}_{\mathbf{g}} + \mathbf{s}'\mathbf{r}_5 + \mathbf{r}_6 & (\mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6) \in \mathfrak{S}^3_{w_r}(\mathcal{V}). \end{array} \right.$$

Soit $\tilde{\mathbf{h}}$ la matrice définie par :

$$\tilde{\mathbf{h}} = \begin{pmatrix} \mathbf{I}_n & \mathcal{I}(\mathbf{h}) & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathcal{I}(\mathbf{s}) & \mathbf{I}_n & 0 & 0 & 0 & \mathrm{Gab}_{\mathbf{g}} \\ 0 & 0 & 0 & \mathbf{I}_n & \mathcal{I}(\mathbf{h}') & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{I}(\mathbf{s}') & \mathbf{I}_n & \mathrm{Gab}_{\mathbf{g}} \end{pmatrix},$$

et soit $\mathbf{d} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4)$. Soit $\tilde{\mathbf{r}} = (\mathbf{r}_1, \mathbf{r}_2 \dots \mathbf{r}_6, \mu)$ un témoin de la relation $\mathcal{R}$ tel

que $(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$, $(\mathbf{r}_4, \mathbf{r}_5, \mathbf{r}_6) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$, et $\boldsymbol{\mu} \in \mathbb{F}_{q^m}^k$. On a alors $\tilde{\mathbf{h}} \cdot \tilde{\mathbf{r}}^\top = \mathbf{d}^\top$.

Par conséquent, $\tilde{\mathbf{r}}$ est un témoin pour le protocole décrit en figure 3.4 lorsqu'il est instancié en utilisant $(\tilde{\mathbf{h}}, \mathbf{d})$. Dans la figure 3.5, nous présentons la preuve interactive que deux chiffrés utilisant deux paires de clés différentes chiffrent un même message $\boldsymbol{\mu}$. Le protocole prend en entrée le couple $(\tilde{\mathbf{h}}, \mathbf{d})$ où $\tilde{\mathbf{h}} = (\|_{i=1}^7 \tilde{\mathbf{h}}_i)$.

Théorème 5. *Le protocole décrit en figure 3.5 est statistiquement à divulgation nulle de connaissance dans le modèle de l'oracle aléatoire.*

Démonstration. De façon similaire à la preuve de connaissance pour un chiffré de RQC, la preuve utilise les techniques décrites dans [Ste96, KTX08, LNSW13]. Nous allons construire un simulateur S qui, étant donné les données publiques du protocole en entrée et interagissant avec un vérifieur malhonnête $\widetilde{\mathsf{V}}$, retourne une retranscription simulée avec une probabilité de $\frac{2}{3}$ qui est statistiquement proche de la distribution obtenue avec des échanges réels. Les valeurs publiques en entrée sont $(\mathbf{H}, \mathbf{c})$, le simulateur S commence par choisir un challenge aléatoire $\bar{ch} \in \{0, 1, 2\}$, qui représente une prédiction du challenge que le vérifieur $\widetilde{\mathsf{V}}$ ne choisira pas.

$\diamondsuit$ **Premier cas** $\bar{ch} = 0$ **:**

S génère :

$$\begin{array}{lll}
\mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q)\,; & \mathbf{P}'_1, \mathbf{P}'_2, \ldots, \mathbf{P}'_6 \xleftarrow{\$} \mathsf{GL}_n(q)\,; & \mathbf{P}'_7 \xleftarrow{\$} \mathsf{GL}_k(q)\,; \\
\mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_6 \xleftarrow{\$} \mathcal{V}\,; & \mathbf{v}'_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,; & \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\mathfrak{K}}\,; \\
(\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})\,; & (\mathbf{r}'_4, \mathbf{r}'_5, \mathbf{r}'_6) \in \mathfrak{S}^3_{w_r}(\mathcal{V})\,; & \mathbf{r}'_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,;
\end{array}$$

et envoie le commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

- $cm'_1 = \mathrm{Hash}(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \sum_{i=1}^7 \mathbf{H}_i(\mathbf{v}'^\top_i + \mathbf{r}'^\top_i) \mid \mathbf{z}'_1)$
- $cm'_2 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{z}'_2)$
- $cm'_3 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3)$.

Une fois reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la façon suivante :

- Si $ch = 0$, retourner $\perp$ et interrompre.

1. [Commitment] P génère $\mathbf{Q} \xleftarrow{\$} \mathsf{GL}_m(q)$; $\{\mathbf{P}_i\}_{i\in[\![1,6]\!]} \xleftarrow{\$} \mathsf{GL}_n(q)$; $\mathbf{P}_7 \xleftarrow{\$} \mathsf{GL}_k(q)$; $\{\mathbf{v}_i\}_{i\in[\![1,6]\!]} \xleftarrow{\$} \mathcal{V}$; $\mathbf{v}_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k$; $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \xleftarrow{\$} 1^{\mathfrak{K}}$ et envoie le commitment CMT :=$(cm_1 \mid cm_2 \mid cm_3)$, où :
   - $\diamond$ $cm_1 = \text{Hash}(\mathbf{Q} \mid \|_{i=1}^7 \mathbf{P}_i \mid \sum_{i=1}^7 \tilde{\mathbf{H}}_i \mathbf{v}_i^\top \mid \mathbf{z}_1)$
   - $\diamond$ $cm_2 = \text{Hash}(\|_{i=1}^7 \mathbf{Q} * \mathbf{v}_i\mathbf{P}_i \mid \mathbf{z}_2)$
   - $\diamond$ $cm_3 = \text{Hash}(\|_{i=1}^6 \mathbf{Q} * (\mathbf{v}_i + \mathbf{r}_i)\mathbf{P}_i \mid \mathbf{Q} * (\mathbf{v}_7 + \boldsymbol{\mu})\mathbf{P}_7 \mid \mathbf{z}_3)$
2. [Challenge] V envoie un challenge aléatoire $ch \in \{0, 1, 2\}$ à P.
3. [Réponse] P répond en envoyant :
   - $\diamond$ Si $ch = 0$, soit $\mathbf{E} = \mathbf{Q}$, $\forall i \in [\![1, 7]\!]$, soit $\mathbf{a}_i = \mathbf{v}_i$, $\mathbf{F}_i = \mathbf{P}_i$.
     Retourner RSP :=$(\mathbf{E} \mid \|_{i=1}^7 \mathbf{F}_i \mid \|_{i=1}^7 \mathbf{a}_i \mid \mathbf{z}_1 \mid \mathbf{z}_2)$
   - $\diamond$ Si $ch = 1$, soit $\mathbf{I} = \mathbf{Q}$, $\forall i \in [\![1, 6]\!]$ soit $\mathbf{b}_i = \mathbf{v}_i + \mathbf{r}_i$, $\mathbf{b}_7 = \mathbf{v}_7 + \boldsymbol{\mu}$ et $\forall i \in [\![1, 7]\!]$ soit $\mathbf{J}_i = \mathbf{P}_i$.
     Retourner RSP :=$(\mathbf{I} \mid \|_{i=1}^7 \mathbf{J}_i \mid \|_{i=1}^7 \mathbf{b}_i \mid \mathbf{z}_1 \mid \mathbf{z}_3)$
   - $\diamond$ Si $ch = 2$, $\forall i \in [\![1, 7]\!]$ soit $\mathbf{c}_i = \mathbf{Q} * \mathbf{v}_i\mathbf{P}_i$, $\forall i \in [\![1, 6]\!]$ soit $\mathbf{d}_i = \mathbf{Q} * \mathbf{r}_i\mathbf{P}_i$ et $\mathbf{d}_7 = \mathbf{Q} * \boldsymbol{\mu}\mathbf{P}_7$.
     Retourner RSP :=$(\|_{i=1}^7 \mathbf{c}_i \mid \|_{i=1}^7 \mathbf{d}_i \mid \mathbf{z}_2 \mid \mathbf{z}_3)$
4. [Vérification], V effectue les vérifications suivantes :
   - $\diamond$ Si $ch = 0$, vérifier que :
     - $\star$ $cm_1 = \text{Hash}(\mathbf{E} \mid \|_{i=1}^7 \mathbf{F}_i \mid \sum_{i=1}^7 \tilde{\mathbf{H}}_i \mathbf{a}_i^\top \mid \mathbf{z}_1)$,
     - $\star$ $cm_2 = \text{Hash}(\|_{i=1}^7 \mathbf{E} * \mathbf{a}_i\mathbf{F}_i \mid \mathbf{z}_2)$.
   - $\diamond$ Si $ch = 1$, vérifier que :
     - $\star$ $cm_1 = \text{Hash}(\mathbf{I} \mid \|_{i=1}^7 \mathbf{J}_i \mid \sum_{i=1}^7 \tilde{\mathbf{H}}_i \mathbf{b}_i^\top - \mathbf{d} \mid \mathbf{z}_1)$,
     - $\star$ $cm_3 = \text{Hash}(\|_{i=1}^7 \mathbf{I} * \mathbf{b}_i\mathbf{J}_i \mid \mathbf{z}_3)$.
   - $\diamond$ Si $ch = 2$, vérifier que :
     - $\star$ $cm_2 = \text{Hash}(\|_{i=1}^7 \mathbf{c}_i \mid \mathbf{z}_2)$,
     - $\star$ $cm_3 = \text{Hash}(\|_{i=1}^7 \mathbf{c}_i + \mathbf{d}_i \mid \mathbf{z}_3)$,
     - $\star$ $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$, $(\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$.

V renvoie $d = 1$ (acceptation) si toutes les vérifications sont validées. Dans le cas contraire, il renvoie $d = 0$ (rejet).

FIGURE 3.5 – Preuve d'un duo de chiffrés de RQC d'un même message $\boldsymbol{\mu}$

- $\diamond$ Si $ch = 1$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}_i' \mid \|_{i=1}^7 \mathbf{v}_i' + \mathbf{r}_i' \mid \mathbf{z}_1' \mid \mathbf{z}_3')$
- $\diamond$ Si $ch = 2$, retourner RSP :=$(\|_{i=1}^7 \mathbf{Q}' * \mathbf{v}_i'\mathbf{P}_i' \mid \|_{i=1}^7 \mathbf{Q}' * \mathbf{r}_i'\mathbf{P}_i' \mid \mathbf{z}_2' \mid \mathbf{z}_3')$.

$\diamondsuit$ **Deuxième cas** $\bar{ch} = 1$

S génère :

$$\begin{array}{lll} \mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q)\,; & \mathbf{P}'_1, \mathbf{P}'_2, \ldots, \mathbf{P}'_6 \xleftarrow{\$} \mathsf{GL}_n(q)\,; & \mathbf{P}'_7 \xleftarrow{\$} \mathsf{GL}_k(q)\,; \\ \mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_6 \xleftarrow{\$} \mathcal{V}\,; & \mathbf{v}'_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,; & \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\aleph}\,; \\ (\mathbf{r}'_1, \mathbf{r}'_2, \mathbf{r}'_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})\,; & (\mathbf{r}'_4, \mathbf{r}'_5, \mathbf{r}'_6) \in \mathfrak{S}^3_{w_r}(\mathcal{V})\,; & \mathbf{r}'_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,; \end{array}$$

et envoyer le commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

- $cm'_1 = \mathrm{Hash}(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \sum_{i=1}^7 \mathbf{H}_i \mathbf{v}_i'^{\top} \mid \mathbf{z}'_1)$
- $cm'_2 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{z}'_2)$
- $cm'_3 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3)$.

Après avoir reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la façon suivante :

- Si $ch = 0$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \|_{i=1}^7 \mathbf{v}'_i \mid \mathbf{z}'_1 \mid \mathbf{z}'_2)$
- Si $ch = 1$, retourner $\perp$ et interrompre.
- Si $ch = 2$, retourner RSP :=$(\|_{i=1}^7 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \|_{i=1}^7 \mathbf{Q}' * \mathbf{r}'_i \mathbf{P}'_i \mid \mathbf{z}'_2 \mid \mathbf{z}'_3)$.

$\diamondsuit$ **Troisième cas** $\bar{ch} = 2$

En utilisant l'algèbre linéaire, S calcule le vecteur $\mathbf{r}' = (\mathbf{r}'_1, \mathbf{r}'_2 \ldots \mathbf{r}'_7)$, tel que $\mathbf{H}\mathbf{r}'^{\top} = \mathbf{c}^{\top}$. Par la suite, il génère :

$$\begin{array}{lll} \mathbf{Q}' \xleftarrow{\$} \mathsf{GL}_m(q)\,; & \mathbf{P}'_1, \mathbf{P}'_2, \ldots, \mathbf{P}'_6 \xleftarrow{\$} \mathsf{GL}_n(q)\,; & \mathbf{P}'_7 \xleftarrow{\$} \mathsf{GL}_k(q)\,; \\ \mathbf{v}'_1, \mathbf{v}'_2, \ldots, \mathbf{v}'_6 \xleftarrow{\$} \mathcal{V}\,; & \mathbf{v}'_7 \xleftarrow{\$} \mathbb{F}_{q^m}^k\,; & \mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3 \xleftarrow{\$} 1^{\aleph}\,; \end{array}$$

et retourne commitment CMT :=$(cm'_1 \mid cm'_2 \mid cm'_3)$ à $\widetilde{\mathsf{V}}$, où :

$$cm'_1 = \mathrm{Hash}(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \sum_{i=1}^7 \mathbf{H}_i \mathbf{v}_i'^{\top} \mid \mathbf{z}'_1)$$

$$cm'_2 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * \mathbf{v}'_i \mathbf{P}'_i \mid \mathbf{P}'_7 \mid \mathbf{z}'_2)$$

$$cm'_3 = \mathrm{Hash}(\|_{i=1}^7 \mathbf{Q}' * (\mathbf{v}'_i + \mathbf{r}'_i)\mathbf{P}'_i \mid \mathbf{z}'_3).$$

Après avoir reçu le challenge ch par $\widetilde{\mathsf{V}}$, le simulateur S répond de la façon suivante :

- Si $ch = 0$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \|_{i=1}^7 \mathbf{v}'_i \mid \mathbf{z}'_1 \mid \mathbf{z}'_2)$
- Si $ch = 1$, retourner RSP :=$(\mathbf{Q}' \mid \|_{i=1}^7 \mathbf{P}'_i \mid \|_{i=1}^7 \mathbf{v}'_i + \mathbf{r}'_i \mid \mathbf{z}_1 \mid \mathbf{z}_3)$

$\diamond$ Si $ch = 2,$ retourner $\perp$ et interrompre.

On peut remarquer que la probabilité que le simulateur renvoie $\perp$ est égale à $\frac{1}{3}$. De plus, lorsque le simulateur n'est pas interrompu, la distribution générée est statistiquement proche de celle obtenue lors des échanges réels lorsque la fonction de hachage est modélisée à l'aide d'un oracle aléatoire. Par conséquent, nous avons construit un simulateur qui parvient à simuler en temps polynomial le protocole avec une probabilité de $\frac{2}{3}$ sans avoir la connaissance des valeurs secrètes du protocole en question. □

Théorème 6. *S'il existe un prouveur malhonnête disposant d'un algorithme probabiliste en temps polynomial $\widetilde{\mathsf{P}}$ capable de convaincre un vérifieur avec une probabilité de $\frac{2}{3} + \varepsilon$, où ε est non négligeable, alors il existe un extracteur de connaissance probabiliste en temps polynomial qui renvoie, avec une écrasante probabilité, $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_7)$ tel que $\sum_{i=1}^{7} \tilde{\mathbf{h}}_i \mathbf{y}_i^\top = \mathbf{c}^\top$, $(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$ et $(\mathbf{y}_4, \mathbf{y}_5, \mathbf{y}_6) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$.*

Démonstration. Nous allons construire un extracteur de connaissance $\mathcal{K}$. Soit $\widetilde{\mathsf{P}}$ un prouveur malhonnête capable de convaincre un vérifieur avec une probabilité de $\frac{2}{3} + \varepsilon$. En appliquant la technique de Véron [Vér97], consistant à rembobiner $\widetilde{\mathsf{P}}$ un nombre de fois polynomial en $\frac{1}{\varepsilon}$, l'extracteur de connaissance est alors capable d'obtenir, avec une probabilité écrasante un commitment, avec lequel $\widetilde{\mathsf{P}}$ peut valider l'ensemble des trois challenges. Par conséquent, $\mathcal{K}$ obtient les équations suivantes :

$\diamond$ $cm_1 = \text{Hash}(\mathbf{E} \mid \|_{i=1}^{7} \mathbf{F}_i \mid \sum_{i=1}^{7} \tilde{\mathbf{H}}_i \mathbf{a}_i^\top) = \text{Hash}(\mathbf{I} \mid \|_{i=1}^{7} \mathbf{J}_i \mid \sum_{i=1}^{7} \tilde{\mathbf{H}}_i \mathbf{b}_i^\top - \mathbf{c}^\top)$

$\diamond$ $cm_2 = \text{Hash}(\|_{i=1}^{7} \mathbf{E} * \mathbf{a}_i \mathbf{F}_i) = \text{Hash}(\|_{i=1}^{7} \mathbf{c}_i)$

$\diamond$ $cm_3 = \text{Hash}(\|_{i=1}^{7} \mathbf{I} * \mathbf{b}_i \mathbf{J}_i) = \text{Hash}(\|_{i=1}^{7} \mathbf{c}_i + \mathbf{d}_i)$

$\diamond$ $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$, $(\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6) \in \mathfrak{S}_{w_r}^3(\mathcal{V})$.

Comme la fonction Hash est modélisée comme un oracle aléatoire (un adversaire est incapable de trouver une collision via cet oracle), il s'ensuit que :

$\diamond$ $\mathbf{E} = \mathbf{I}$ et $\forall i \in [\![1, 7]\!],\ \mathbf{F}_i = \mathbf{J}_i$ et $\sum_{i=1}^{7} \tilde{\mathbf{H}}_i \mathbf{a}_i^\top = \sum_{i=1}^{4} \tilde{\mathbf{H}}_i \mathbf{b}_i^\top - \mathbf{c}^\top$

$\diamond$ $\forall i \in [\![1, 7]\!],\ \mathbf{E} * \mathbf{a}_i \mathbf{F}_i = \mathbf{c}_i,\ \mathbf{I} * \mathbf{b}_i \mathbf{J}_i = \mathbf{c}_i + \mathbf{d}_i.$

Soit $i \in [\![1, 7]\!]$. On a $\mathbf{I} * \mathbf{b}_i \mathbf{J}_i = \mathbf{E} * \mathbf{b}_i \mathbf{F}_i = \mathbf{c}_i + \mathbf{d}_i$. Il suit que $\mathbf{E} * (\mathbf{b}_i - \mathbf{a}_i) \mathbf{F}_i = \mathbf{d}_i$, ce qui implique que $(\mathbf{b}_i - \mathbf{a}_i) = \mathbf{E}^{-1} * \mathbf{d}_i \mathbf{F}_i^{-1}$.

$$\forall i \in [\![1,7]\!],\ (\mathbf{b}_i - \mathbf{a}_i) = \mathbf{E}^{-1} * \mathbf{d}_i \mathbf{F}_i^{-1}.$$

Comme $\mathbf{E}^{-1} \in \mathsf{GL}_m(q)$ et $\mathbf{F}_i^{-1} \in \mathsf{GL}_m(q)$, on a $(\mathbf{b}_1 - \mathbf{a}_1, \mathbf{b}_2 - \mathbf{a}_2, \mathbf{b}_3 - \mathbf{a}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$ et $(\mathbf{b}_4 - \mathbf{a}_4, \mathbf{b}_5 - \mathbf{a}_5, \mathbf{b}_6 - \mathbf{a}_5) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$.

Par conséquent, l'extracteur de connaissance $\mathcal{K}$ obtient le vecteur $\mathbf{y}_i = \mathbf{b}_i - \mathbf{a}_i$, avec $i \in [\![1,7]\!]$, tel que : $\sum_{i=1}^{7} \tilde{\mathbf{H}}_i \mathbf{y}_i^\top = \mathbf{c}^\top$, $(\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$ et $(\mathbf{y}_4, \mathbf{y}_5, \mathbf{y}_6) \in \mathfrak{S}^3_{w_r}(\mathcal{V})$. □

Protocole non-interactif. Pour rendre le protocole non-interactif, l'on applique la transformation de Fiat-Shamir [FS86, PS96] dans le modèle de l'oracle aléatoire. Considérons une fonction de hachage $\mathcal{H} : \{0,1\}^* \to \{0,1,2\}^\kappa$ (où $\kappa = \|\log \mathfrak{K}\|$) modélisée comme un oracle aléatoire. On obtient alors la preuve non-interactive suivante $\Pi = (\|_{i=1}^{\kappa} \mathrm{CMT}^{(i)} \mid \|_{i=1}^{\kappa} ch^{(i)} \mid \|_{i=1}^{\kappa} \mathrm{RSP}^{(i)})$, où $\|_{i=1}^{\kappa} ch^{(i)} = \mathcal{H}(M \mid \|_{i=1}^{\kappa} \mathrm{CMT}^{(i)})$ et M est un message aléatoire.

3.6 Deux protocoles basés sur ce HPS

Les Hash Proof Systems peuvent être utilisés comme une primitive pour construire divers protocoles cryptographiques. Nous allons donner dans cette section deux applications possibles, un schéma de chiffrement à témoin (Witness Encryption scheme - WE) et un schéma d'authentification basé sur les mots de passe (Password Authenticated Key Exchange - PAKE) basé sur le HPS décrit en figure 3.3. Les constructions que nous présentons peuvent être adaptées au modèle UC par le fait que la zone problématique décrite au chapitre 3.7 est complètement nullifiée par la preuve de connaissance explicitée précédemment, permettant de réduire l'espace ambiant aux seuls chiffrés de RQC. Cela nécessiterait l'utilisation d'un schéma de commitment extractible/équivoque ce qui peut être achevé en utilisant un commitment basé sur les codes à la Haralambiev [BC15], que nous ne décrirons pas ici.

3.6.1 Un schéma de chiffrement à témoin

Le concept de chiffrement à témoin a été introduit par Garg et al. [GGSW13]. Soit $\mathcal{L}$ un langage $\mathcal{NP}$ inclu dans un espace ambiant $\mathcal{X}$, W un mot du langage et w une valeur témoin permettant de vérifier l'appartenance de W au langage en un temps polynomial. Soit x un élément de l'espace ambiant. Un chiffrement

à témoin permet de chiffrer un message μ à partir de x. Le récepteur du message chiffré peut alors déchiffrer ce message si x est un mot du langage et s'il connait un témoin w de l'appartenance de x au langage. Dans un tel schéma, il est d'ailleurs possible que la personne chiffrant le message ne sache pas si x appartient au langage.

Définition 52 (Chiffrement à témoin)**.** *Un chiffrement à témoin pour un langage $\mathcal{NP}$ noté $\mathcal{L}$ est composé de deux algorithmes :*

- ◇ *Chiffrement$_{WE}(\mathfrak{K}, W, \mu)$: Etant donné un paramètre de sécurité $\mathfrak{K}$, un message μ et un élément $W \in \mathcal{X}$, renvoyer un chiffré $\mathbf{c}$.*
- ◇ *Déchiffrement$_{WE}(\mathbf{c}, w)$: A partir du chiffré $\mathbf{c}$ et du témoin w de l'appartenance de W au langage, renvoyer le message initial μ ou $\perp$.*

Une stratégie commune [ABP15, BBDQ18] pour construire un schéma de chiffrement à témoin consiste à utiliser un HPS exact de la façon suivante : étant donné un mot du langage W et un message μ, l'envoyeur génère la clé de hachage hk, la clé de projection hp, le haché $\mathcal{H}_{\mathsf{hk}}$ et masque le message μ en utilisant le haché $\mathcal{H}_{\mathsf{hk}}$. Pour le déchiffrement, le récepteur exploite son témoin w associé au mot W ainsi que la clé projetée hp afin de calculer le haché projeté $\mathcal{H}_{\mathsf{hp}}$ et de retrouver la valeur du message μ. La même technique peut être utilisée avec un HPS approximatif, par l'ajout d'un code correcteur d'erreurs permettant de se débarrasser du différentiel entre les deux hachés, comme décrit sur la figure 3.6.

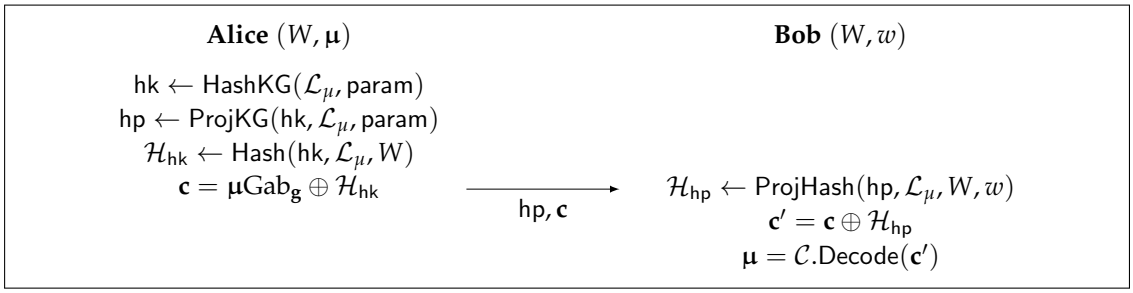

FIGURE 3.6 – Un chiffrement à témoin basé sur notre HPS

3.6.2 Un schéma d'échange de clés authentifié par mot de passe

Les schémas d'authentification basés sur des mots de passe (*Password Authenticated Key Exchange* - PAKE) ont été introduits pour la première fois par Bellovin et Merritt [BM92]. L'objectif d'un tel protocole est de permettre à un utilisateur de générer une clé cryptographique de forte entropie à partir d'un mot de

passe partagé pw qui soit mémorisable par l'être humain, et donc de relativement faible entropie, et ce sans recourir à l'utilisation d'une infrastructure à clé publique.

Dans un tel protocole, un adversaire contrôlant l'ensemble des communications du réseau ne doit pas pouvoir monter une attaque offline par dictionnaire pour récupérer une quelconque information sur le mot de passe utilisé. Il s'avère que les Hash Proofs Systems sont des primitives assez intéressantes pour l'élaboration de tels schémas. Plusieurs publications scientifiques basées sur les réseaux euclidiens [KV09, ZY17, BBDQ18] permettent la construction de PAKE post-quantiques, cependant nous sommes les premiers, à notre connaissance, à proposer un PAKE post-quantique qui soit basé sur les codes correcteurs d'erreurs.

[BBC+13] ont proposé en 2013 une construction permettant de construire un PAKE sécurisé en une seule passe dans le modèle BPR [BPR00]. Leur construction requiert que chaque utilisateur envoie un chiffré CCA-2 de leur mot de passe ainsi que la clé projetée d'un KV-HPS et dont le langage est composé de l'ensemble des chiffrés valides du mot de passe en question.

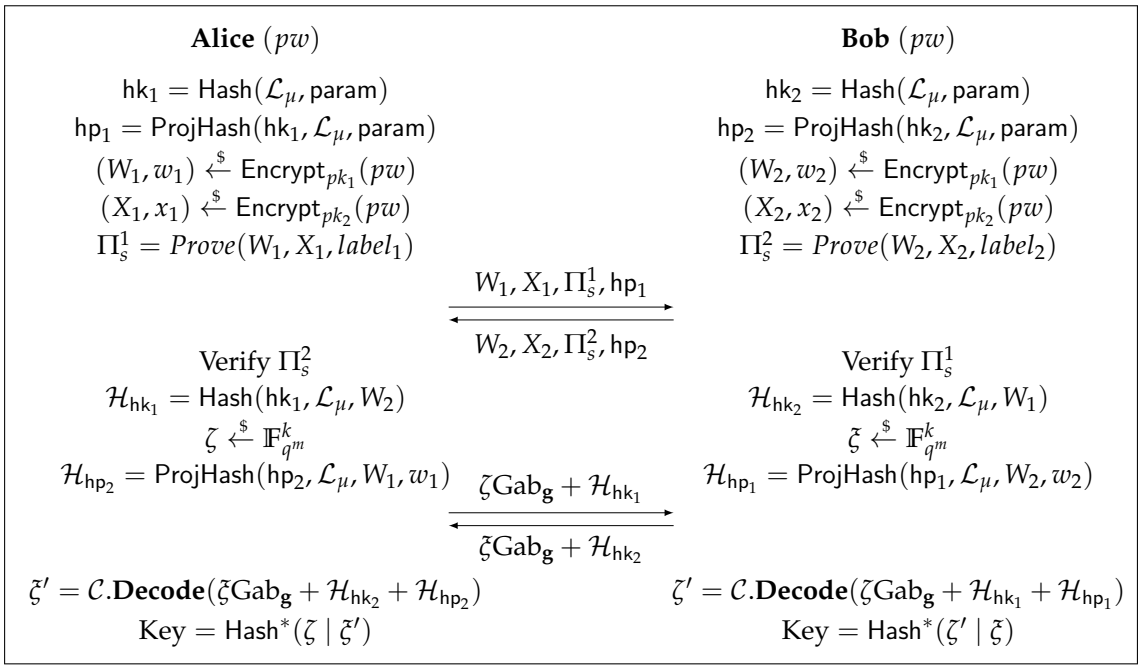

FIGURE 3.7 – Schéma d'échange de clés authentifié par mot de passe en deux passes

Nous avons d'ores et déjà montré en section 3.4 comment nous pouvons obtenir un tel HPS à partir du chiffrement à clé publique RQC. En utilisant la

transformation de Naor Yung [NY90], il est possible d'obtenir un chiffrement CCA-2 de pw à partir de deux chiffrements CPA (Chosen Plaintext Attacks secure) de pw sous deux clés différentes et d'une preuve à divulgation nulle de connaissance permettant de démontrer que les chiffrés ont la forme voulue et se déchiffrent en le même message. Nous avons également vu comment il est possible de construire une telle preuve de connaissance par l'utilisation d'un protocole à la Stern adapté pour la métrique rang en section 3.5.1.

Nous pouvons donc, en exploitant les primitives citées précédemment, directement obtenir le PAKE décrit en figure 3.7. La première passe correspond au PAKE à proprement parlé, la deuxième passe étant une phase de réconciliation nécessaire afin de supprimer le différentiel dû au fait que le HPS ne soit pas exact. Ce protocole peut également basculer sous une forme asynchrone à trois passes en regroupant les deux passes envoyées par Bob.

3.7 Paramètres et performances

Dans cette section, nous allons donner des paramètres pour les protocoles présentés à la section précédente. Ces paramètres sont valides pour les deux protocoles. La sécurité de ces constructions dépend des instances suivantes :

- ◇ Sécurité de la clé de hachage hk : pour récupérer la valeur de hk à partir de hp, il faut considérer l'instance du problème 2-IRSD suivant :

$$\begin{pmatrix} \mathcal{I}(\mathbf{s}) & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{pmatrix} = \mathsf{hp}.$$

- ◇ Sécurité de la valeur témoin : pour récupérer la valeur des $(\mathbf{r_i})_{i\in[\![1,3]\!]}$, il faut considérer l'instance du problème 3-IRSD suivante :

$$\begin{pmatrix} \mathbf{I}_n & 0 & \mathcal{I}(\mathbf{h}) \\ 0 & \mathbf{I}_n & \mathcal{I}(\mathbf{s}) \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_3 \\ \mathbf{r}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 - \boldsymbol{\mu}\mathrm{Gab}_{\mathbf{g}} \end{pmatrix}.$$

- ◇ Sécurité liée à la smoothness : la preuve de KV-smoothness implique deux problèmes différents :
  - ⋆ Le premier est une instance du problème 2-IRSD :

$$\begin{pmatrix} \mathbf{I}_n & \mathcal{I}(\mathbf{h}) \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \mathbf{s}.$$

  - ⋆ Le deuxième est une instance du problème FIRSD :

$$\begin{pmatrix} \mathbf{s} & \mathbf{I}_n & 0 \\ \mathbf{c}_2 - \mu \mathrm{Gab}_{\mathbf{g}} & 0 & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 \end{pmatrix} = \begin{pmatrix} \mathsf{hp} \\ \mathcal{H}_{\mathsf{hk}} \end{pmatrix}.$$

Décodage des codes de Gabidulin : De façon à pouvoir décoder les codes de Gabidulin, nous avons besoin d'avoir $m \geqslant n$ et $\|\mathbf{x} \cdot \mathbf{r}_2 - \mathbf{y} \cdot \mathbf{r}_1 + \mathbf{r}_3\| \leq \left\lfloor \frac{n-k}{2} \right\rfloor$, ce qui est équivalent à $(\omega_x + 1) \cdot \omega_r \leq \left\lfloor \frac{n-k}{2} \right\rfloor$.

Les deux approches principales pour résoudre le problème du décodage par syndrome en métrique rang sont combinatoires et algébriques. Alors que les attaques combinatoires sont longtemps restées la meilleure approche pour attaquer ce problème [GRS15, AGHT18], de récentes approches algébriques ont changé la donne [BBC$^+$20a, BBC$^+$20b]. Nous allons nous baser sur [BBC$^+$20b] pour le calcul des paramètres de sécurité. Nous allons nous restreindre aux complexités pour les codes $[3n, n]$ car les complexités correspondantes pour les codes $[2n, n]$ sont plus élevées.

- ◇ hyb3n(a) : correspond aux attaques de type hybride pour les codes de longueur $3n$, a étant défini comme le plus petit entier positif pour atteindre le cas surdéterminé, $a = 0$ signifiant que les paramètres ont déjà atteint le cas surdéterminé.
- ◇ hyb2n(a) : correspond aux attaques hybrides pour des supports homogènes pour un code de longueur $2n$, a étant défini de la même façon que précédemment.
- ◇ und2n : correspond aux attaques pour des supports homogènes dans le cas sous-déterminé pour un code de longueur $2n$, b étant défini comme le plus petit entier positif tel que le nombre d'inconnues est inférieur au nombre d'équations.
- ◇ und3n : correspond aux attaques dans le cas sous-déterminé pour un code de longueur $3n$. b étant défini de la même façon que précédemment.
- ◇ comb3n : correspond à la complexité des attaques combinatoires pour un code de longueur $3n$.

La table 3.3 montre que la longueur des vecteurs considérés n'est pas excessive. Pour le protocole PAKE en revanche, il faut considérer des longueurs approximativement cent fois plus élevées pour l'utilisation de la preuve de connais-

Instance	q	n	m	k	w_r	w_x	w_α	Vecteurs (bits)	Sécurité (bits)
I	2	137	139	4	7	8	11	19 043	128
II	2	211	223	4	10	9	12	47 053	192
III	2	283	293	3	13	10	15	82 919	256

TABLE 3.3 – Paramètres et niveaux de sécurité associés, cas homogène.

Instance	hybr3n(a)	und3n(b)	comb3n	Sécurité
I	138(0)	162(1)	220	128
II	201(0)	229(1)	560	192
III	264(0)	293(1)	1 028	256

TABLE 3.4 – Niveaux de sécurité et complexités associées, cas homogène.

sance à la Stern (section 3.5.1) suivie de la transformation de Fiat Shamir.

Il est possible de considérablement décroître ces paramètres en considérant le cas non-homogène du problème du décodage par syndrome en métrique rang (voir la dernière spécification de RQC d'avril 2020), *i.e.* le cas où les vecteurs recherchés ne sont plus nécessairement de même support. On aurait $\mathsf{Supp}(r_1) = \mathsf{Supp}(r_2) \subset \mathsf{Supp}(r_3)$ avec $\|r_1\| = \|r_2\| = w_1$ et $\|r_3\| = w_1 + w_2$. De façon similaire, on aurait $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3$ tels que $\mathsf{Supp}(\boldsymbol{\alpha}_1) = \mathsf{Supp}(\boldsymbol{\alpha}_2) \subset \mathsf{Supp}(\boldsymbol{\alpha}_3)$, $\|\boldsymbol{\alpha}_1\| = \|\boldsymbol{\alpha}_2\| = w_1$ et $\|\boldsymbol{\alpha}_3\| = w_1 + w_2$.

Les tableaux 3.3 et 3.4 montrent les complexités dans le cas homogène pour permettre davantage de clarté et de simplicité. Les tableaux 3.5 et 3.6 montrent des jeux de paramètres dans le cas non-homogène. Pour le détail des calculs dans le cas non-homogène, le lecteur est invité à se référer à [AAB+20].

Instance	q	n	m	k	w_1	w_2	w_x	Vecteurs (bits)	Sécurité
I	2	127	131	3	7	6	7	16 637	128
II	2	163	167	5	8	8	8	27 221	192
III	2	181	191	3	9	7	9	34 571	256

TABLE 3.5 – Paramètres et niveaux de sécurité associés, cas non-homogène.

Instance	hybr3n(a)	und3n(b)	hybr2n(a)	und2n(b)	Sécurité
I	216(0)	240(1)	156(5)	151(1)	128
II	268(0)	293(1)	325(23)	228(3)	192
III	295(2)	304(1)	545(43)	300(5)	256

TABLE 3.6 – Niveaux de sécurité et complexités associées, cas non-homogène.

Chapitre 4

Le problème LRE

Le problème Learning Parity with Noise (LPN) est un problème connu en cryptographie, et notamment en cryptographie à bas coût. Ce problème est intéressant à la fois d'un point de vue théorique et d'un point de vue pratique. D'un point de vue théorique, ce problème est équivalent au problème du décodage d'un code aléatoire, ce dernier ayant été prouvé $\mathcal{NP}$-complet [BMVT78]. D'un point de vue pratique, les protocoles basés sur ce problème sont souvent simples et peu coûteux en terme de taille des messages échangés. Ces protocoles sont souvent utilisés pour le fonctionnement d'objets aux capacités très restreintes comme les tags RFID.

La section 4.1 définira le problème Learning Parity with Noise. La section 4.2 donnera une description du protocole HB, premier protocole d'authentification à tirer parti du problème LPN. La section 4.3 décrira le problème LRE, adaptation du problème LPN à la métrique rang. Enfin, la section 4.4 présentera le protocole $\mathrm{HB}_{\mathsf{LRE}}$, adaptation du protocole d'authentification HB au problème LRE.

4.1 Le problème LPN

Soit k un entier naturel et considérons l'espace vectoriel ambiant $\mathcal{V} = \mathbb{F}_2^k$. On munit $\mathcal{V}$ d'un produit à valeurs dans $\mathbb{F}_2$ de la manière suivante, si $\mathbf{a} = (a_1, \ldots, a_k) \in \mathcal{V}$ et $\mathbf{b} = (b_1, \ldots, b_k) \in \mathcal{V}$, $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + \cdots + a_k b_k \in \mathbb{F}_2$.

Définition 53 (Distribution LPN)**.** *Etant donné une valeur secrète* $\mathbf{s} \in \mathcal{V}$*, un entier naturel* n *correspondant à un nombre d'échanges, un réel* $p \in]0, \frac{1}{2}[$ *représentant une*

probabilité, des éléments $v_1, \ldots, v_n \in \mathbb{F}_2$ *tels que, pour tout* $i \in [\![1, n]\!]$, $\mathbb{P}[v_i = 1] = p$, *et des vecteurs aléatoires* $\mathbf{a}_1, \ldots, \mathbf{a}_n \xleftarrow{\$} \mathcal{V}$, *la distribution* $\mathsf{LPN}(\mathbf{s}, n, p)$ *est la distribution obtenue en renvoyant les doublets* $(\mathbf{a}_i, (\mathbf{s} \cdot \mathbf{a}_i) \oplus v_i)$, *pour* $i \in [\![1, n]\!]$.

Définition 54 (Problème LPN)**.** *Etant donné des doublets* $(\mathbf{a}_i, (\mathbf{s} \cdot \mathbf{a}_i) \oplus v_i)$, *pour* $i \in [\![1, n]\!]$ *issus de la distribution* LPN, *le problème* LPN *consiste à retrouver la valeur de* $\mathbf{s}$.

Toute la difficulté du problème relève du fait qu'une partie des doublets ne correspondent pas aux produits $\mathbf{s} \cdot \mathbf{a}_i$ et sont erronés avec une probabilité p, rendant ainsi impossible la méthode de résolution directe du problème consistant à trouver la solution par élimination de Gauss.

4.2 Le protocole d'authentification HB

Un premier protocole d'authentification, simple et exploitant ce problème de façon assez naturelle, fut proposé par Hopper et Blum en 2001 [HB01]. Supposons que P (le prouveur) veuille prouver sa connaissance de la valeur secrète $\mathbf{s}$ à V (le vérifieur), qui possède lui aussi la connaissance de $\mathbf{s}$. Soit $t \in \mathbb{N}$ une valeur seuil d'acceptation, telle que $pn \leqslant t$. Le protocole fonctionne alors tel que présenté dans la figure 4.1. Nous notons $\mathcal{B}(1, p)$ la distribution de Bernoulli valant 1 avec probabilité p.

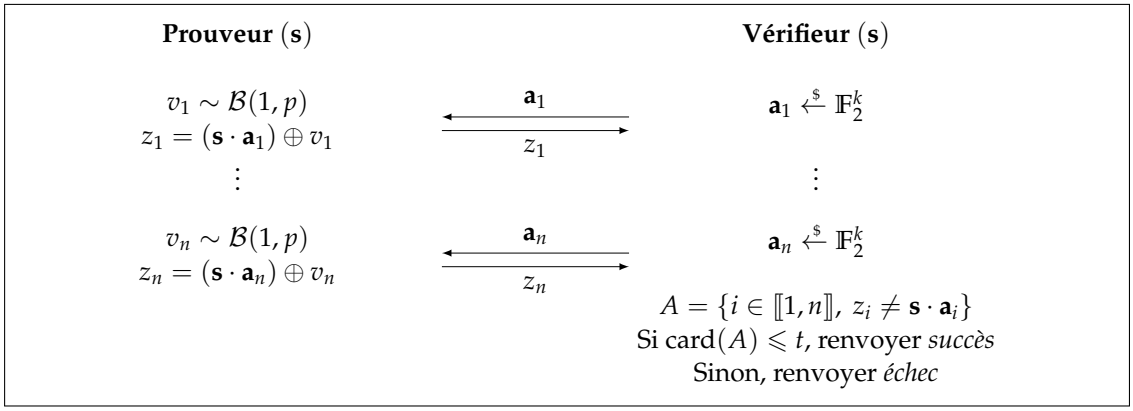

FIGURE 4.1 – Protocole HB

4.3 Le problème LRE

4.3.1 Introduction du problème

Le problème Learning with Rank Errors (LRE) est une adaptation du problème LPN à la métrique rang.

Définition 55 (Distribution LRE)**.** *Etant donné une distribution de l'erreur $\mathcal{X}$ sur $\mathbb{F}_{q^m}^l$. Soit $\mathbf{s} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ un vecteur secret. La distribution* LRE *est la distribution obtenue en renvoyant des doublets de la forme $(\mathbf{A}, \mathbf{sA} + \mathbf{e})$ où $\mathbf{A} \xleftarrow{\$} \mathbb{F}_{q^m}^{n \times l}$ et $\mathbf{e} \sim \mathcal{X}$.*

Définition 56 (Problème LRE)**.** *Etant donné des doublets de la forme $(\mathbf{A}, \mathbf{sA} + \mathbf{e})$ issus de la distribution* LRE*, le problème* LRE *consiste à retrouver la valeur secrète $\mathbf{s}$.*

Notation 5. *Si $\mathbf{e}$ est choisi aléatoirement selon la distribution uniforme sur l'ensemble des vecteurs de rang au plus r, alors on notera le problème* $\mathsf{LRE}(q, m, n, l, r)$*, ou simplement* $\mathsf{LRE}(m, n, l, r)$ *si $q = 2$. Si $\mathbf{e}$ est choisi selon une distribution $\mathcal{X}$, on notera le problème* $\mathsf{LRE}(q, m, n, l, \mathcal{X})$*.*

Définition 57 (Version décisionnelle du problème LRE)**.** *Etant donné des doublets de la forme $(\mathbf{A}, \mathbf{u}) \in \mathbb{F}_{q^m}^{n \times l} \times \mathbb{F}_{q^m}^n$ suivant la distribution* LRE *ou la distribution uniforme, la version décisionnelle du problème* LRE*, consiste à pouvoir déterminer, avec un avantage non négligeable, si ces éléments ont été générés à partir de la distribution* LRE *ou de la distribution uniforme sur $\mathbb{F}_{q^m}^{n \times l} \times \mathbb{F}_{q^m}^n$.*

4.3.2 Propriétés

Proposition 23 (Unicité)**.** *Le problème* $\mathsf{LRE}(m, n, l, r)$ *a, avec une forte probabilité, une unique solution dès lors que le nombre de doublets disponible est plus grand que $\frac{mn}{(m-r)(l-r)}$.*

Démonstration. La preuve est proche de celle pour le calcul de la borne de Gilbert-Varshamov en métrique rang, la forme de l'erreur est seulement adaptée pour correspondre au problème LRE.

Soit $\mathbf{A} = (\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_N)$ et $\mathbf{e} = (\mathbf{e}_1 \mid \cdots \mid \mathbf{e}_N)$ la concaténation des matrices et des erreurs obtenues à partir de N doublets. $\mathbf{A}$ est alors une matrice de taille $n \times Nl$ et $\mathbf{e}$ un vecteur de taille Nl. Si $N > \frac{mn}{(m-r)(l-r)}$, alors $Nl > n$. $\mathbf{A}$ est alors une matrice génératrice d'un code $[Nl, n]$ sur $\mathbb{F}_{q^m}$. Soit $\mathbf{H}$ une de ses matrices de parité, *i.e.* une matrice $(Nl - n) \times n$ de rang plein sur $\mathbb{F}_{q^m}$ telle que $\mathbf{AH}^\top = 0$.

On a alors :

$$\mathbf{e}\mathbf{H}^\top = (\mathbf{y}_1 \mid \cdots \mid \mathbf{y}_N) \cdot \mathbf{H}^\top \in \mathbb{F}_{q^m}^{Nl-n}$$

avec, pour tout $i \in [\![1, N]\!]$, $\mathbf{y}_i = \mathbf{s} \cdot \mathbf{A}_i + \mathbf{e}$.

Le vecteur $\mathbf{e}$ est l'unique solution de la forme $(\mathbf{e}_1 \mid \cdots \mid \mathbf{e}_N)$ de cette équation avec une grande probabilité si le nombre d'erreurs de cette forme est inférieur à $q^{m(Nl-n)}$. Ce nombre est égal à $\prod_{i=0}^{r-1} \frac{(q^m-q^i)(q^l-q^i)}{q^r-q^i} = \mathcal{O}(q^{Nr(m+l-r)})$. On a alors les équivalences suivantes :

$$\begin{aligned} q^{Nr(m+l-r)} &< q^{m(Nl-n)} \\ Nr(m+l-r) &< m(Nl-n) \\ mn &< N(ml - rm - r(l-r)) \\ \frac{mn}{(m-r)(l-r)} &< N. \end{aligned}$$

□

Proposition 24 (Réduction pire cas/cas moyen)**.** *Supposons qu'il existe un algorithme probabiliste en temps polynomial $\mathcal{A}$ résolvant le problème $\mathsf{LRE}(q, m, n, l, \mathcal{X})$ avec une probabilité non négligeable. Considérons une instance spécifique de $\mathsf{LRE}(q, m, n, l, \mathcal{X})$ d'inconnue $\mathbf{s}$. Alors on peut montrer qu'il existe un algorithme $\mathcal{A}'$ qui retrouve $\mathbf{s} \in \mathbb{F}_{q^m}^n$ avec une probabilité non négligeable à partir de doublets de la forme $(\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{e})$, $\mathbf{A} \xleftarrow{\$} \mathbb{F}_{q^m}^{n \times l}$, $\mathbf{e} \sim \mathcal{X}$.*

Démonstration. Soit $\mathcal{A}$ l'algorithme probabiliste permettant de résoudre en temps polynomial le problème $\mathsf{LRE}(q, m, n, l, \mathcal{X})$ avec une probabilité non négligeable p. Considérons une instance spécifique du problème $\mathsf{LRE}(q, m, n, l, \mathcal{X})$ d'inconnue $\mathbf{s}$. Soit $\mathbf{s}' \xleftarrow{\$} \mathbb{F}_{q^m}^n$. Posons $\mathbf{s}'' = \mathbf{s} + \mathbf{s}'$ et $\mathbf{y} = \mathbf{s}\mathbf{A} + \mathbf{e} + \mathbf{s}'\mathbf{A} = \mathbf{s}''\mathbf{A} + \mathbf{e}$. $\mathbf{s}'$ étant distribué aléatoirement selon la loi uniforme, $\mathbf{s}''$ l'est aussi. A partir du doublet $(\mathbf{A}, \mathbf{y})$, l'algorithme $\mathcal{A}$ renvoie $\mathbf{s}''$ avec une probabilité non négligeable p. Si c'est le cas, alors en calculant $\mathbf{s}'' - \mathbf{s}'$, on retrouve la valeur $\mathbf{s}$ de notre instance spécifique. □

Conjecture 1. *La distribution $\mathsf{LRE}(q, m, n, l, r)$ est pseudoaléatoire, i.e. il n'existe aucun distingueur probabibiliste capable de distinguer, en temps polynomial, la distribution LRE de la distribution uniforme sur $\mathbb{F}_{q^m}^{n \times l} \times \mathbb{F}_{q^m}^l$.*

Discussion autour de cette conjecture : bien qu'il n'existe pas de réduction, la

version décisionnelle du problème LRE est liée à la version décisionnelle du problème LPN. Il est en effet possible de transformer des doublets du problème LPN en des doublets très proches de la distribution LRE.

On utilise une transformation inspirée de [GZ16] pour intégrer ces distributions à la métrique rang. Soit $m > 8l$ et $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_l$ des éléments $\mathbb{F}_2$-indépendants de $\mathbb{F}_{2^m}$. Soit :

$$\begin{array}{rccc} \phi : & \mathbb{F}_2^l & \to & \mathbb{F}_{2^m}^l \\ & (x_1, \ldots, x_l) & \mapsto & (\boldsymbol{\alpha}_1 x_1, \ldots, \boldsymbol{\alpha}_l x_l). \end{array}$$

ϕ transforme un vecteur de $\mathbb{F}_2^l$ de poids de Hamming r en un vecteur de $\mathbb{F}_{2^m}^l$ de rang r. Cette application peut être étendue aux matrices de $\mathbb{F}_2^{n\times l}$.

$$\begin{array}{rccc} \Phi : & \mathbb{F}_2^{n\times l} & \to & \mathbb{F}_{2^m}^{n\times l} \\ & \begin{pmatrix} x_{11} & \ldots & x_{1l} \\ \vdots & \vdots & \vdots \\ x_{n1} & \ldots & x_{nl} \end{pmatrix} & \mapsto & \begin{pmatrix} \boldsymbol{\alpha}_1 x_{11} & \ldots & \boldsymbol{\alpha}_l x_{1l} \\ \vdots & \vdots & \vdots \\ \boldsymbol{\alpha}_1 x_{n1} & \ldots & \boldsymbol{\alpha}_l x_{nl} \end{pmatrix}. \end{array}$$

Les doublets issus de la distribution LPN sont de la forme $(\mathbf{a}_i, (\mathbf{s}\cdot\mathbf{a}_i)\oplus e_i)$ avec, pour tout $i \in [\![1, n]\!]$, $\mathbf{a}_i \in \mathbb{F}_2^l$ et $e_i \in \mathbb{F}_2$. On peut alors regrouper ces doublets pour les écrire sous forme matricielle : $(\mathbf{A}, \mathbf{sA}+\mathbf{e})$ avec $\mathbf{A} \in \mathbb{F}_2^{n\times l}$ et $\mathbf{e} \in \mathbb{F}_2^n$.

Soit $\mathbf{M}$ une matrice $l \times l$ à coefficients dans $\mathbb{F}_{2^m}$ et dont les coefficients génèrent un $\mathbb{F}_2$ sous-espace vectoriel de $\mathbb{F}_{2^m}$ de dimension 2. La distribution alors obtenue en considérant des éléments de la forme $(\Phi(\mathbf{A})\mathbf{M}, \Phi(\mathbf{sA}+\mathbf{e})\mathbf{M})$ est proche de la distribution $\mathsf{LRE}(m, n, l, 2r)$. En effet, on a $\Phi(\mathbf{sA}+\mathbf{e})\mathbf{M} = \mathbf{s}\Phi(\mathbf{A})\mathbf{M} + \Phi(\mathbf{e})\mathbf{M}$ et $\Phi(\mathbf{e})\mathbf{M}$ est un vecteur de poids $2r$ avec une grande probabilité. En revanche, la matrice $\Phi(\mathbf{A})\mathbf{M}$ n'est pas aléatoire.

4.3.3 Premières attaques contre le problème LRE

Différentes approches possibles

Présentons ci-après deux approches possibles d'attaques sur le problème LRE :

- ◇ Une première approche de type algébrique consistant à résoudre un système multivarié d'inconnus $\mathbf{s}$ et les vecteurs $\mathbf{e}_i$ en utilisant des bases de Gröbner. Cette approche à l'avantage d'être indépendante de la taille du corps de base q, c'est la raison pour laquelle ce type d'approche est généralement plus efficace lorsque $q > 2$. Cependant la complexité, en temps et en espace requis, dépend fortement du nombre d'inconnues dans le système. Dans notre cas, chaque doublet $(\mathbf{A}, \mathbf{sA} + \mathbf{e})$ ajoute chaque coordonnée de $\mathbf{e}$ à l'ensemble des inconnues, ce qui fait que cette approche n'est pas efficace pour résoudre de manière efficace le problème LRE.
- ◇ Une autre approche possible, de type combinatoire, consiste à tester plusieurs coordonnées de l'erreur $\mathbf{e}_i$ ou du sous-espace généré par les coordonnées de $\mathbf{e}_i$ (son support) de façon à se ramener à la résolution d'un système linéaire. Comme ces algorithmes cherchent à deviner des informations sur l'erreur, ils sont plus efficaces lorsque q est petit. Ce type d'approche exploite de manière efficace l'ensemble des doublets disponibles, ce qui en fait une bien meilleure approche contre le problème LRE.

Première attaque

La première attaque est une adaptation de l'algorithme GRS [GRS15]. Soit $(\mathbf{A}_i, \mathbf{sA}_i + \mathbf{e}_i)_{1 \leqslant i \leqslant N}$ des doublets issus du problème LRE. Soit $\mathbf{E}_1, \ldots, \mathbf{E}_N$ les supports respectifs des erreurs $\mathbf{e}_1, \ldots, \mathbf{e}_N$, *i.e.* les sous-espaces vectoriels à coefficients dans $\mathbb{F}_q$ générés par ces erreurs. L'idée est de parvenir à déterminer N $\mathbb{F}_q$-sous-espaces, $\mathbf{F}_1, \ldots, \mathbf{F}_N$ de dimension r' tels que $\mathbf{E}_i \subset \mathbf{F}_i$, pour tout $i \in [\![1, N]\!]$. Ainsi, on pourra exprimer les coordonnées $\mathbf{e}_{ij}$ de chaque $\mathbf{e}_i$ dans une base de $\mathbf{F}_i$, ce qui nous donne Nlr' inconnues sur $\mathbb{F}_q$. Supposons que $Nl > n$ et posons $\mathbf{H}$ une matrice de parité du code généré par $\mathbf{A} = (\mathbf{A}_1 \mid \cdots \mid \mathbf{A}_N)$. Si l'on pose, pour tout $i \in [\![1, N]\!]$, $\mathbf{y}_i = \mathbf{sA}_i + \mathbf{e}_i$, alors on a :

$$(\mathbf{y}_1 \mid \cdots \mid \mathbf{y}_N)\mathbf{H}^\top = \mathbf{eH}^\top \in \mathbb{F}_{q^m}^{Nl-n}.$$

Cette expression nous donne $m(Nl - n)$ équations sur $\mathbb{F}_q$. Ce système d'équations admet une unique solution avec une forte probabilité si $Nlr' \leqslant m(Nl - n)$.

La probabilité que les sous-espaces $\mathbb{F}_i$ aient été bien choisis est de :

$$\left(\begin{bmatrix} r' \\ r \end{bmatrix}_q \Big/ \begin{bmatrix} r \\ m \end{bmatrix}_q \right)^N = \mathcal{O}\left(q^{Nr(m-r')}\right).$$

En choisissant $r' = m - \lceil \frac{mn}{Nl} \rceil$, on obtient alors une complexité de :

$$\mathcal{O}\left(m^3(Nl-n)^3 q^{Nr\lceil \frac{mn}{Nl} \rceil}\right).$$

On peut améliorer la complexité de cette attaque en exploitant la structure algébrique de $\mathbb{F}_{q^m}$. Considérons la matrice $\mathbf{A}' = \left(\frac{\mathbf{A}}{\mathbf{y}}\right)$ qui génère un code $\mathbb{F}_{q^m}$-linéaire $[Nl, n+1]$ noté $\mathcal{C}'$. Par linéarité, le vecteur $\mathbf{e}$ appartient à $\mathcal{C}'$ ainsi que tous les multiples $\boldsymbol{\alpha}\mathbf{e}, \boldsymbol{\alpha} \in \mathbb{F}_{q^m}$. Il est alors possible d'exploiter cette propriété pour calculer $\mathbf{e}$ avec une complexité moyenne de :

$$C_1 = \mathcal{O}\left(m^3(Nl-n)^3 q^{Nr\lceil \frac{m(n+1)}{Nl} \rceil - m}\right) \text{ opérations dans } \mathbb{F}_q.$$

Deuxième attaque

L'idée derrière la deuxième attaque est d'essayer d'annuler certaines coordonnées de $\mathbf{y}$ par combinaison linéaire sur $\mathbb{F}_q$. En effet, soit $\mathbf{y}_i = \mathbf{s}\mathbf{A}_i + \mathbf{e}_i$ un doublet issu de la distribution LRE. Il existe une matrice $\mathbf{M}_i \in \mathbb{F}_q^{l \times l}$ telle que $\mathbf{e}_i\mathbf{M}_i = (\mathbf{e}'_1, \ldots, \mathbf{e}'_r, 0, \ldots, 0)$ où $\mathbf{e}'_1, \ldots, \mathbf{e}'_r$ génèrent le support de $\mathbf{e}$. Ainsi, $\mathbf{y}_i\mathbf{M}_i = \mathbf{s}\mathbf{A}_i\mathbf{M}_i + \mathbf{e}_i$. Maintenant séparons la matrice $\mathbf{y}_i\mathbf{M}_i$ en deux vecteurs de tailles r et $l-r$ et la matrice $\mathbf{A}_i\mathbf{M}_i$ en deux matrices de tailles $n \times r$ et $n \times (l-r)$ telles que $\mathbf{y}_i\mathbf{M}_i = (\mathbf{y}'_{i_1} \mid \mathbf{y}'_{i_2})$, $\mathbf{A}_i\mathbf{M}_i = (\mathbf{A}'_{i_1} \mid \mathbf{A}'_{i_2})$. On a ainsi $\mathbf{y}'_{i_2} = \mathbf{s}\mathbf{A}'_{i_2}$. On obtient finalement $(l-r)$ équations de s sans erreur. Si nous avons N doublets tels que $N(l-r) \geqslant n$, on peut obtenir suffisamment d'équations pour calculer $\mathbf{s}$ par inversion d'une matrice $n \times n$ sur $\mathbb{F}_{q^m}$. Chaque choix de la matrice $\mathbf{M}_i$ a une probabilité q^{-r} d'obtenir un zéro pour une coordonnée de l'erreur, l'algorithme aura donc une complexité de $C_2 = \mathcal{O}\left(n^3 m \log m \log\log m \cdot q^{nr}\right)$.

Asymptotiquement, dans le cas typique $q = 2$ et $l = m = n$, on a :

$$\log_2 C_1 = \tilde{\mathcal{O}}(rn + r - n)$$
$$\log_2 C_2 = \tilde{\mathcal{O}}(rn).$$

4.4 Le protocole $\mathrm{HB}_{\mathsf{LRE}}$

4.4.1 Présentation du protocole

Le protocole HB [HB01] est un protocole d'authentification basé sur le problème LPN. C'est un shéma symétrique très utilisé pour des architectures restreintes qui a été largement étudié. Dans cette section, nous allons introduire le protocole $\mathrm{HB}_{\mathsf{LRE}}$, une variante du protocole HB basée sur le problème LRE. Définissons tout d'abord ce que l'on entend par protocole d'authentification symétrique. Un protocole d'authentification symétrique est défini par une paire d'algorithmes (P, V) partageant un secret $\mathbf{s}$. L'interaction entre le prouveur P et le vérifieur V, étant donné les inputs $\mathbf{x}$ et $\mathbf{y}$, est notée $\langle P(\mathbf{x}), V(\mathbf{y})\rangle$. Le vérifieur acceptera le prouveur (que l'on note $\langle P, V\rangle = \mathsf{acceptation}$) lorsque $P(\mathbf{x}) = P(\mathbf{y})$ et le rejettera dans le cas contraire.

Définition 58 (Protocole d'authentification symétrique)**.** *Un protocole d'authentification symétrique est une paire d'algorithmes probabilistes en temps polynomial (P, V) partageant un secret $\mathbf{s}$ tels que :*

- ◇ *(i) quel que soit $\mathbf{s}$, $\langle P(\mathbf{s}), V(\mathbf{s})\rangle = \mathsf{acceptation}$*
- ◇ *(ii) pour chaque doublet $(\mathbf{x}, \mathbf{y})$ avec $\mathbf{x} \neq \mathbf{y}$, $\langle P(\mathbf{x}), V(\mathbf{y})\rangle = \mathsf{acceptation}$ avec une probabilité négligeable.*

Le protocole $\mathrm{HB}_{\mathsf{LRE}}$ est une des applications possibles du problème LRE et illustre comment ce dernier peut-être utilisé pour la construction de protocoles cryptographiques. Il est intéressant de constater que contrairement au protocole HB initial, le protocole $\mathrm{HB}_{\mathsf{LRE}}$ ne souffre pas de problèmes de faux négatifs.

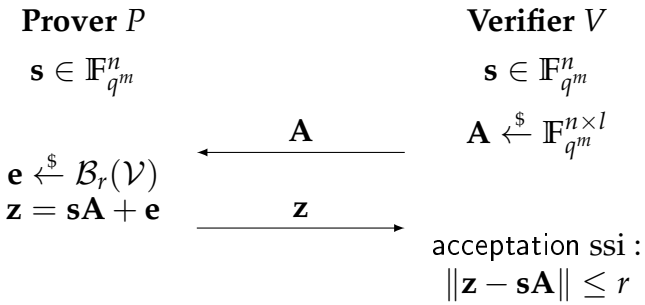

FIGURE 4.2 – Le protocole $\mathrm{HB}_{\mathrm{LRE}}$

De nombreuses variantes du protocole HB ont été proposées : se référer à [JW+05, GRS08, KPV+17] pour des exemples de dérivés du protocole initial et à [KSN17] pour un état de l'art sur les protocoles HB basés sur le problème LPN. Chacun

de ces protocoles augmente la sécurité ou l'efficacité du protocole HB de différentes façons. La plupart de ces variantes sont probablement transposables au problème LRE.

4.4.2 Modèles de sécurité

La notion de sécurité principale concernant les protocoles d'authentification symétrique est la sécurité contre les usurpations d'identités. Cette notion sera décrite suivant trois modèles de sécurité différents : modèle passif, modèle actif et contre l'attaque de l'homme du milieu (MITM - Man in the middle attack) [LM13]. Ces modèles peuvent être formalisés comme un jeu en deux étapes : une phase d'apprentissage suivie d'une phase d'usurpation. Selon le modèle considéré, l'adversaire peut avoir accès à davantage d'oracles, et ainsi à davantage de pouvoir.

- ◇ Dans le modèle passif, l'adversaire a seulement accès aux échanges entre le prouveur et le vérifieur honnêtes durant la phase d'apprentissage. Durant la phase d'usurpation, l'adversaire essayera de se faire passer pour un prouveur interagissant avec un vérifieur honnête.
- ◇ Le modèle actif est un modèle de sécurité plus fort, où l'attaquant à accès non seulement aux échanges disponibles dans le modèle passif, mais il peut également interagir avec un prouveur honnête (dans le modèle de détection) ou avec à la fois le prouveur honnête et le vérifieur (modèle préventif). Pendant la phase d'usurpation, l'adversaire ne pourra interagir qu'avec le vérifieur.
- ◇ Enfin, le modèle MITM, présente le modèle de sécurité le plus fort. En effet, dans ce modèle, l'attaquant à accès aux échanges et peut intéragir avec le prouveur et le vérifieur durant les phases d'apprentissage et d'usurpation.

4.4.3 Sécurité contre les attaques passives

Prouvons ci-après la sécurité du protocole $\mathrm{HB}_{\mathsf{LRE}}$ dans la modèle passif. De manière informelle, il s'agit de montrer que si un attaquant est capable de casser la sécurité du protocole dans ce modèle, alors il sera capable de résoudre de façon efficace le problème LRE. De la même manière que dans [KSS10], décrivons de

façon formelle le jeu de sécurité associé au modèle passif :

- ◇ **Mise en place :** Génération de la clé partagée ($\mathbf{s} \xleftarrow{\$} \mathbb{F}_{q^m}^n$) et envoie au prouveur P et au vérifieur V.
- ◇ **Phase d'apprentissage :** L'adversaire $\mathcal{A}$ peut accéder à un nombre polynomial d'échanges du protocole. On notera ces échanges par $\mathsf{trans}(\langle P, V \rangle)$.
- ◇ **Phase d'usurpation :** L'adversaire $\mathcal{A}$ joue le rôle du prouveur et interagit avec le vérifieur V. On note par $\langle \mathcal{A}, V \rangle = \mathsf{acceptation}$ une exécution du protocole où le vérifieur V authentifie l'adversaire $\mathcal{A}$.

On dira que l'adversaire remporte le jeu si le vérifieur V accepte $\mathcal{A}$ à la fin du protocole.

Définition 59 (Sécurité contre les attaques passives)**.** *Un protocole d'authentification est sécurisé contre les attaques passives si, pour tout adversaire $\mathcal{A}$, l'avantage $Adv^{passive}_{\mathcal{A},\mathrm{HB}_{\mathsf{LRE}}}$ est négligeable, avec :*

$$Adv^{passive}_{\mathcal{A},\mathrm{HB}_{\mathsf{LRE}}} = \mathbb{P}\left[\langle \mathcal{A}, V \rangle = \mathsf{acceptation} \mid \mathbf{s} \xleftarrow{\$} \mathbb{F}_{q^m}^n \ ; \ \mathsf{trans}(\langle P, V \rangle)\right]$$

Soit $A_{\mathbf{s},\mathbf{r}} = \{(\mathbf{A}, \mathbf{sA} + \mathbf{e}) \mid \mathbf{A} \xleftarrow{\$} \mathbb{F}_{q^m}^{l \times n} \ ; \ \mathbf{e} \xleftarrow{\$} \mathcal{B}_r^l(\mathbb{F}_{q^m})\}$ l'ensemble des instances du problème LRE et $\mathcal{U}$ la distribution uniforme sur $\mathbb{F}_{q^m}^{l \times n} \times \mathbb{F}_{q^m}^l$.

Théorème 7. *S'il existe un adversaire $\mathcal{A}$ interceptant au plus k exécutions du protocole $\mathrm{HB}_{\mathsf{LRE}}$, en un temps t et atteignant $Adv^{passive}_{\mathcal{A},\mathrm{HB}_{\mathsf{LRE}}} = \delta$, alors il existe un algorithme D permettant de réaliser $(k+1)$ requêtes d'oracle, en un temps en $\mathcal{O}(t)$ et tel que :*

$$\left| \mathbb{P}[D^{A_{\mathbf{s},\mathbf{r}}}(1^n) = 1 \mid \mathbf{s} \xleftarrow{\$} \mathbb{F}_{q^m}^n] - \mathbb{P}\left[D^{\mathcal{U}}(1^n) = 1\right] \right| \geq \delta - p$$

avec $p = \left(\sum_{i=0}^{2r} \prod_{j=0}^{i-1} (q^l - q^j) \cdot \binom{m}{i}_q \right) \ / \ q^{ml}$ *où* $\binom{m}{i}_q$ *représente le coefficient binomial de Gauss i.e. le nombre de sous-espaces de dimension i dans un espace vectoriel de dimension m sur un corps fini à q éléments.*

En reprenant l'idée de la preuve du théorème n°2 de [KSS10], construisons un algorithme efficace D capable de distinguer entre des instances de $A_{\mathbf{s},\mathbf{r}}$ et des instances de la distribution uniforme $\mathcal{U}$ en considérant un adversaire $\mathcal{A}$.

Démonstration. Supponsons que D ait accès à un oracle retournant $(\mathbf{A}, \mathbf{z})$ et procédant de la manière suivante :

1. D démarre la phase d'apprentissage de $\mathcal{A}$. Chaque fois que $\mathcal{A}$ fait une requête pour obtenir un échange du protocole, D obtient $(\mathbf{A}, \mathbf{z})$ depuis son oracle et retourne cette valeur à $\mathcal{A}$.
2. Lorsque $\mathcal{A}$ est prêt pour la phase d'usurpation, D obtient un doublet $(\bar{\mathbf{A}}, \bar{\mathbf{z}})$ depuis son oracle. D envoie le challenge $\bar{\mathbf{A}}$ à $\mathcal{A}$ et reçoit en retour la réponse $\mathbf{z}'$.
3. D renvoie 1 si et seulement si $\|\bar{\mathbf{z}} - \mathbf{z}'\| \leq 2r$.

Deux cas de figure sont à considérer selon l'oracle avec lequel D interagit :

◇ Si D interagit avec $\mathcal{U}$: La probabilité que D renvoie 1 est la même que la probabilité qu'un vecteur choisi aléatoirement de taille l soit dans la boule de rayon $2r$ centrée en 0 (voir [Loi06]) :

$$p = \frac{|\mathcal{B}^l_{q^m}(\mathbf{0}, 2r)|}{|\mathbb{F}^l_{q^m}|} = \frac{\sum_{i=0}^{2r} \left(\prod_{j=0}^{i-1} (q^l - q^j) \begin{bmatrix} m \\ i \end{bmatrix}_q \right)}{q^{ml}}.$$

Cette probabilité est négligeable tant que les paramètres q, m, r, l et n sont choisis correctement.

◇ Si D interagit avec $A_{\mathbf{s},\mathbf{r}}$: Soit $\mathbf{z}^* = \mathbf{sA}$ la réponse du prouveur ne contenant aucune erreur. On a $\mathbb{P}[\|\bar{\mathbf{z}} - \mathbf{z}^*\| \leq r] = 1$ car $\bar{\mathbf{z}}$ est distribué exactement de la même façon qu'une réponse d'un prouveur honnête et $\mathbb{P}[\|\mathbf{z}' - \mathbf{z}^*\| \leq r] = \delta$ car $\mathcal{A}$ parvient à usurper le prouveur dans ces cas là. Par suite, D renvoie 1 avec une probabilité d'au moins δ car :

$$\begin{aligned} \mathbb{P}[\|\bar{\mathbf{z}} - \mathbf{z}'\| \leq 2r] &\geq \mathbb{P}\left[\|\bar{\mathbf{z}} - \mathbf{z}^*\| \leq r \cap \|\mathbf{z}^* - \mathbf{z}'\| \leq r) \right] \\ &= \mathbb{P}[(\|\mathbf{z}^* - \mathbf{z}'\| \leq r)] \\ &= \delta. \end{aligned}$$

Ainsi, D est capable de distinguer entre des instances issue de $A_{\mathbf{s},\mathbf{r}}$ et de $\mathcal{U}$ avec une probabilité d'au moins $\delta - p$. □

Dans le cas où $l = m = n$, on a $p \approx q^{2r(2n-2r)-n^2} = q^{4r(n-r)-n^2}$.

4.4.4 Comparaison avec le protocole HB initial

Pour atteindre un niveau de sécurité de 80 bits, des travaux précédents [LF06, BL12, GJL14, ZJW16, BV16, ZG17]) suggèrent l'utilisation d'instances $(592, 1/8)$ du problème LPN, *i.e.* une clé secrète de 592 bits et un taux d'erreur égal à $\frac{1}{8}$. Il est à noter que ces paramètres pourraient ne pas être suffisants selon de recents travaux [EKM17]. Etant donné un taux d'erreurs de $\frac{1}{8}$, pour atteindre un taux d'erreurs (completeness) inférieur à 2^{-40} et un taux de faux positifs (soundness) de 2^{-80}, il faudrait instancier le protocole HB 441 fois [LF06]. Ainsi, avec le protocole HB initial, 32 Kio de données devraient être échangées entre le prouveur et le récepteur pour atteindre un tel niveau de sécurité. En considérant le problème LRE, l'on peut choisir une instance $(m, n, l, r) = (18, 18, 18, 6)$ pour atteindre 80 bits de sécurité. Ces paramètres entrainent alors une taille de clé secrète de 324 bits et ne nécéssitent que 648 bits de données échangées durant l'exécution du protocole. Pour 128 bits de sécurité, on pourra utiliser une instance $(20, 20, 20, 7)$ du problème LRE, ce qui correspond à 400 bits pour la taille de clé et 800 bits de données échangées. L'utilisation du problème LRE permet donc de réduire de façon drastique la quantité de données échangées pour un niveau de sécurité donné comparé au protocole initial, la contrepartie étant que celui-ci requiert davantage de capacités de calculs que son prédécesseur.

Chapitre 5

Implémentation pour architectures 32 bits

Le NIST a recommandé, durant le processus de standardisation de shémas post-quantiques, le portage des implémentations soumises à des architectures plus restreintes dont la taille maximum des registres n'excède pas 32 bits, la taille du registre principal des ordinateurs de bureau utilisés de nos jours possédant quant à eux 64 bits. Les différents algorithmes des schémas RQC et ROLLO ont par conséquent dû être adaptés pour satisfaire cette contrainte.

La section 5.1 décrit les différents microcontrôleurs utilisés. La section 5.2 explicite les modifications relatives à la multiplication des éléments sur le corps $\mathbb{F}_{q^m}$. La section 5.3 décrira la méthode utilisée pour réduire le produit de deux éléments modulo un polynôme unitaire de degré m, afin d'obtenir *in fine* un nouvel élément de $\mathbb{F}_{q^m}$. La section 5.4 explicitera le produit de deux vecteurs de l'espace vectoriel $\mathcal{V}$. La section 5.5 décrira la librairie RBC qui intègre les travaux sur le portage sur architecture 32 bits décrits précédemment. Enfin, la section 5.6 présentera les performances obtenues sur microcontrôleur à partir de l'api pqm4 [KRSS] et les mettra en perspective avec les performances d'autres schémas post-quantiques candidats à l'appel à standardisation.

5.1 Microcontrôleurs visés

Trois cartes de développement différentes ont été considérées :

La première carte m'a été recommandée par mes collègues de Worldline car

simple pour débuter dans le développement IoT (Internet of Things). Il s'agissait du Particle Photon [Par], une petite carte dotée d'une puce Wi-Fi Cypress, d'un microcontrôleur STM32 ARM Cortex M3 contenant un processeur cadencé à 120 MHz (STM32F205RGY6), de 128 ko de mémoire RAM et de 1 mo de mémoire flash.

La seconde a été choisie en fonction des recommandations du NIST d'implémenter les schémas post-quantiques en compétition sur des cartes dotées d'un processeur ARM Cortex-m4. Notre choix s'est porté sur la carte de développement STM32 Nucleo-144 [STMa]. Celle-ci est dotée du microcontrôleur STM32F439ZI [STMd] comportant un processeur cadencé jusqu'à 180 MHz, 256 ko de mémoire RAM et 2 mo de mémoire flash. Ce microcontrôleur nous a semblé intéressant de part son générateur d'aléatoire intégré et la possibilité d'effectuer certaines fonctions de chiffrement et de hachage au niveau hardware (AES-128/192/256, en mode GCM et CCM, Triple DES, et hash MD5, SHA-1 et SHA-2).

Enfin, la mise en place d'une librairie dédiée à la cryptographie post-quantique pour ARM Cortex-M4, la librairie pqm4 [KRSS], visant à pouvoir tester et comparer les performances des différents schémas soumis à l'appel à projet du NIST et ciblant spécifiquement la carte STM32F407G-DISC1 [STMc] m'a amené à implémenter également les schémas sur cette dernière. Elle dispose du microcontrôleur STM32F407VGT6 [STMb] doté d'un processeur ARM Cortex-M4 cadencé à 168 MHz, de 196 ko de mémoire RAM et jusqu'à 1 mo de mémoire flash. Elle intègre de plus un générateur d'aléatoire intégré.

5.2 Multiplication sur $\mathbb{F}_{q^m}$

Dans les implémentations optimisées de RQC et ROLLO disponibles sur [MAB+20b, MAB+20a] et ciblant des architectures disposant d'une unité logique d'arithmétique (ALU - arithmetic logic unit) de 64-bits, la multiplication multiprécision entre des éléments de $\mathbb{F}_{q^m}$ s'effectue en utilisant plusieurs multiplications sans retenue (CLMUL - carry-less multiplication) sur des entiers de longueurs 64 bits puis en réassemblant les différentes sommes ensemble pour obtenir le résultat final de la multiplication.

Précisons qu'une multiplication sans retenue correspond simplement à la multiplication qui s'opère entre deux polynômes, la seule différence avec la mul-

tiplication classique entre des entiers étant que l'on ne propage pas de retenue lorsque l'on additionne les sous-résultats entre-eux. En d'autres termes, les sous-résultats intermédiaires sont xorés (XOR *i.e.* ou exclusif) au lieu d'être ajoutés.

Ces implémentations exploitent le jeu d'instructions AVX (Advanced Vectors Extensions) disponible sur la plupart des microprocesseurs Intel et AMD de nos jours.

Malheureusement, cette stratégie pour effectuer la multiplication multiprécision entre des éléments de $\mathbb{F}_{q^m}$ ne serait pas efficace sur une carte disposant d'un processeur ARM Cortex-M4, car la multiplication sans retenue ne peut pas être effectuée au niveau hardware sur ce type de carte. Par conséquent, il nous a fallu choisir une méthode alternative pour effectuer ces multiplications, et notre choix s'est porté sur l'algorithme « left-to-right comb method with windows of width w »[HMV06]. Décrivons la manière dont fonctionne cet algorithme :

Soit a et b deux polynômes de degré au plus $m-1$ et c le résultat de la multiplication sans retenue de a par b, de degré maximal $2m-2$.

$$a(z) = a_0 + a_1 z + \cdots + a_{m-1} z^{m-1} \text{ et } b(z) = b_0 + b_1 z + \cdots + b_{m-1} z^{m-1}$$

En mémoire, les bits représentants les coefficients de chaque polynôme sont stoqués par paquets de $W = 32$ bits. Considérons par exemple $m = 163$ et $w = 4$ pour illustrer notre propos, notons $A[0]$ les 32 bits de poids faibles de a et par $A[5]$ les derniers trois bits non nuls de a, comme illustré sur la figure 5.1.

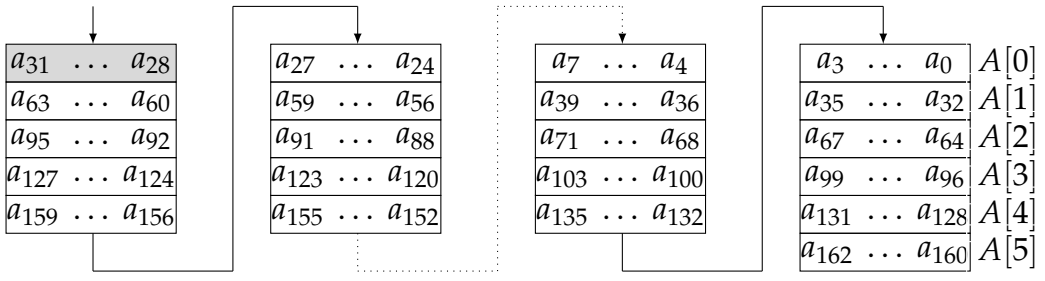

FIGURE 5.1 – Représentation de la fenêtre glissante pour gérer les bits de a

L'idée derrière cet algorithme est de traiter les bits de a par paquets de w bits, en commençant pour les bits de poids les plus forts parmi chacun des $A[i]$ et de poursuivre jusqu'aux bits de poids faibles. La première étape consiste à pré-

calculer tous les produits $u(z) \cdot b(z)$, pour tous les polynômes $u(z)$ de degré inférieur à w possibles. Une fois cette étape de précalcul terminée, les bits de a sont lus par paquets de w bits et le produit avec b est effectué, en exploitant la table de précalculs précédemment générée, jusqu'à ce que tous les bits de a aient été lus.

La figure 5.1 donne une description de la façon dont les bits de a sont traités, pour une description détaillée, voir algorithme 1.

Notation 6. *Si* $C = (C[n], \ldots, C[1], C[0])$ *désigne un tableau de valeurs, alors* $C\{j\}$ *désignera le tableau tronqué* $(C[n], \ldots, C[j+1], C[j])$.

Algorithm 1 Left-to-right comb method with windows of width w [HMV06]

Input : Binary polynomials $a(z)$ and $b(z)$ of degree at most $m-1$.
Output : $c(z) = a(z) \cdot b(z)$.
Compute $B_u = u(z) \cdot b(z)$ for all polynomials $u(z)$ of degree at most $w-1$.
$C \leftarrow 0$.
for $k \leftarrow \frac{W}{w} - 1,\ 0$ **do**
    for $j \leftarrow 0,\ \lceil \frac{m}{W} \rceil - 1$ **do**
        Let $u = (u_{w-1}, \ldots, u_1, u_0)$ where u_i is bit $(wk+i)$ of $A[j]$.
        Add B_u to $C\{j\}$.
    end for
    If $k \neq 0$ then $C \leftarrow C \cdot z^w$.
end for
return C

5.3 Réduction des éléments de $\mathbb{F}_{q^m}$

Dans cette section, nous allons décrire la méthode que nous avons utilisée pour réduire des polynômes de degré au plus $2m-2$ modulo un polynôme de degré m. Celle-ci est inspirée de [HMV06] et l'algorithme de réduction qui en découle doit être adapté à chaque valeur de m considérée.

Ecrivons le module sous la forme $f(z) = z^m + r(z)$ où $r(z)$ désigne un polynome de degré au plus $m-1$. Nous avons :

$$\begin{aligned} c(z) &= c_{2m-2}z^{2m-2} + \cdots + c_m z^m + c_{m-1}z^{m-1} + \cdots + c_1 z + c_0 \\ &= (c_{2m-2}z^{m-2} + \cdots + c_m)z^m + c_{m-1}z^{m-1} + \cdots + c_1 z + c_0 \\ &\equiv (c_{2m-2}z^{m-2} + \cdots + c_m)r(z) + c_{m-1}z^{m-1} + \cdots + c_1 z + c_0 \mod f(z). \end{aligned}$$

Pour les schémas ROLLO et RQC, le module considéré est généralement un trinôme ou un pentanôme dont les termes au milieu sont proches les uns des autres. Cette caractéristique permet de considérer la réduction d'un polynôme en sommant des sous-mots au polynôme de départ. A chaque étape, une somme pour chaque monome intérieur du trinôme ou pentanôme de départ est impliquée, soit deux sommes dans le cas d'un trinôme et quatre sommes dans le cas d'un pentanôme. Considérons un exemple pour illustrer nos propos avec $m = 163$ et $f(z) = z^{163} + z^7 + z^6 + z^3 + 1$.

Soit $c(z)$ un polynôme de degré maximum égal à $2m - 2$, *i.e.* de degré 324. Considérons le bloc $C[9]$ de c correspondant au polynôme $c_{319}z^{319} + \cdots + c_{289}z^{288}$. On a :

$$\begin{aligned}
z^{288} &\equiv z^{132} + z^{131} + z^{128} + z^{125} \mod f(z) \\
z^{289} &\equiv z^{133} + z^{132} + z^{129} + z^{126} \mod f(z) \\
&\vdots \\
z^{319} &\equiv z^{163} + z^{162} + z^{159} + z^{156} \mod f(z).
\end{aligned}$$

En considérant les quatre colonnes du côté droit des congruences décrites ci-dessus, on peut voir que la réduction du bloc $C[9]$ peut être effectuée en ajoutant $C[9]$ quatre fois à C, tel que le bit de poids faible de $C[9]$ soit ajouté respectivement au bit 132, 131, 128 et 125, comme illustré par la figure 5.2.

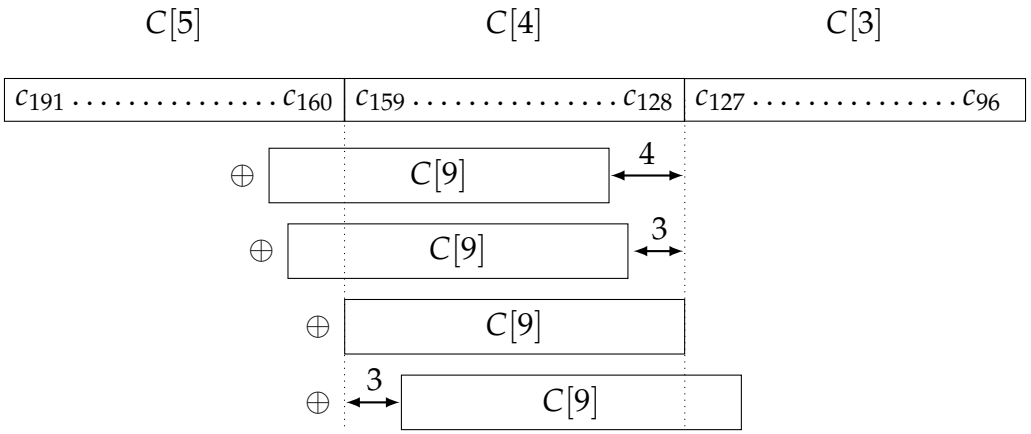

FIGURE 5.2 – Réduction du bloc de 32 bits $C[9]$ modulo $f(z) = z^{163} + z^7 + z^6 + z^3 + 1$

On peut remarquer que cette méthode implique de diviser le bloc $C[9]$ en diffé-

rentes sous-parties afin de pouvoir les ajouter aux blocs de C correspondants.

5.4 Multiplication sur $\mathbb{F}_{q^m}^n$

Au vu des paramètres requis pour les schémas RQC et ROLLO, nous avons choisi d'effectuer la multiplication entre deux vecteurs sur $\mathcal{V} = \mathbb{F}_{q^m}^n$ en utilisant un mixte entre la méthode Karatsuba et la multiplication naïve en-dessous d'un certain seuil. Nous avons utilisé un simple Karatsuba récursif, ce qui signifie qu'à la fin de chaque étape de la récursion le nombre de coefficients des polynômes mis en jeux est divisé par deux. Bien que le principe de la méthode de Karatsuba soit simple à comprendre, l'algorithme de multiplication qui en découle est très efficace.

Rappelons très succintement le principe de la méthode de Karatsuba : soit a et b deux polynômes de degré 1. La multiplication naïve résulterait en 4 multiplications et une addition sur $\mathbb{F}_{q^m}$. En effet, $a(z) \cdot b(z) = (a_0 + a_1 z)(b_0 + b_1 z) = a_0 b_0 + (a_0 b_1 + a_1 b_0)z + a_1 b_1 z^2$. Il faut donc calculer les produits $a_0 b_0$, $a_0 b_1$, $a_1 b_0$ et $a_1 b_1$, et effectuer l'addition $a_0 b_1 + a_1 b_0$.

L'astuce derrière la méthode de Karatsuba est de parvenir à effectuer cette même multiplication en recourant à moins de multiplications sur $\mathbb{F}_{q^m}$, quitte à augmenter le nombre d'additions sur $\mathbb{F}_{q^m}$ requises, l'addition étant bien plus rapide à effectuer qu'une multiplication. La méthode de Karatsuba se base sur l'observation suivante : $a_0 b_1 + a_1 b_0 = (a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1$. Ainsi, il suffit de calculer les produits $a_0 b_0$, $a_1 b_1$ et $(a_0 + a_1)(b_0 + b_1)$ et d'effectuer les sommes $a_0 + a_1$, $b_0 + b_1$, $X - a_0 b_0$ et $Y - a_1 b_1$ pour pouvoir déterminer tous les coefficients du polynôme produit. Il suffit donc de 3 multiplications dans $\mathbb{F}_{q^m}$ et de 4 additions dans $\mathbb{F}_{q^m}$ pour effectuer la multiplication entre les polynômes a et b. On a gagné 1 multiplication au détriment de 3 additions supplémentaires par rapport à la méthode naïve. Il suffit donc que le coût pour effectuer 3 additions dans $\mathbb{F}_{q^m}$ soit inférieur au coût d'une multiplication dans $\mathbb{F}_{q^m}$ pour que cette méthode surpasse la méthode naïve, ce qui est généralement le cas. Cette méthode peut bien sûr être généralisée aux polynômes de n'importe quel degré, et peut être utilisée de façon récursive.

Comme démontré dans [WP06], le simple Karatsuba récursif, celui que nous avons choisi d'effectuer, est quasiment toujours la meilleure version de Karatsuba pour minimiser le nombre de multiplications sous-jacentes à effectuer

dans $\mathbb{F}_{q^m}$.

5.5 Intégration à la librairie RBC

La librairie rbc-lib [AB+20] est une librairie écrite en langage C dédiée à la métrique rang mettant l'accent sur les performances tout en ne négligeant pas l'aspect facilité d'utilisation. La librairie RBC est composée :

- ◇ d'une couche principale (*core layer*) permettant d'effectuer des opérations arithmétiques sur des éléments, des vecteurs et des polynômes sur $\mathbb{F}_{2^m}$ et disposant de certaines fonctions adaptées à la métrique rang
- ◇ d'une couche code (*code layer*) permettant l'accès aux implémentations des encodages et décodages des principaux codes correcteurs utilisés en métrique rang, à savoir les codes de Gabidulin et les codes LRPC
- ◇ Une couche schéma (*scheme layer*) donnant l'accès aux implémentations des schémas RQC et ROLLO, des schémas cryptographiques candidats à l'appel à standardisation du NIST

Dual API.

L'API offerte par rbc-lib vise deux types d'audience différents :

La première, destinée aux utilisateurs finaux (*end users*) a pour but de permettre l'utilisation des schémas cryptographiques de la librairie ainsi que des implémentations liées aux benchmarks desdits schémas.

La deuxième, dédiée aux utilisateurs plus avancés (*advanced users*) permet de disposer de l'ensemble des fonctions dédiées à la métrique rang, d'ajouter d'autres schémas reposant sur la métrique rang et de contribuer à l'évolution de la librairie.

Les utilisateurs auront accès aux fonctions de la couche schéma de la librairie, tandis que les utilisateurs avancés auront accès à l'API toute entière, *i.e.* aux trois couches décrites précédemment.

Corps finis supportés.

La librairie ne supporte dans l'état actuel (RBC v1.0) que les corps finis de la forme $\mathbb{F}_{q^m}$ avec $q = 2$, ce qui représente la majorité des corps finis généralement utilisés en métrique rang. Concernant l'arithmétique sur les corps finis,

deux approches sont possibles : fournir des algorithmes génériques à toutes les valeurs de m où des algorithmes spécifiques, plus optimisés, pour certaines valeurs de m spécifiques. La première approche permet plus de simplicité et permet un usage plus étendu de la librairie, tandis que la deuxième approche, celle adoptée par RBC, permet d'optimiser un certain nombre d'algorithmes et offre donc de meilleures performances. Ce choix a un impact limité sur l'utilisateur final, et bien qu'il ajoute un peu de complexité pour les utilisateurs avancés, le système de préprocessing utilisé par la librairie permet de limiter le nombre de fonctions et d'algorithmes à considérer *in fine*.

Préprocessing et compilation.

La librairie dispose d'un système de préprocessing et de compilation par l'intermédiaire de scripts Python facilitant le développement pour les utilisateurs avancés et permettant de personnaliser les fonctions disponibles pour les utilisateurs, et ainsi de réduire la taille en mémoire nécessaire au projet de chaque utilisateur.

Elle utilise un système de template dans la couche principale permettant de générer des codes optimisés pour chaque corps fini tout en évitant la redondance de code. De plus, elle permet la specialisation automatique du code source pour les couches code et schéma, permettant ainsi à un utilisateur d'écrire du code générique qui sera automatiquement traduit pour les instances spécifiques des corps finis spécifiés dans le fichier de configuration. Ainsi, il est possible d'écrire du code générique tout en conservant la possibilité d'utiliser plusieurs instances différentes d'un schéma simultanément. A titre d'exemple, il est possible d'utiliser une implémentation utilisant uniquement ROLLOI puis écrire un programme appelant à la fois ROLLOI-128 et ROLLOI-192 dépendant respectivement de $\mathbb{F}_{2^{67}}^{83}$ et de $\mathbb{F}_{2^{79}}^{97}$ simultanément tout en évitant la redondance de code.

Le système de préprocessing permet également de personnaliser la compilation de la librairie en spécifiant les options désirées dans le fichier de configuration. L'utilisateur a pour l'instant le choix entre les architectures x86 (32 bits), x64 (64 bits) avec la possibilité d'utiliser la fonction CLMUL (carry-less multiplication, *i.e.* multiplication sans retenue, très utile pour les opérations sur les corps finis) et l'AVX2 (Advanced Vector Extensions, jeu d'instructions permettant la vectorisation, *i.e.* la capacité d'un registre à pouvoir effectuer plusieurs opérations mathématiques simultanément, opérations qui nécessitent en temps

normal l'utilisation d'un registre plus petit). Le système de préprocessing génèrera alors le code en conséquence de façon à fournir les meilleurs algorithmes disponibles selon l'architecture visée. Les utilisateurs ont aussi la possibilité de choisir les cryptosystèmes qui leur sont nécessaires, limitant ainsi l'impact en mémoire des fichiers générés.

Implémentations externes.

La librairie dépend de plusieurs primitives cryptographiques en dehors de celles liées spécifiquement à la métrique rang, comme l'accès à un générateur de nombres aléatoires, un seedexpander, SHA2, FIPS202 ou encore l'AES. Les implémentations de ces primitives sont extraites de BearSSL [Por16], OpenSSL [Ope], PQClean [PQC], SUPERCOP [Sup] projects et [Nis16b, Gue10]. De plus, la structure de Minunit [Min14] et la librairie $\texttt{mp}\mathbb{F}_q$ sont utilisées afin de réaliser les tests unitaires.

5.6 Performances de RQC et ROLLO

Comme évoqué précédemment, les implémentations des schémas RQC et ROLLO ont été intégrées à la librairie pqm4 en vu de pouvoir :

- ◇ Evaluer les performances de ces schémas de façon fiable grâce au code mis en place par les équipes à l'origine de la librairie pqm4. Leur API permet des mesures de cycles avec un processeur cadencé à la vitesse de 24 MHz, pour des mesures plus fiables.
- ◇ Permettre de comparer facilement les perfomances de ces schémas vis-à-vis des autres schémas post-quantiques déjà intégrés à pqm4.

Dans cette section, nous allons donc présenter les performances des schémas RQC et ROLLO sur la carte de développement STM32F407G-DISC1 et les comparer aux autres schémas post-quantiques d'ores et déjà intégrés. Seules deux implémentations des schémas RQC et ROLLO sur microcontrôleur ont été soumises dans la littérature.

La première est une implémentation efficace de ROLLO-I (*i.e.* LAKE) [LMB+19] qui exploite un crypto co-processeur ARM SecurCore SC300 pour effectuer les opérations dans $\mathbb{F}_{q^m}$. Le fait que cette implémentation utilise des paramètres de ROLLO qui ne sont plus à jour et de l'utilisation d'un co-processeur pour

booster la vitesse d'exécution des opérations sur $\mathbb{F}_{q^m}$ fait que leurs mesures ne peuvent pas être directement comparées aux nôtres.

La deuxième est tirée de la librairie pqm4 [KRSS]. Elle est basée sur les implémentations soumises dans le processus de standardisation du NIST ciblant les architectures x64. Nos implémentations, ciblant spécifiquement les architectures x86, offrent une amélioration conséquente par rapport à ces dernières. En effet, nous avons pu observer que les mesures en cycle étaient plus de deux fois plus rapides avec les adaptations que nous avons effectuées.

Les mesures que nous présentons utilisent les scripts de benchmark disponibles dans l'api pqm4 et suivent donc la méthodologie présentée dans [KRSS]. En particulier, toutes les mesures sont effectuées avec un processeur cadencé à 24 MHz. Pour chaque schéma, 100 exécutions ont été effectuées en utilisant le compilateur `arm-none-eabi-gcc` version 10.1.0.

Les temps moyens pour ROLLO-I, ROLLO-II et RQC sont présentés dans les tableaux 5.1, 5.2 et 5.3 respectivement. Aucune valeur n'est reportée pour RQC-256 car l'implémentation actuelle dépasse la mémoire disponible du microcontrôleur. Les temps pour ROLLO-III ne sont pas non plus reportés car ce schéma a été retiré par ses auteurs durant le processus de standardisation du NIST. Le tableau 5.4 décrit les performances de plusieurs schémas d'échange de clés post-quantiques intégrés dans pqm4 dont l'implémentation est en langage C et atteignant une sécurité de 128 bits. Le tableau 5.5 présente quant à lui les performances de schémas dont l'implémentation est optimisée pour spécifiquement cibler les architectures ARM Cortex-m4 en utilisant le langage assembleur.

Schéma	Keygen	Encaps	Decaps
ROLLO-I-128	16 927 603	1 926 332	7 009 943
ROLLO-I-192	22 466 486	2 271 969	7 839 572
ROLLO-I-256	45 424 004	3 769 338	15 039 516

TABLE 5.1 – Performances de ROLLO-I sur ARM Cortex-M4 en cycles

Schéma	Keygen	Encaps	Decaps
ROLLO-II-128	85 063 257	6 844 408	17 321 266
ROLLO-II-192	128 155 854	9 687 469	24 668 141
ROLLO-II-256	152 145 827	10 867 964	29 573 929

TABLE 5.2 – Performances de ROLLO-II sur ARM Cortex-M4 en cycles

Schéma	Keygen	Encaps	Decaps
RQC-128	5 756 747	11 340 541	71 551 978
RQC-192	12 324 464	24 632 358	150 108 887

TABLE 5.3 – Performances de RQC sur ARM Cortex-M4 en cycles

Schéma	Keygen	Encaps	Decaps
frodokem640shake	80 587 016	79 696 017	79 144 730
kyber512	653 616	883 740	981 642
newhope512cca	715 680	1 128 510	1 186 054
ntruhps2048509	106 694 544	2 838 551	7 766 558
ntrulpr653	56 520 202	112 440 360	168 157 956
sikep434	672 303 199	1 100 796 989	1 174 307 957
sntrup653	599 438 684	56 563 524	170 044 505

TABLE 5.4 – Performances de certains KEM sur ARM Cortex-M4 en cycles. Implémentations en C pur, 128 bits de sécurité.

Schéma	Keygen	Encaps	Decaps
frodokem640aes	48 350 369	47 135 457	46 604 758
kyber512	470 998	596 970	555 224
newhope512cca	582 009	870 621	825 352
ntruhps2048509	77 457 221	606 804	555 866
sikep434	48 264 153	78 912 215	84 277 568

TABLE 5.5 – Performances de certains KEM sur ARM Cortex-M4 en cycles. Langage C et assembleur, 128 bits de sécurité.

Chapitre 6

Conclusion

La conception d'ordinateurs quantiques et la mise en oeuvre d'algorithmes quantiques tels que ceux de Shor et de Grover sont une menace importante pour la sécurité de nos communications, en particulier en ce qui concerne la cryptographie à clé publique. De plus, les objets connectés sont de plus en plus nombreux dans notre quotidien et les enjeux autour du respect de la vie privée et de la sécurité de ces objets sont des préoccupations majeures de nos sociétés modernes. La cryptographie dite à bas coût est par conséquent un enjeu crucial pour assurer la confidentialité, l'intégrité et l'authenticité de nos communications.

Cette thèse m'a permis d'explorer ces deux problématiques en m'orientant principalement sur l'utilisation des codes correcteurs d'erreurs et de la métrique rang. Elle a eu un aspect théorique d'une part avec l'élaboration d'un Hash Proof System et l'introduction d'un nouveau problème cryptographique, à savoir le problème LRE, et un aspect pratique d'autre part par la découverte du développement sur microcontrôleur et des adaptations nécessaires à la mise en œuvre d'implémentations sur une architecture 32 bits.

La première partie a concerné la conception d'un Hash Proof System basé sur les codes correcteurs d'erreurs. Celui-ci s'est basé sur la langage des chiffrés valides de RQC, chiffrement à clé publique post-quantique candidat à l'appel à standardisation du NIST. Ce Hash Proof System à l'avantage d'être KV, ce qui signifie que la génération de la clé de projection est indépendante du mot du langage considéré, et de totalement supprimer la zone problématique des éléments qui ne sont ni concernés par la propriété de *correctness*, ni concernés par

la propriété de *smoothness*, ce qui n'avait jusqu'alors jamais été atteint à notre connaissance par les constructions précédentes. Atteindre cet objectif nous a poussé à élaborer une preuve à divulgation nulle de connaissance sur les chiffrés de RQC ainsi que sur des couples de chiffrés chiffrant un même message μ en étendant les possibilités offertes par le *Rank Concatenated Stern's Protocol* [ABCG16]. Nous avons ensuite proposé deux applications à notre primitive : La conception d'un schéma d'authentification par mot de passe (PAKE) dans le modèle de l'oracle aléatoire ainsi qu'un schéma de chiffrement par témoin (Witness encryption scheme) dans le modèle standard. Le principal défaut de notre construction est son recours à une preuve de connaissance pour s'assurer de la forme des éléments manipulés. Bien que cette preuve de connaissance nous ait permis de nous défaire du gap entre l'ensemble des chiffrés valides d'un message $\mathbf{\mu}$, qui constitue le langage de notre HPS, et l'ensemble des éléments qui ne sont pas dans le langage, cette preuve de connaissance à un impact fort sur la quantité d'information à échanger lors de la conception de protocole, comme on peut le voir avec le PAKE présenté en figure 3.7. En effet, dans ce type de preuve à la Stern, un adversaire peut tromper un vérifieur honnête avec une probabilité de $\frac{2}{3}$, ce qui implique de répéter à de nombreuses reprises le protocole pour obtenir une confiance suffisante de la connaissance des valeurs considérées. La conception d'un tel Hash Proof System (*i.e.* sans gap) sans le recours à une telle preuve de connaissance ou réduisant considérablement la taille des messages à échanger est une question qui reste ouverte.

Nous avons également introduit une nouvelle variante du problème LPN qui est un problème fortement étudié dans le domaine de la cryptographie à bas coût en adaptant celui-ci à la métrique rang : Le problème Learning Rank with Errors (LRE). Nous avons étudié deux de ses propriétés : l'unicité de la solution et la réduction pire cas/cas moyen. Nous avons également discuté du caractère pseudoaléatoire de la distribution LRE. Une première application de ce problème a été présentée : le protocole d'authentification symétrique $\mathrm{HB}_{\mathsf{LRE}}$ variante du protocole HB exploitant le problème LRE. Nous avons montré que le problème $\mathrm{HB}_{\mathsf{LRE}}$ engendrait des coûts de communication extrêmement réduits en comparaison du protocole initial, la contrepartie étant qu'il nécéssite davantage de capacité de calcul. A l'avenir, les travaux entrepris sur le protocole $\mathrm{HB}_{\mathsf{LRE}}$ pourraient être étendus afin d'en améliorer les performances et la sécurité, en prenant en considération des attaques actives et de type Man In The Middle. La conception d'autres protocoles basés sur le problème LRE serait

également un champ d'étude intéressant.

Enfin, un troisième axe d'étude a consisté à adapter et implémenter, en langage C et sur différents microcontrôleurs, des schémas cryptographiques post-quantiques candidats à l'appel à standardisation du NIST. Les travaux ce sont principalement orientés sur la métrique rang et les schémas RQC [MAB+20b], Ouroroboros-R, LAKE et LOCKER (ROLLO [MAB+20a]). Toutes ces implémentations ont ensuite été adaptées pour pouvoir intégrer la librairie RBC, une librairie dédiée spécifiquement à la cryptographie basée sur la métrique rang. Les jeux de paramètres ayant évolué au fur et à mesure de la thèse, notamment lors de la découverte récentes de nouvelles attaques algébriques sur la métrique rang [BBC+20a, BBC+20b], les implémentations ont dû être adaptées à chaque fois en conséquence. Pour prolonger les travaux entrepris, de nouveaux algorithmes plus efficaces en terme de complexité peuvent éventuellement être découverts, les implémentations de ceux-ci peuvent éventuellement être améliorées, et enfin certaines fonctions peuvent être envisagées en langage plus bas niveau comme le langage assembleur.

Bibliographie

[AAB+20] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. RQC – Rank Quasi-Cyclic. 2020. `https://pqc-rqc.org/`.

[AB+20] Nicolas Aragon, Loïc Bidoux, et al. RBC : A library dedicated to Rank-Based Cryptography. 2020. `https://rbc-lib.org`.

[ABB+17] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, et al. BIKE – bit flipping key encapsulation. 2017.

[ABCG16] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A practical group signature scheme based on rank metric. In *International Workshop on the Arithmetic of Finite Fields*, pages 258–275. Springer, 2016.

[ABD+17a] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LAKE – low rank parity check codes key exchange. 2017.

[ABD+17b] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LOCKER – Low rank parity check codes encryption. First round submission to the NIST post-quantum cryptography call, 2017.

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems : New constructions and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 69–100. Springer, 2015.

[AGHT18] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2421–2425. IEEE, 2018.

[AJPS17] Divesh Aggarwal, Antoine Joux, Anupam Prakash, and Mikos Santha. Mersenne-756839. *Submission to the NIST PQC project*, 2017.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 298–307. IEEE, 2003.

[ALR18] Daniel Augot, Pierre Loidreau, and Gwezheneg Robert. Generalized gabidulin codes over fields of any characteristic. *Designs, Codes and Cryptography*, 86(8) :1807–1848, 2018.

[AMBD+18] Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64(5) :3927–3943, 2018.

[BBC+13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *Annual Cryptology Conference*, pages 449–475. Springer, 2013.

[BBC+20a] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Algebraic attacks for solving the rank decoding and minrank problems without Gröbner basis. *arXiv preprint arXiv :2002.08322*, 2020.

[BBC+20b] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Improvements of algebraic attacks for solving the rank decoding and minrank problems. 2020.

[BBD09] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. Post-quantum cryptography, 2009.

[BBDQ18] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In *IACR International Workshop on Public Key Cryptography*, pages 644–674. Springer, 2018.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In *International Conference on Applied Cryptography and Network Security*, pages 65–86. Springer, 2015.

[BCD+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo : Take off the ring ! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1006–1018, 2016.

[BCL+17] Daniel J Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, et al. Classic McEliece : conservative code-based cryptography. *NIST submissions*, 2017.

[BCLvV17] Daniel J Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime : reducing attack surface at low cost. In *International Conference on Selected Areas in Cryptography*, pages 235–260. Springer, 2017.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber : a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[BHG+16] Fabrice Ben Hamouda-Guichoux et al. *Diverse modules and zero-knowledge*. PhD thesis, Paris Sciences et Lettres, 2016.

[BHH+15] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS : practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer, 2015.

[BL12] Daniel J Bernstein and Tanja Lange. Never trust a bunny. *RFIDSec*, 7739 :137–148, 2012.

[BM92] Steven Michael Bellovin and Michael Merritt. Encrypted key exchange : Password-based protocols secure against dictionary attacks. 1992.

[BMVT78] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3) :384–386, 1978.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *International conference on the theory and applications of cryptographic techniques*, pages 139–155. Springer, 2000.

[BV16] Sonia Mihaela Bogos and Serge Vaudenay. Observations on the LPN solving algorithm from eurocrypt" 16. Technical report, 2016.

[CDK+20] Ming-shing Chen, Jintai Ding, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. 2020.

[Che96] Kefei Chen. A new identification algorithm. In Ed Dawson and Jovan Golić, editors, *Cryptography : Policy and Algorithms*, pages 244–249, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[CS02a] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[CS02b] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 45–64. Springer, 2002.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium : A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber : Module-LWR based key exchange, CPA-secure encryption and CCA-secure kem. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018.

[EKM17] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. *IACR Cryptology ePrint Archive*, 2017 :78, 2017.

[EKP+07] Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6) :522–533, 2007.

[Fey82] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1982.

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon : Fast-fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process*, 2018.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[Gab85] Ernest Mukhamedovich Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1) :3–16, 1985.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013.

[GJL14] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2014.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *International*

Conference on the Theory and Applications of Cryptographic Techniques, pages 524–543. Springer, 2003.

[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*, volume 2013, 2013.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[GRS08] Henri Gilbert, Matthew JB Robshaw, and Yannick Seurin. HB# : Increasing the security and efficiency of HB+. *Lecture Notes in Computer Science*, 4965 :361–378, 2008.

[GRS15] 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, 2015.

[GSZ11] Philippe Gaborit, Julien Schrek, and Gilles Zémor. Full cryptanalysis of the Chen identification protocol. In *International Workshop on Post-Quantum Cryptography*, pages 35–50. Springer, 2011.

[Gue10] Shay Gueron. Intel Advanced Encryption Standard (AES) new instructions set, 2010.

[GZ16] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Transactions on Information Theory*, 62(12) :7245–7252, 2016.

[Ham50] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2) :147–160, 1950.

[HB01] Nicholas J Hopper and Manuel Blum. Secure human identification protocols. In *International conference on the theory and application of cryptology and information security*, pages 52–66. Springer, 2001.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.

[HMV06] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[JAC+17] David Jao, Reza Azarderakhsh, Matt Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalili, Brian Koziel, Brian Lamacchia, Patrick Longa, et al. SIKE : supersingular isogeny key encapsulation. 2017.

[JW+05] Ari Juels, Stephen A Weis, et al. Authenticating pervasive devices with human protocols. In *Crypto*, volume 3621, pages 293–308. Springer, 2005.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 78–95. Springer, 2005.

[KPV+17] Eike Kiltz, Krzysztof Pietrzak, Daniele Venturi, David Cash, and Abhishek Jain. Efficient authentication from hard learning problems. *Journal of Cryptology*, 30(4) :1238–1275, 2017.

[KRSS] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4 : Post-quantum crypto library for the ARM Cortex-M4. `https://github.com/mupq/pqm4`.

[KSN17] Aravind Karrothu, Research Scholar, and Jasmine Norman. An analysis of LPN based HB protocols. In *Advanced Computing (ICoAC), 2016 Eighth International Conference on*, pages 138–145. IEEE, 2017.

[KSS10] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and concurrent security of the HB and HB+ protocols. *Journal of cryptology*, 23(3) :402–421, 2010.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 372–389. Springer, 2008.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 636–652. Springer, 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *Theory of Cryptography Conference*, pages 293–310. Springer, 2011.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In *SCN*, volume 4116, pages 348–359. Springer, 2006.

[LM13] Vadim Lyubashevsky and Daniel Masny. Man-in-the-middle secure authentication schemes from LPN and weak PRFs. In *Advances in Cryptology–CRYPTO 2013*, pages 308–325. Springer, 2013.

[LMB+19] Jérôme Lablanche, Lina Mortajine, Othman Benchaalal, Pierre-Louis Cayrel, and Nadia El Mrabet. Optimized implementation of the NIST PQC submission ROLLO on microcontroller. *IACR Cryptol. ePrint Arch.*, 2019 :787, 2019.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *Public-Key Cryptography–PKC 2013*, pages 107–124. Springer, 2013.

[Loi06] Pierre Loidreau. Properties of codes in rank metric. *arXiv preprint cs/0610057*, 2006.

[Loi07] Pierre Loidreau. *Metrique rang et cryptographie*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2007.

[Lom11] Chris Lomont. Introduction to intel advanced vector extensions. *Intel white paper*, 23, 2011.

[MAB+17] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Gilles Zémor, and INSA-CVL Bourges. Ouroboros-R. 2017. `https://pqc-ouroborosr.org/`.

[MAB+18] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and INSA-CVL Bourges. HQC – Hamming quasi-cyclic. *NIST PQC Round*, 2 :4–13, 2018. `https://pqc-hqc.org/`.

[MAB+19] Carlos Aguilar Melchor, Nicolas Aragon, Magali Bardet, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Ayoub Otmani, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. ROLLO – Rank-Ouroboros, LAKE & LOCKER. 2019. `https://pqc-rollo.org`.

[MAB+20a] Carlos Aguilar Melchor, Nicolas Aragon, Magali Bardet, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Ayoub Otmani, Olivier Ruatta,

Jean-Pierre Tillich, and Gilles Zémor. ROLLO – Rank-Ouroboros, LAKE & LOCKER. 2020. https://pqc-rollo.org.

[MAB+20b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Maxime Bros, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Rank Quasi-Cyclic. 2020. https://pqc-rqc.org.

[Min14] Minunit, a minimal unit testing framework for C/C++, 2014. https://github.com/siu/minunit.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices : Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[MVZJ18] Vasileios Mavroeidis, Kamer Vishi, Mateusz D Zych, and Audun Jøsang. The impact of quantum computing on present cryptography. *arXiv preprint arXiv :1804.00200*, 2018.

[Nis16a] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. 2016.

[Nis16b] *NIST Post-Quantum Standardization Process*, 2016. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[Ope] OpenSSL. https://www.openssl.org/.

[Ore33] Oystein Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35(3) :559–584, 1933.

[Par] Particle. Particle photon. https://docs.particle.io/photon/.

[Per13] Edoardo Persichetti. Code-based public-key encryption resistant to key leakage. In *International Conference on Availability, Reliability, and Security*, pages 44–54. Springer, 2013.

[Pet83] Fabien Petitcolas. La cryptographie militaire, 1883.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

[Por16] Thomas Pornin. BearSSL : A smaller SSL/TLS library, 2016. `https://bearssl.org/`.

[PQC] PQClean : Clean, portable, tested implementations of post-quantum cryptography. `https://github.com/PQClean/PQClean`.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer, 1996.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6) :1–40, 2009.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[San10] Gérald Santucci. The internet of things : Between the revolution of the internet and the metamorphosis of objects. *Vision and Challenges for Realising the Internet of Things*, pages 11–24, 2010.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2) :303–332, 1999.

[Sim94] D. R. Simon. On the power of quantum computation. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 116–123, 1994.

[Ste96] Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6) :1757–1768, 1996.

[STMa] STMicroelectronics. NUCLEO-144. `https://www.st.com/en/evaluation-tools/nucleo-f429zi.html`.

[STMb] STMicroelectronics. STM32F407G. `https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.html`.

[STMc] STMicroelectronics. STM32F407G-DISC1. `https://www.st.com/en/evaluation-tools/stm32f4discovery.html`.

[STMd] STMicroelectronics. STM32F429ZI. `https://www.st.com/en/evaluation-tools/nucleo-f429zi.html`.

[Sup] SUPERCOP, measuring the performance of cryptographic software. https://bench.cr.yp.to/supercop.html.

[Vér97] Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1) :57–69, 1997.

[WP06] André Weimerskirch and Christof Paar. Generalizations of the Karatsuba algorithm for efficient implementations. *IACR Cryptology ePrint Archive*, 2006 :224, 2006.

[ZCH+19] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, and Oussama Danba. PQC round-2 candidate : NTRU. Technical report, Technical report, NIST, 2019.

[ZG17] Bin Zhang and Xinxin Gong. New algorithms for solving LPN. Cryptology ePrint Archive, Report 2017/780, 2017. https://eprint.iacr.org/2017/780.

[ZJW16] Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster algorithms for solving LPN. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 168–195. Springer, 2016.

[ZY17] Jiang Zhang and Yu Yu. Two-round PAKE from approximate SPH and instantiations from lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 37–67. Springer, 2017.

Résumé

La conception d'ordinateurs quantiques aura un impact majeur sur la cryptographie que nous utilisons tous les jours pour sécuriser nos communications, en particulier sur la cryptographie dite asymétrique. De plus, la multiplication des objets connectés dans notre quotidien et leurs limitations en terme de capacité de calcul et de mémoire disponible en font des cibles privilégiées. Cette thèse est à la croisée des chemins de ces deux problématiques, et est découpée en trois axes d'études indépendants.

Le premier porte sur la conception d'une primitive cryptographique appelée Hash Proof System (HPS). En nous basant sur le langage des chiffrés de RQC, nous allons montrer comment construire une telle primitive, puis nous en présenterons deux applications possibles : un schéma de chiffrement à témoin d'une part, puis un protocole d'échange de clés authentifié uniquement basé sur la connaissance d'un mot de passe commun, permettant ainsi de se passer d'une infrastructure à clé publique.

Le deuxième axe d'étude est l'introduction d'un nouveau problème cryptographique, le problème LRE (Learning Rank with Errors), adaptation du problème LPN à la métrique rang. Nous étudierons quelques-unes de ses propriétés puis en présenterons une première application possible, un protocole d'authentification symétrique nommé $\mathrm{HB}_{\mathsf{LRE}}$, adaptation du protocole HB au problème LRE.

Enfin, le troisième axe d'étude est l'adaptation et l'implémentation de plusieurs schémas cryptographiques soumis au NIST, à savoir RQC et ROLLO (qui regroupe les schémas LAKE, LOCKER et Ouroboros-R) sur microcontrôleurs. Nous montrerons les divers algorithmes qui ont été utilisés et en décrirons leur fonctionnement. Nous présenterons la libraire RBC, une librairie mise au point afin de faciliter l'utilisation et la conception de schémas basés sur la métrique rang qui intègrent les travaux d'implémentation réalisés durant cette thèse. Enfin, nous présenterons les performances de ces implémentations sur un microcontrôleur doté d'un processeur de type ARM Cortex-M4 et les mettrons en perspective avec les performances d'autres schémas cryptographiques post-quantiques soumis à l'appel à standardisation du NIST.

Mots clés : Codes Correcteurs, Post-quantique, Hash Proof System, RQC, ROLLO, Chiffrement à témoin, PAKE, HB, ARM Cortex-M4

Abstract

The design of a quantum computer would have a tremendous impact on the cryptography used nowadays, particularly in the asymetric cryptography domain. Moreover, the increasing use of connected objects in our everyday life and their limitations in term of computation capabilities and available memory make them a key target. This thesis was focused on those two problematics, and is divided into three independent axis.

The first one proposes the conception of a cryptographic primitive called Hash Proof System (HPS). Based on the language of RQC ciphertexts, we will show how to construct such a primitive and will present two applications of it : a witness encryption scheme in one hand and a password authenticated key exchange protocol on the other hand, yielding the use of a public key infrastructure unnecessary.

The second one introduces a new cryptographic problem called Learning Rank with Errors (LRE), an adaptation of the LPN problem to the rank metric setting. We will show some of its properties and describe a first application of it, a symmetric authentication scheme nammed $\mathrm{HB}_{\mathsf{LRE}}$, which is an adaptation of the HB protocol fitted to the LRE problem.

Finally, the third axis of study is the adaptation and the implementation of several cryptographic schemes submitted to the NIST, namely RQC and ROLLO (which is a merge of LAKE, LOCKER and Ouroboros-R) on microcontrollers. We will describe the different algorithms that we used and explain how they works. We will present the rbc-library, a library aiming to facilitate the use and conception of rank metric based schemes which make use of the implementations work realised during this thesis. Lastly, we will show the performances obtained by those implementations on a microcontroller equipped with an ARM Cortex-M4 processors and compare those results with other post-quantum cryptographic schemes submitted in the NIST standardisation process.

Keywords : Error-correcting codes, Post-quantum, Hash Proof System, RQC, ROLLO, Witness Encryption, PAKE, HB, ARM Cortex-M4