



UNIVERSITE GASTON DE BERGER DE SAINT LOUIS

UNITÉ DE RECHERCHE ET FORMATIONS DES SCIENCES APPLIQUÉES

DÉPARTEMENT DE MATHÉMATIQUES

MÉMOIRE DE MASTER 2

CRYPTOGRAPHIE POST-QUANTIQUE : Etude de quelques Algorithmes de Chiffrement basés sur les Codes Correcteurs d'Erreurs admis au concours de NIST 2021



SORY KONÉ
Etudiant en master 2
cryptographie, codes et
applications

Le président du Jury :
Examineur : Pr.
encadreur : Dr Bernadette
FAYE

Table des matières

1	La cryptographie de 1978 à Nos jours : De RSA à Mc Eliece	3
1.1	Arithmétique des codes correcteurs d'erreurs	5
1.1.1	Code correcteur d'erreur	5
1.1.2	Capacité de détection	6
1.1.3	Distance de Hamming	6
1.1.4	Codes linéaires	7
1.1.5	Codes de Reed-Solomon Généralisés	9
1.1.6	Codes de Goppa	12
1.1.7	Les Codes de Goppa binaires	16
1.1.8	Code Cyclique	21
1.1.9	Définition et description	21
1.1.10	Représentation polynomiale	21
1.1.11	Polynôme générateur	23
1.1.12	Codes quasi-cycliques	27
1.2	Les codes LDPC et MDPC	28
1.2.1	Définitions de quelques notions de sécurité	28
1.2.2	Les Problèmes de Base de sécurité	31
2	LE CONCOURS DE NIST	33
2.1	Motivation	34
2.2	Les Algorithmes de chiffrement	35
2.3	Classic McEliece	35
2.3.1	Paramètres du système	35

2.3.2	Génération de clé	36
2.3.3	Encodage	37
2.3.4	Décodage	37
2.3.5	Encapsulation	39
2.3.6	Décapsulation	40
2.3.7	Analyse de sécurité de classic McEliève	41
2.3.8	Les Attaques sur le système de McEliève classique	41
2.4	BIKE :Bit-flipping Key Encapsulation	42
2.4.1	Problème de Base de sécurité	43
2.4.2	Les Paramètres du système de BIKE	46
2.4.3	Génération de clé	47
2.4.4	Encodage	47
2.4.5	Désencapsulation	48
2.4.6	fonction de décodage de BIKE	49
2.4.7	Les Attaques sur le système de BIKE	51
2.4.8	Analyse de Sécurité	52
2.5	Hamming Quasi - Cyclic	52
2.5.1	Problèmes de Bases de Sécurité	55
2.5.2	Les Paramètres du Système	58
2.5.3	Génération de clés	59
2.5.4	Encryption / Encapsulation	59
2.5.5	Décryptage / Décapsulation	61
2.5.6	Analyse de Sécurité	61
2.5.7	Les attaques sur HQC	62
3	Les algorithmes d'attaques de Décodage par ensemble d'information	63
3.1	L'informatique quantique	65
3.1.1	Etats quantiques -Normes- Mesure -Les portes - circuits quantiques .	65
3.1.2	Les opérations sur les qubits	69
3.1.3	Les n-qubits	69

3.1.4	Oracle	70
3.1.5	Transformation quantique	72
3.1.6	L'algorithme de Grover	75
3.1.7	Transformation de Grover	78
3.1.8	Etapes de l'algorithme de Grover	81
3.2	Les outils mathématiques des ISD	85
3.2.1	Le coefficient binomial et la fonction d'entropie	85
3.2.2	Codes poinçonnés	85
3.2.3	Dénombrement	87
3.2.4	Distance de Gilbert-Varshamov	87
3.2.5	Combinatoire et probabilités	88
3.2.6	Un algorithme de jointure :Merge-Join	89
3.2.7	Théorie des graphes	90
3.2.8	Produits de graphe	92
3.3	Les Algorithmes quantiques	93
3.3.1	Algorithme de Grover	93
3.3.2	Marche aléatoire sur un graphe	95
3.3.3	Algorithme de Recherche de collisions	98
3.4	Décodage par ensemble d'information (ISD)	99
3.4.1	Algorithme de Prange	99
3.5	Squelette d'un algorithme ISD	102
3.5.1	Décodage par ensemble d'information quantique	104
3.6	Les algorithmes d'attaques ISD et ses dérivés	105
3.6.1	Les Algorithmes de Prange et de Bernstein	106
3.6.2	Les algorithmes ISD avancés avec technique de représentation	108
3.6.3	ISD avancé 2 avec technique de représentation étendue	118
3.6.4	Algorithme MMT* classique	118
3.6.5	Algorithme de $MMT^{\#}$	120
3.6.6	Les recherches de voisins proches : Algorithme de May-Ozerov	121
3.6.7	Version classique	123

4	Etudes comparatives	127
4.1	Classic McEliece	128
4.1.1	Avantages	128
4.1.2	Les inconvénients	128
4.1.3	Cryptanalyse du système	129
4.2	BIKE	129
4.2.1	Les avantages	129
4.2.2	Les inconvénients	130
4.2.3	Cryptanalyse du système	131
4.3	Hamming quasi-cyclic	132
4.3.1	Les avantages	133
4.3.2	Les inconvénients	133
5	Quelques implémentations	135
6	Conclusion et Perspectives	136

Liste des tableaux

2.1	Les paramètres de classic Mc Eliece	41
2.2	Tableau récapitulatif du taux d'échec de décodage de suivant le niveau de sécurité 1 et 3	52
2.3	Tableau des paramètres des codes de Reed-Solomon	55
2.4	Les paramètres des codes hqc pour les codes BCH tensoriels avec codes à répétition	58
2.5	Tableau récapitulatif des paramètres de hqc-RMS	59
3.1	Les portes de Pauli	67
3.2	Les porte de Hadamard	67
3.3	Les algorithmes ISD classiques	106
3.4	Les algorithmes ISD quantiques	106
4.1	Les meilleures attaques contre MDPC (or QC-MDPC) codes	132
4.2	Tableau comparatif des candidats sur le niveau de sécurité 5	134

Table des figures

3.1	<i>Recherche_{MMT}</i> - illustration sous forme d'arbre	
	Violet : recherche de collisions classiques	
	vert : technique des représentations	
	⌘ :Merge-Join	114
3.2	illustration sous forme d'arbre	126

Introduction Générale

Le codage de canal (numérique) transforme la séquence d'information utile en une séquence discrète codée nommée mot de code. En 1948, Shannon [**Shannon**] a démontré que lorsque le taux de transmission du système est inférieur à la capacité du canal de transmission, les erreurs causées par le bruit du canal peuvent être réduites à un niveau arbitrairement bas par l'utilisation d'un codage et d'un décodage approprié. À partir de ce moment-là, les chercheurs ont commencé à étudier différentes méthodes de construction des codes correcteurs d'erreurs

Un code correcteur d'erreurs est un système qui permet de détecter et/ou corriger les erreurs qui peuvent se produire lors de la transmission ou du stockage de données numériques. Voici un schéma général du fonctionnement d'un code correcteur d'erreurs.

Encodage : Les données d'entrée sont divisées en blocs de données. Un algorithme d'encodage transforme chaque bloc de données en un bloc de données plus long, en ajoutant des bits de redondance (appelés bits de parité) aux données d'origine. Ces bits supplémentaires sont calculés de manière à créer des caractéristiques spécifiques permettant de détecter et/ou corriger les erreurs.

Transmission ou stockage : Les blocs de données encodés sont transmis sur un canal de communication ou stockés dans un support de stockage.

Réception ou récupération : Les blocs de données encodés sont reçus ou récupérés depuis le support de stockage.

Détection d'erreurs : Lors de la réception, le décodeur vérifie les bits de redondance pour détecter si des erreurs se sont produites lors de la transmission ou du stockage. Il compare les bits de parité calculés à ceux reçus pour détecter les différences.

Correction d'erreurs (si possible) : Si des erreurs sont détectées et que le code correcteur d'erreurs est capable de corriger ces erreurs, l'algorithme de décodage peut utiliser les bits de redondance pour corriger les erreurs et récupérer les données d'origine.

Récupération des données d'origine : Après la détection et/ou la correction des erreurs, les données d'origine sont récupérées à partir des blocs de données encodés et des bits de redondance.

Il existe différents types de codes correcteurs d'erreurs, tels que les codes de Hamming, les codes cycliques, les codes de Reed-Solomon, les turbo codes, les codes LDPC et les codes polaires, qui utilisent diverses techniques pour détecter et corriger les erreurs.

Il est essentiel de noter que l'utilisation de codes correcteurs d'erreurs en combinaison avec le chiffrement ne doit pas être considérée comme une méthode de chiffrement à part entière, mais plutôt comme un moyen d'améliorer la fiabilité et l'intégrité des données chiffrées. Le

chiffrement assure la confidentialité, tandis que les codes correcteurs d'erreurs améliorent la robustesse des données face aux erreurs.

Le choix d'un code correcteur d'erreurs dépend des exigences spécifiques de l'application, telles que la quantité d'erreurs attendue et les contraintes de performance et de complexité. De ce fait, en 1978 Robert McEliece a développé un système de codage basé sur le code de Goppa Binaire mais à cause de la taille des clés publiques celui-ci a été abandonné en faveur du système de RSA.

Cependant, au bout de deux décennies les progrès récents de l'informatique quantique ont posé un sérieux problème pour la cryptographie traditionnelle à clé publique, dont la sécurité repose sur la dureté de la factorisation de grands entiers et de calcul des logarithmes discrets dans un groupe cyclique.

En fait l'algorithme de Shor [1] peut calculer la factorisation entière et les opérations logarithmiques discrètes en temps polynomial sur un ordinateur quantique, réduisant ainsi considérablement la marge de sécurité primitive actuelle de cryptographie à clé publique. Pour faire face à la menace de la machine quantique, les cryptanalystes se sont lancés dans l'élaboration de nouveaux cryptosystèmes qui ne seront basés sur le problème du logarithme discret et la durée de factorisation des grands nombres : C'est l'ère de la cryptographie post-quantique. La recherche de cryptosystèmes post-quantiques s'effectue dans les domaines des codes correcteurs d'erreurs, les courbes elliptiques, l'isogénie entre autres et plusieurs résultats s'en sont suivis. Ainsi le concours de NIST lancé en 2016 visant à sélectionner un ou plusieurs algorithmes de chiffrement post-quantique sûrs et efficaces, qui seraient largement acceptés et implémentés dans les systèmes d'information à l'avenir. Celui-ci est à son quatrième tour de 2017 à 2021 à l'issue duquel il a été retenu que 3 chiffrements dans le domaine des codes correcteurs d'erreurs que sont : le McEliece classique, le BIKE (Bit-flipping Key Encapsulation) et le HQC (Hamming Quasi Cyclic).

Au cours de notre rédaction nous aborderons dans le chapitre I les notions mathématiques sur lesquelles se sont fondées les candidats et leur base de sécurité.

Dans le chapitre II, nous parlerons de la cryptographie de 1978 à nos jours en relevant les travaux et les améliorations effectués dans le domaine des codes correcteurs d'erreurs.

Dans le chapitre III nous évoquerons NIST et son concours à savoir son objectif et les critères de sélection des candidats en faisant leur cryptanalyse détaillée.

Dans le chapitre IV nous étudierons les algorithmes d'attaques sur les candidats en mettant en exergue les structures mathématiques et informatiques sur lesquels ils se fondent.

Dans le dernier chapitre nous allons faire une étude comparative des avantages et inconvénients des candidats et donner une implémentation d'un candidat.

Chapitre 1

La cryptographie de 1978 à Nos jours : De RSA à Mc Elièce

Contents

1.1	Arithmétique des codes correcteurs d'erreurs	5
1.1.1	Code correcteur d'erreur	5
1.1.2	Capacité de détection	6
1.1.3	Distance de Hamming	6
1.1.4	Codes linéaires	7
1.1.5	Codes de Reed-Solomon Généralisés	9
1.1.6	Codes de Goppa	12
1.1.7	Les Codes de Goppa binaires	16
1.1.8	Code Cyclique	21
1.1.9	Définition et description	21
1.1.10	Représentation polynomiale	21
1.1.11	Polynôme générateur	23
1.1.12	Codes quasi-cycliques	27
1.2	Les codes LDPC et MDPC	28
1.2.1	Définitions de quelques notions de sécurité	28
1.2.2	Les Problèmes de Base de sécurité	31

En 1976, Diffie et Hellman ont publié l'idée de la clé publique ou de la cryptographie asymétrique. Jusqu'à cette époque, la cryptographie était basée sur l'idée de la cryptographie symétrique dans laquelle l'expéditeur et le destinataire devaient avoir une clé secrète partagée pour pouvoir communiquer en toute sécurité. La cryptographie asymétrique permet à un expéditeur de chiffrer un message avec une clé publique publiée qui ne peut être déchiffrée que par quelqu'un qui avait une clé privée associée. Les premiers schémas basés sur ce message étaient RSA et McEliece tous deux publiés en 1978. RSA a basé sa sécurité sur la dureté de factoriser de grands nombres et McEliece a basé sa sécurité sur le problème du décodage un code Goppa aléatoire. RSA a été préféré à McEliece comme l'un des schémas de cryptographie asymétrique les plus populaires des 40 dernières années, principalement en raison de la petite taille de sa clé publique par comparaison.

En 1994, Shor a proposé quelques algorithmes pour les ordinateurs quantiques qui a rendu les problèmes auparavant irréalisables sur le plan informatique de la factorisation de grands nombres et trouver des logarithmes discrets de nombres dans des anneaux modulaires est soudainement faisable.

Dans les décennies suivantes, cela n'a pas eu beaucoup d'effet sur la popularité du RSA en tant que norme de cryptage asymétrique. Mais, maintenant avec l'avènement des ordinateurs quantiques de tailles de bits croissantes, ce problème est revenu. Dans le même temps, les grandes tailles de clé requis par les systèmes basés sur des codes ne sont plus autant un problème en raison de l'augmentation accès aux réseaux à haut débit. Nous avons plusieurs classes de codes :

Les codes polaires sont une classe de codes correcteurs d'erreurs qui ont été introduits par Erdal Arikan en 2009. Ils se sont avérés être une avancée significative dans le domaine des codes correcteurs d'erreurs et ont attiré beaucoup d'attention depuis leur création. Ces codes sont devenus un élément clé dans les normes de communication numérique avancées telles que le 5G.

Turbo codes : Les turbo codes, introduits par C. Berrou, A. Glavieux et P. Thitimajshima en 1993, ont révolutionné les performances des codes correcteurs d'erreurs. Ils sont basés sur l'utilisation de deux ou plusieurs codes correcteurs d'erreurs en parallèle, combinés avec un schéma d'itération itératif pour améliorer considérablement les performances de décodage.

Codes LDPC : Les codes LDPC (Low-Density Parity-Check), proposés à l'origine par Robert Gallager dans les années 1960, ont été redécouverts dans les années 1990 par David MacKay. Ces codes utilisent des matrices creuses et présentent des performances de décodage proches des capacités théoriques des canaux de communication.

Codes polaires : Les codes polaires, introduits par Erdal Arikan en 2009, ont été un autre développement majeur dans les codes correcteurs d'erreurs. Comme mentionné précédem-

ment, ils permettent d'atteindre la limite de la capacité de Shannon de manière asymptotique et ont été adoptés dans les normes de communication 5G.

Codes Turbo-product : Les codes Turbo-product sont une extension des turbo codes et des codes LDPC, proposée par Claude Berrou en 2006. Ils permettent d'obtenir de très bonnes performances en utilisant des décodeurs simplifiés.

Codes polynomiaux : Les codes polynomiaux, introduits en 2007 par O. Ytrehus, sont une autre classe de codes correcteurs d'erreurs qui ont des performances prometteuses dans certains scénarios.

Ces améliorations ont contribué à faire évoluer les codes correcteurs d'erreurs vers des performances de plus en plus élevées, tout en réduisant la complexité des décodeurs. Ils sont largement utilisés dans une multitude d'applications, allant des communications numériques (télécommunications, Internet) aux systèmes de stockage de données (disques durs, mémoires flash) en passant par les communications spatiales et les systèmes embarqués.

En 2016, le National Institute of Standards and Technology (NIST) a publié une appel à propositions pour des cryptosystèmes post-quantiques qui ne seraient pas basés sur la difficulté à factoriser de grands nombres ou à trouver des logarithmes discrets .

Dans ce qui suit, nous rappelons quelques notions de bases sur les codes.

1.1 Arithmétique des codes correcteurs d'erreurs

1.1.1 Code correcteur d'erreur

Un code correcteur d'erreur est une technique de codage basée sur la redondance. Elle est destinée à corriger les erreurs de transmission d'une information sur un canal de communication peu fiable.

Exemple :

1. 0101 \rightarrow 01010101 code à répétition
2. Alpha Tango Charlie \rightarrow ATC
3. TCP-IP (Internet)

Codage avec redondance

- Soit un message initial $u = (u_1, u_2, u_3, \dots, u_k)$
- Une fonction $\varphi(u) = w = (w_1, w_2, \dots, w_r)$
- message reçu $(u, w) = (u_1, u_2, \dots, u_k, w_1, w_2, \dots, w_r)$

w : la redondance

r : la longueur de w

(w_1, w_2, \dots, w_r) : Les bits de contrôle

Corriger : Il faut que φ^{-1} existe $\rightarrow \varphi$ est injectif

Chaque mot de code est de longueur $n = k + r$. L'ensemble de tous les mots obtenus de cette façon forme un **code** de **longueur** n et de **dimension** k .

Notation : φ est la fonction de codage

$Im\varphi$ est le code

- Détecter une erreur :

$$(u, w) = (u_1, u_2, \dots, u_k, w_1, w_2, \dots, w_r)$$

$$S(u) = \varphi(u_1, u_2, \dots, u_k) - w_1, w_2, \dots, w_r = 0$$

$$\varphi(u_1, u_2, \dots, u_k) \neq (w_1, w_2, \dots, w_r)$$

Calcul du Syndrôme

$$\varphi(u_1, u_2, \dots, u_k) = (y_1, y_2, \dots, y_r)$$

$$S(u) = (y_1 - w_1, y_2 - w_2, \dots, y_r - w_r)$$

— Si $S(u) \neq 0 \rightarrow$ Il y'a erreurs.

— $S(u) = 0 \rightarrow$ Il n'y a pas d'erreurs

1.1.2 Capacité de détection

Un code C de longueur n est dit **t-détection (e-correction)** s'il permet de détecter **t erreurs** (resp. **e-erreurs**).

1.1.3 Distance de Hamming

Soit F un alphabet fini, pour deux éléments de F on appelle distance de Hamming de X et Y le nombre de composantes pour lesquelles X et Y diffèrent :

Elle est notée $d(X, Y) = \text{card}\{i \in \{1, \dots, n\} / x_i \neq y_i\}$.

Exemple : $F = \{0, 1, 2, 3, 4, 5\}$ et $n = 5$

Soient $X = (0, 1, 2, 3, 1)$ et $Y = (0, 2, 1, 3, 1)$, la distance de Hamming de X et Y est $d(X, Y) = 2$.

L'application $d : F^n \times F^n \rightarrow \mathbb{N}$

$$(x, y) \rightarrow d(x, y)$$

d est une distance

1. $d(x, y) \geq 0$
2. $d(x, y) = d(y, x)$
3. $d(x, z) \leq d(x, y) + d(y, z)$
4. si $d(x, y) = 0 \rightarrow x = y$

Poids de Hamming

On appelle le poids de Hamming noté $\omega(m)$, le nombre de bits non nuls de m .

Exemple : $\omega(0, 1, 1) = 2$

Proposition 1 Soit C un code longueur n est C est t -correcteurs s'il peut corriger t -erreurs $\forall x \in \mathbb{F}^n, \text{card}\{c \in C / d(x, c) \leq t\} \leq 1$.

Remarque 1 Pour pouvoir corriger, il faut que c soit l'unique mot de code le plus proche de x .

La distance minimale d'un code

C'est la plus petite distance entre les mots de C .

On le note : $d_H(C) = \min\{d(x, y) / x, y \in C^2, x \neq y\}$.

Notation : Si C est de cardinal m , de longueur n et distance minimale d , on dit que C est un (m, n, d) -code.

1.1.4 Codes linéaires

Un code linéaire est caractérisé par :

- Matrice génératrice, Matrice de contrôle
- Code systématique
- décodage d'un code linéaire
- Syndrome

Exemple

1. Code de Hamming
2. Codes cycliques
3. Codes BCH
4. de Reed Salomon

Un code linéaire

Définition 1 Un code linéaire de longueur n est un sous espace vectoriel $C \in \mathbb{F}_2^k$. La lettre k désigne la dimension, le nombre de mot de code est 2^k . le poids $\omega(x) = \{i / x_i \neq 0\}$.

Notation : \mathbf{C} est un code (n,k,d) .

Si $\mathbb{F} = \mathbb{F}_2 = \{0, 1\}$, $\forall n > 0$, si l'on pose $m = 2^n$, l'application

$$\begin{aligned}\varphi : \mathbb{F}^n &\rightarrow \mathbb{Z}/m\mathbb{Z} \\ (c_0, c_1, \dots, c_n) &\rightarrow c_0 + 2c_1 + \dots + 2^{n-1}c_{n-1}\end{aligned}$$

est une bijection .

Définition 2 *Un code de longueur n et de dimension k est sous espace vectoriel de dimension k de \mathbb{F}_q^n .*

Proposition 2 *Si \mathbf{C} est un code linéaire, l'ensemble des distances entre les mots de code de \mathbf{C} est l'ensemble des poids des mots de \mathbf{C} .*

Preuve :

Soient $x, y \in \mathbf{C}$ alors $x - y \in \mathbf{C}$ car \mathbf{C} est un sous espace vectoriel.

$$d(x, y) = \omega(x - y).$$

Donc toute distance est un poids d'un mot de \mathbf{C} .

Réciproquement si x est un mot de \mathbf{C} , $\omega(x) = d(x, 0)$.

Donc tout poids est une distance de mots de \mathbf{C} .

Conséquence : La distance **minimale** de \mathbf{C} est égale au plus petit poids des mots de \mathbf{C} .

Remarque :

1. Cette conséquence nous donne une amélioration dans la recherche de la distance minimale.
2. Un code linéaire est entièrement déterminé par une base (k -mot de code)

Ceci nous conduit à la notion de matrice génératrice.

Une matrice génératrice

Une matrice génératrice G d'un code (n, k) linéaire, la matrice $k \times n$ de rang k dont les lignes forment une base de \mathbf{C} .

Alors on a : $\mathbf{C} = m \times \mathbf{G}$, $m \in \mathbb{F}_2^k$.

L'application $m \rightarrow mG$ est un isomorphisme de \mathbb{F}_2^k sur \mathbf{C} .

La matrice de contrôle de Parité

Définition 3 La matrice de contrôle peut être vue comme la matrice génératrice du code dual

$$C^\perp = \{y \in \mathbb{F}_2^n, \forall c \in C, y \cdot c = 0\}$$

où \cdot est le produit scalaire usuel.

Proposition 3 Soit \mathbf{H} une matrice de contrôle du code C . La distance minimale d de C est caractérisée par les propriétés suivantes :

- $d - 1$ colonnes de \mathbf{H} sont toujours linéairement indépendantes.
- il y a un système de d colonnes de \mathbf{H} qui est lié.

Preuve

Supposons que $d - 1$ colonnes de \mathbf{H} sont toujours linéairement indépendantes. Soit $c = (c_1, \dots, c_n)$ un mot de code. On a $\mathbf{H}^t c = 0$. Si $w(c) < d$ alors on a relation entre moins de d colonnes de \mathbf{H} . Inversement un mot c de poids d donne une relation entre d colonnes de \mathbf{H} .

1.1.5 Codes de Reed-Solomon Généralisés

Définition 4 (Codes de Reed-Solomon Généralisés)

Soient un entier $m \geq 1$, n un entier et q une puissance d'un nombre premier tels que $n \leq q^m$.

Soient $x = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ tels que $x_i \neq x_j, i, j = 1, \dots, n$ si $i \neq j$ et $v = (v_1, \dots, v_n) \in \mathbb{F}_{q^m}^n$ (chaque v_i est non nul). Le code de Reed-Solomon Généralisé de longueur n , de dimension k , de support x et de multiplieur v , noté $\text{GRS}_k(x, v)$ est :

$$\text{GRS}_k(x, v) = \{(v_1 f(x_1), \dots, v_n f(x_n)) : f \in \mathbb{F}_{q^m}[x] : \deg(f) < k\}$$

Proposition 4 Soit un code de Reed-Solomon généralisé $\text{GRS}_k(x, v)$ de matrice de contrôle \mathbf{H} . Alors la matrice \mathbf{H} est sous la forme :

$$\mathbf{H} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1 x_1 & v_2 x_2 & \cdots & v_n x_n \\ v_1 x_1^2 & v_2 x_2^2 & \cdots & v_n x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_1 x_1^{k-1} & v_2 x_2^{k-1} & \cdots & v_n x_n^{k-1} \end{pmatrix}$$

Une matrice de contrôle d'un code de Reed-Solomon généralisé est le produit d'une matrice de Vandermonde et d'une matrice diagonale.

Soit \mathbf{H} une matrice de parité d'un code $\text{GRS}_k(x, v)$.

On a : $\mathbf{H} = \mathbf{V}_{n-k,n}^{(x)} \cdot \mathbf{D}_v$ où $\mathbf{V}_{n-k,n}^{(x)}$ est une matrice de Vandermonde et $\mathbf{D}_v = \text{diag}(v_1, \dots, v_n)$ est une matrice diagonale avec

$$\mathbf{V}_{n-k,n}^{(x)} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{k-1} & x_2^{k-1} & \dots & x_n^{k-1} \end{pmatrix} \quad \text{et} \quad \mathbf{D}_v = \begin{pmatrix} v_1 & 0 & \dots & 0 \\ 0 & v_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & v_n \end{pmatrix}.$$

Tout code linéaire C satisfait à la borne de Singleton. Autrement dit, si C est un $[n, k, d]$ -code linéaire, alors $d \leq n - k + 1$. En particulier, un code de Reed-Solomon généralisé vérifie aussi cette condition. La proposition suivante énonce une propriété particulière qu'ont les codes GRS : le critère d'optimalité MDS.

Proposition 5 Un code de Reed-Solomon généralisé est MDS (Maximum Distance Separable)[Ndollane].

Preuve

Soit C un code $\text{GRS}_k(x, v)$ sur \mathbb{F}_q de paramètres $[n, k, d]$. Alors $d \leq n - k + 1$ (Borne de Singleton).

Soit $c \in C$, alors $\exists f \in \mathbb{F}_q[x]$ avec $\deg(f) \leq k - 1$ tel que $c = (v_1 \cdot f(x_1), v_2 \cdot f(x_2), \dots, v_n \cdot f(x_n))$. Un polynôme de degré inférieur à k a au plus $k - 1$ racines. Donc c a au moins $n - k + 1$ composantes non nulles. Ainsi le poids de Hamming de c est supérieur ou égal à $n - k + 1$ ($w_H(c) \geq n - k + 1$.)

D'où le poids minimal du code C est supérieur ou égal à $n - k + 1$. Étant donné que le poids minimal d'un code linéaire est égal à sa distance minimale, alors $d \geq n - k + 1$. Par suite, $d = n - k + 1$.

Exemple : Considérons le corps $\mathbb{F}_8 = \mathbb{F}_2[x]/(x^3 + x + 1)$. Soit α un élément primitif de \mathbb{F}_8 .

Le code

$$C = \left\{ \left(\alpha^3 \cdot f(\alpha^5), f(\alpha^2), \alpha \cdot f(1), \alpha \cdot f(\alpha^4), f(\alpha^3), \alpha^2 \cdot f(0), \alpha^5 \cdot f(\alpha^6), f(\alpha) \right) \mid f \in \mathbb{F}_8[x] : \deg(f) < 3 \right\}$$

est un code de Reed-Solomon généralisé sur \mathbb{F}_8 de paramètres $[8, 3, 6]$,

de support $x = (\alpha^5, \alpha^2, 1, \alpha^4, \alpha^3, 0, \alpha^6, \alpha)$ et de multiplieur $v = (\alpha^3, 1, \alpha, \alpha, 1, \alpha^2, \alpha^5, 1)$. Une matrice génératrice \mathbf{H} du code $\text{GRS}_3(x, v)$ est :

$$\begin{pmatrix} \alpha^3 & 1 & \alpha & \alpha & 1 & \alpha^2 & \alpha^5 & 1 \\ \alpha & \alpha^2 & \alpha & \alpha^5 & \alpha^3 & 0 & \alpha^4 & \alpha \\ \alpha^6 & \alpha^4 & \alpha & \alpha^2 & \alpha^6 & 0 & \alpha^3 & \alpha^2 \end{pmatrix}.$$

Définition 5 (Polynôme de Lagrange)

Soient $(\alpha_0, \dots, \alpha_{n-1}) \in \mathbb{F}_q^n$ avec $\alpha_i \neq \alpha_j$ si $i \neq j$. Définissons les polynômes de Lagrange par :

$$\forall i \in [0, n-1] \mathbf{L}_i[x] = \prod_{j \in [0, n-1], j \neq i} (x - \alpha_j) \in \mathbb{F}_q[x]_n.$$

En plus d'être des codes MDS, les codes GRS ont encore une autre propriété : le dual d'un code GRS est aussi un code GRS de même support (mais pas de même multiplieur).

Propriété :

Le dual d'un code $GRS_k(\alpha, v)$ est le code $GRS_{n-k}(\alpha, v')^\perp$ où $v' \in \mathbb{F}_q^{*n}$ tel que

$$\forall i \in [0, n-1], v'_i = v_i^{-1} \mathbf{L}(\alpha_i)^{-1}$$

Remarque :

Si $n = q - 1$, on dit que le code de Reed-Solomon généralisé est primitif. Si $v_i = 1 \forall i \in \{1, \dots, n\}$, on dit que le code est normalisé. On obtient ainsi un code de Reed-Solomon tout simplement.

Donnons la définition d'une fonction d'évaluation. Soit $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$ deux à deux distincts, avec $n < q$.

Définition 6 La fonction d'évaluation associée à $\alpha_1, \dots, \alpha_n$ est :

$$\begin{aligned} ev : \mathbb{F}_q[x] &\longrightarrow \mathbb{F}_q^n \\ f &\longrightarrow (f(\alpha_1), \dots, f(\alpha_n)). \end{aligned}$$

Code de Reed-Solomon**Définition 7 [Ndollane]p44**

Soient $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$, avec $n < q$ et $\alpha_i \neq \alpha_j$ si $i \neq j$, et ev la fonction d'évaluation associée à $\alpha_1, \dots, \alpha_n$. Soit

$$\mathbf{L}_k = \{f \in \mathbb{F}_q[x] : \deg f < k\},$$

l'ensemble des polynômes de $\mathbb{F}_q[x]$ de degré strictement inférieur à k .

Le code de Reed-Solomon de dimension k et de support $\alpha = (\alpha_1, \dots, \alpha_n)$, noté $RS(\alpha, k)$, est

$$RS(\alpha, k) = C = ev(\mathbf{L}_k)$$

Les codes de Reed-Solomon sont des codes d'évaluation. Les mots de code d'un code de Reed-Solomon sont des vecteurs d'évaluation de polynômes de degré inférieur à la dimension du code. D'après la propriété sur le dual des GRS, on peut définir un code GRS comme étant un dual d'un code d'évaluation. Cela permet de construire deux familles d'algorithmes de décodage des codes RS :

1. décodage par interpolation, pour un code d'évaluation ;
2. décodage par syndrome, pour un code dual de code d'évaluation.

Exemple :

Considérons le corps $\mathbb{F}_8 = \mathbb{F}_2[x]/(x^3 + x + 1)$. Soit α un élément primitif de \mathbb{F}_8 .

Le code

$$C = \{(f(\alpha), f(1), f(\alpha^2), f(0)) \mid f \in \mathbb{F}_8[x] : \deg f < 2\}$$

est un code de Reed-Solomon de longueur 4, de dimension 2 et de support $x = (\alpha, 1, \alpha^2, 0)$. Une matrice génératrice \mathbf{G} de C est

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \alpha & 1 & \alpha^2 & 0 \end{pmatrix}$$

1.1.6 Codes de Goppa

Définition 8 [Ndollane] p45

Soient $m, n > 1, g(x) \in \mathbb{F}_{q^m}[x]$ un polynôme de degré $t = n - k$ et $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_n) \in \mathbb{F}_{q^m}^n$, tel que $\forall i = 1, \dots, n \quad \gamma_i \neq 0$.

Le code de Goppa de support γ et de polynôme générateur g est défini par :

$$\Gamma(\gamma, g) = \left\{ (c_1, c_2, \dots, c_n) \in \mathbb{F}_{q^m}^n \mid \sum_{i=1}^n \frac{c_i}{z - \gamma_i} \mod g(z) = 0 \right\}.$$

Remarque : Si le polynôme g est irréductible, on dit que $\Gamma(\gamma, g)$ est un code de Goppa irréductible. On dit que le polynôme de Goppa g est séparable s'il n'admet que de racines simples.

Le polynôme $S_y(z) = \sum_{i=1}^n \frac{c_i}{z - \gamma_i} \mod g(z)$ est appelée polynôme syndrome du mot $y \in \mathbb{F}_{q^m}^n$.

Les codes de Goppa admettent des matrices de parité sous forme de Cauchy sous certaines conditions.

Le théorème suivant nous donne ce résultat.

Théorème 1 [Ndollane]p45

Soit le polynôme $g = (x - z_1)(x - z_2) \cdots (x - z_t)$ avec $z_i \in \mathbb{F}_q^m$ et deux à deux distincts.

Si g est le polynôme de Goppa du code de Goppa $\Gamma(\gamma, g)$ où $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_n)$ alors la matrice de parité de ce code est égale à la matrice de Cauchy $C(z, \gamma)$ associée à γ et z où $z = (z_1, z_2, \dots, z_t)$.

Preuve :

Pour tout $c \in \mathbb{F}_q^n$, on a :

$$\begin{aligned}
 c \in \Gamma(\gamma, g) &\iff \sum_{i=1}^n \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{g(x)} \\
 &\iff \sum_{i=1}^n \frac{c_i}{x - \gamma_i} \equiv 0 \pmod{x - z_j}, \forall j = 1, \dots, t \\
 &\iff \sum_{i=1}^n \frac{c_i}{x - \gamma_i} = 0, \forall j = 1, \dots, t \\
 &\iff \sum_{i=1}^n \frac{c_i}{z_j - \gamma_i} = 0, \forall j = 1, \dots, t \\
 &\iff C(z, \gamma)^t c = 0
 \end{aligned}$$

Alors la matrice de Cauchy $C(z, \gamma)$ est une matrice de parité du code de Goppa $\Gamma(\gamma, g)$.

Matrice de parité d'un code de Goppa

Soit $\Gamma(\gamma, g)$ un code de Goppa. Une matrice de parité d'un tel code est obtenue à partir de la matrice \mathbf{H}_0 donnée par [Ndollane]p46 :

$$\mathbf{H}_0 = \begin{pmatrix} \frac{1}{g(\gamma_1)} & \cdots & \frac{1}{g(\gamma_n)} \\ \vdots & & \vdots \\ \frac{\gamma_1^{t-1}}{g(\gamma_1)} & \cdots & \frac{\gamma_n^{t-1}}{g(\gamma_n)} \end{pmatrix}$$

Le code de Goppa $\Gamma(\gamma, g)$ est un sous-code sur le sous corps \mathbb{F}_q (le sous-code qui contient tous les mots de code dont les composantes sont dans \mathbb{F}_q) d'un code de Goppa particulier sur le corps \mathbb{F}_{q^m} à q^m éléments.

Pour obtenir une matrice de parité \mathbf{H} du code $\Gamma(\gamma, g)$, on choisit une base de \mathbb{F}_{q^m} sur \mathbb{F}_q . Une telle base est de cardinal m .

Chaque élément de la matrice \mathbf{H}_0 est ensuite décomposé en un vecteur colonne de m éléments de \mathbb{F}_q (en fonction des m éléments de la base). On passe ainsi d'une matrice de taille $t \times n$ sur \mathbb{F}_{q^m} à une nouvelle matrice \mathbf{H} de taille $mt \times n$ sur \mathbb{F}_q .

Pour un cryptosystème basé sur les codes de Goppa, les paramètres du système sont (m, t) et sont connus de tous, la clef privée est (γ, g) et la clef publique est une matrice de parité

$\mathbf{H} \in mt \times n$.

Exemple:[Ndollane]p47 Soit $\Gamma(\gamma, g)$ le code de Goppa tel que $\gamma = (0, 1, w, w^2, w^3, w^4, w^5, w^6)$ où w est un élément primitif de \mathbb{F}_8 et $g(z) = z^2 + z + 1 \in \mathbb{F}_8[z]$. Dans cet exemple, on a $q = 2$.

Les paramètres de ce code sont alors $m = 3, t = 2$.

$\mathbb{F}_8 = \mathbb{F}_2[z]/f(z)$ où $f(z) = z^3 + z + 1$ est un polynôme unitaire irréductible sur \mathbb{F}_2 .

Prenons $B = (1, w, w^2)$ comme base de \mathbb{F}_8 sur \mathbb{F}_2 . On a :

$$\mathbf{H}_0 = \begin{pmatrix} \frac{1}{g(0)} & \frac{1}{g(1)} & \cdots & \frac{1}{g(w^6)} \\ \frac{0}{g(0)} & \frac{1}{g(1)} & \cdots & \frac{w^6}{g(w^6)} \end{pmatrix}$$

L'élément w étant racine du polynôme $f(z)$, on a alors

$$w^3 + w + 1 = 0, \text{ i.e } w^3 = w + 1.$$

Il s'ensuit que $w^4 = w^2 + w$, $w^5 = w^2 + w + 1$, $w^6 = w^2 + 1$

$\frac{1}{g(0)} = 1 \equiv (100)$ dans la base B , $\frac{1}{g(1)} = 1 \equiv (100)$ dans la base B .

$$\begin{aligned}
\frac{1}{g(w)} &= \frac{1}{w^2+w+1} \\
&= \frac{1}{w^3+w^2} \\
&= \frac{1}{w^2(w+1)} \\
&= \frac{1}{w^5} \\
&= w^2 \equiv (001) \text{ dans la base } B \\
\frac{1}{g(w^2)} &= \frac{1}{w^4+w^2+1} \\
&= \frac{1}{w+1} \\
&= \frac{1}{w^3} \\
&= w^4 \\
&= w^2 + w \equiv (011) \text{ dans la base } B \\
\frac{1}{g(w^3)} &= \frac{1}{w^6+w^3+1} \\
&= \frac{1}{w^2+w+1} \\
&= \frac{1}{w^2+w^3} \\
&= \frac{1}{w^5} = w^2 \equiv (001) \text{ dans la base } B \\
\frac{1}{g(w^4)} &= \frac{1}{g(w^2+w)} \\
&= \frac{1}{w^4+w^2+w^2+w+1} \\
&= \frac{1}{w^4+w^3} \\
&= \frac{1}{w^6} \\
&= w \equiv (010) \text{ dans la base } B \\
\frac{1}{g(w^5)} &= \frac{1}{g(w^2+w+1)} \\
&= \frac{1}{w^4+w+1} \\
&= \frac{1}{w^6} \\
&= w \equiv (010) \text{ dans la base } B \\
\frac{1}{g(w^6)} &= \frac{1}{g(w^2+1)} \\
&= \frac{1}{w^4+1+w^2+1+1} \\
&= \frac{1}{w^3} \\
&= w^4 \\
&= w^2 + w \equiv (011) \text{ dans la base } B
\end{aligned}$$

On a donc $H_o = \begin{pmatrix} 1 & 1 & w^2 & w^2+w & w^2 & w & w & w^2+w \\ 0 & 1 & w+1 & w^2+1 & w^2+w+1 & w^2+w+1 & w^2+1 & w+1 \end{pmatrix}$

Pour avoir la matrice de parité \mathbf{H} du code, on remplace chaque élément de la matrice \mathbf{H}_o par le vecteur colonne correspondant à ses composantes dans la base B .

Par suite

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

1.1.7 Les Codes de Goppa binaires

Théorème 2 [Ndollane]p46

Soit $\Gamma(\gamma, g)$ un code de Goppa binaire de polynôme g irréductible de degré t .

Alors la distance minimale d du code vérifie $d \geq 2t + 1$.

Preuve : Soient $c \in \Gamma(\gamma, g)$ et $T_c = \{i : c_i \neq 0\}$ l'ensemble des entiers qui indexent les composantes non nulles du mot de code c .

Posons $\sigma_c(z) = \prod_{i \in T_c} (x - \gamma_i)$, $S_{y(z)} = \sum_{i=1}^n \frac{y_i}{z - \gamma_i} \mod g(z)$ le polynôme syndrôme d'un mot y . Le poids du mot c est égal au cardinal de l'ensemble T_c et donc au degré du polynôme σ_c . Evaluons le degré de σ_c .

Posons $w = \deg \sigma_c$. La dérivée de σ_c est :

$$\begin{aligned} \sigma'_c(z) &= \sum_{j \in T_c} \prod_{i \in T_c / j} (z - \gamma_i) \\ &= \sum_{j \in T_c} \frac{1}{z - \gamma_j} \prod_{i \in T_c} (z - \gamma_i) \\ &= \sum_{j \in T_c} \frac{c_j}{z - \gamma_j} \sigma_c(z) \end{aligned}$$

On a donc

$$\begin{aligned} \sigma'_c(z) \mod g(z) &= \sigma_c(z) \sum_{j \in T_c} \frac{c_j}{z - \gamma_j} \mod g(z) \\ &= S_c(z) \sigma_c(z) \\ &= 0 \text{ car } c \text{ est un mot de code} \end{aligned}$$

Alors $\frac{\sigma'_c(z)}{\sigma_c(z)} \mod g(z) = S_c(z) = 0 \implies \sigma'_c(z) = 0 \mod g(z)$

De plus la dérivée d'un polynôme défini sur un corps fini de caractéristique 2 ne contient que des puissances paires.

Il vient alors que $\sigma'_c(z) = a_p z^{2p} + \dots + a_2 z^2 + a_0$, ce qui équivaut à $\sigma'_c(z) = (a_p z^{2p} + \dots + a_2 z^2 + a_0)^2$ car on a $a_i^2 = a_i$ pour tout $a_i \in \{0, 1\}$ dans un tel corps.

Donc $\deg \sigma'_c \geq 2p$, et puisque $w - 1 \geq \deg \sigma'_c$, alors $w - 1 \geq 2p$.

La relation $\sigma'_c(z) = 0 \mod g(z)$ implique que $g(z)$ divise $f(z)^2$

où $f(z) = a_{2p} z^{2p} + \dots + a_2 z^2 + a_0$ avec $\deg(f) = p$. Puisque le polynôme g est irréductible, alors g divise f et donc $t \leq p$.

Par suite $t \leq p$ et $2p \leq w - 1 \implies 2t \leq 2p \leq w - 1$, i.e $w \geq 2t + 1$.

On a ainsi $w_H(c) \geq 2t + 1$.

D'où $d \geq 2t + 1$.

Remarque :

Le théorème précédent montre que l'algorithme de décodage d'un code de Goppa binaire irréductible peut décoder jusqu'à t erreurs avec t le degré du polynôme générateur du code.

Décodage des codes de Goppa binaires

Il existe de nombreuses techniques pour décoder les codes de Goppa. Mais toutes ces techniques fonctionnent sur le même principe : étant donné un mot $y \in \mathbb{F}_{2^m}^n$, $y = c + e$ où c c'est un mot de code de Goppa et e un vecteur d'erreur de poids au plus égal à t , décoder le mot y revient à :

- calculer son syndrome ;
- déterminer à partir de ce syndrome ce que l'on appelle l'équation clef
- résoudre l'équation clef pour retrouver le vecteur d'erreur e .

L'algorithme de Patterson

C'est le plus ancien algorithme pour décoder les codes de Goppa. Il fut mis au point en 1975 par N. J. Patterson. Cet algorithme n'utilise que le support γ et le polynôme g du code de Goppa $\Gamma(\gamma, g)$ et n'a pas besoin de la matrice de parité pour calculer le syndrome d'un mot donné.[Ndollane]p50

Polynôme Syndrome

Pour tout mot $y = c + e \in \mathbb{F}_{2^m}^n$, le syndrome $S_y(z)$ de y est un polynôme sur \mathbb{F}_{2^m} modulo $g(z)$. Il est donné par :

$$S_{y(z)} = \sum_{i=1}^n \frac{y_i}{z - \gamma_i} \mod g(z).$$

Remarque :

Sachant que c est un mot de code, alors son syndrome est nul et donc y et e ont même syndrome, i.e. $S_y(z) = S_c(z) + S_e(z) = 0 + S_e(z) = S_e(z)$ donc $S_y(z) = S_e(z)$.

L'équation clef

Soit un code de Goppa $\Gamma(\gamma, g)$. Soit y le mot reçu. Alors $y = c + e$ avec $c \in \Gamma(\gamma, g)$ et e un vecteur d'erreur de poids au plus égal à t .

Posons $T_e = \{i : e_i = 1\}$.

L'ensemble T_e est l'ensemble des positions d'erreurs.[Ndollane]p50

Définition 9 Soit $y = c + e \in \mathbb{F}_{2^m}^n$, avec $y = (y_1, y_2, \dots, y_n)$, $c = (c_1, c_2, \dots, c_n)$ un mot du code et $e = (e_1, e_2, \dots, e_n)$ un vecteur d'erreur.

On appelle polynôme localisateur d'erreur et on note σ_e , le polynôme unitaire

$$\sigma_e(z) = \prod_{i \in T_e} (z - \gamma_i).$$

Le polynôme $w_e(z)$ donné par

$$w_e(z) = \sum_{i \in T_e} e_i \prod_{j \in T_e \setminus i} (z - \gamma_j).$$

est appelé polynôme évaluateur d'erreur.

Une racine du polynôme σ_e indique une position d'erreur :

$$\sigma_e(\gamma_i) = 0 \Leftrightarrow e_i = 1 \Leftrightarrow i \in T_e.$$

Cela justifie le nom de polynôme localisateur d'erreur.

Si $y = c + e$, on a $S_y(z) = S_e(z) = \sum_{i=0}^n \frac{e_i}{z - \gamma_i} \mod g(z)$ et $\sigma_e(z) = \prod_{i \in T_e} (z - \gamma_i)$.

$$\begin{aligned} \sigma'_e(z) &= \sum_{i=0}^n \prod_{j \in T_e \setminus i} (z - \gamma_j) \\ &= \sum_{i \in T_e} \frac{1}{z - \gamma_i} \prod_{j \in T_e} (z - \gamma_j) \\ &= \sum_{i \in T_e} \frac{e_i}{z - \gamma_i} \sigma_e(z) \\ &= \sigma_e(z) \sum_{i \in T_e} \frac{e_i}{z - \gamma_i} \end{aligned}$$

Donc $\sigma'_e(z) \mod g(z) = \sigma_e(z) S_e(z)$.

Equation clef

On appelle équation clef, l'équation :

$$\sigma_e(z) S_e(z) = \sigma'_e(z) \mod g(z) \quad (1.1)$$

Résolution de l'équation à clef

Résoudre l'équation à clef revient à retrouver les polynômes $\sigma_e(z)$ et $w_e(z)$, ce qui permettra de retrouver le vecteur d'erreur e . [Ndollane]p51

On peut décomposer le polynôme $\sigma_e(z)$ en une partie paire et une partie impaire de la façon suivante : $\sigma_e(z) = a(z)^2 + zb(z)^2$.

Le vecteur d'erreur e étant de poids au plus égal à t , on a $\deg \sigma_e \leq t$, donc $\deg(a(z)) \leq \left\lfloor \frac{t}{2} \right\rfloor$, $\deg(b(z)) \leq \left\lfloor \frac{t-1}{2} \right\rfloor$.

$$\sigma'_e(z) = 2a(z) + b(z)^2 + 2zb(z)^2 = b(z)^2$$

puisque la caractéristique est égale à 2.

$$\begin{aligned}
b(z)^2 \mod g(z) &= \sigma'_e(z) \mod g(z) \\
&= \sigma_e(z) S_e(z) \\
&= [a(z)^2 + zb(z)^2] S_e(z)
\end{aligned}$$

Puisque $e \notin \Gamma(\gamma, g)$, alors $S_e(z) \neq 0$ et, de plus, pour tout $\gamma_j \in \mathbb{F}_{2^m}$ la fonction $z \mapsto z - \gamma_j$ est irréductible modulo $g(z)$ car ces deux fonctions sont premières entre elles, donc $S_e(z) \in \mathbb{F}_{2^m}[z]/g(z)$ et est inversible modulo $g(z)$.

L'application $\mathbb{F}_{2^m} \mapsto \mathbb{F}_{2^m}$ telle que $a \mapsto a^2$ est un automorphisme de Frobenius sur \mathbb{F}_{2^m} . Alors chaque élément de \mathbb{F}_{2^m} admet une unique racine carrée.

Sachant que $z + S_e^{-1}(z) \in \mathbb{F}_{2^m}[z]/g(z) = \mathbb{F}_{2^m}$, on peut alors poser $\tau_e(z) = \sqrt{z + S_e^{-1}(z)}$. Or $b(z)^2 = (a(z)^2 + zb(z)^2) S_e(z)$, donc $a(z)^2 = b(z)^2 (1 + zS_e(z))$.

D'où

$$a(z) = b(z)\tau_e(z) \mod g(z) (*) \quad (1.2)$$

Pour résoudre l'équation (2.2), il faut déterminer $a(z)$ et $b(z)$ de plus petit degré. La résolution de cette équation peut se faire en appliquant l'algorithme d'Euclide étendu ou l'algorithme de Berlekamp-Massey. Nous donnons ci-dessus un algorithme de décodage d'un code de Goppa binaire irréductible.

Donons un exemple d'application de l'algorithme précédent.

Exemple : Considérons le code de Goppa $\Gamma(\gamma, g)$ avec $\gamma = \mathbb{F}_{2^4}$ et $g(z) = z^2 + z + \beta$ où β est un élément primitif de \mathbb{F}_{2^4} . On a donc $\gamma = (0, 1, \beta, \beta^2, \dots, \beta^{14})$.

Soit $y = (1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0)$ le mot reçu.

Appliquons l'algorithme précédent pour le décodage.

1. Calculons le syndrome $S_y(z)$ de y .

$$S_y(z) = \sum_{i=0}^{15} \frac{y_i}{z - \gamma_i}$$

$$S_y(z) = \frac{1}{z} + \frac{1}{z - \beta} + \frac{1}{z - \beta^3} + \frac{1}{z - \beta^7} + \frac{1}{z - \beta^9} + \frac{1}{z - \beta^{13}}$$

$$S_y(z) = (\beta^{14} + 0 + \beta^{10} + \beta^9 + \beta^{11} + \beta^3) + (\beta^{14} + \beta^{14} + \beta^6 + \beta^{11} + \beta^9 + \beta^{11})z$$

$$S_y(z) = \beta^{13} + \beta^{10}z$$

2. Calculons le polynôme localisateur d'erreur $\sigma_e(z)$.

En appliquant l'algorithme d'Euclide étendu, on trouve $k(z)$ l'inverse de $S_y(z)$ modulo $g(z)$: $k(z) = \beta^{11}z + 1$.

Calculons $\tau_e(z)$: $\tau_e^2(z) = k(z) + z = (\beta^{11} + 1)z + 1$. Soient c_1, c_2 tels que $\tau_e(z) = c_1 +$

Algorithm 1 Décodage d'un code de Goppa binaire irréductible $\Gamma(\gamma, g)$

[Ndollane]

Require: y le mot reçu, t degré du polynôme g tel que y contient au plus t erreurs

Ensure: (r) : le mot envoyé

- 1: Calculer le syndrome du mot y : $S_y(z) = \sum_{i=0}^n \frac{y_i}{z - \gamma_i}$
 - 2: Déterminer le polynôme localisateur d'erreur :
 1. Appliquer l'algorithme d'Euclide étendu pour trouver l'inverse de $S_y(z) \bmod g(z)$: trouver $k(z)$ tel que $S_y(z)K(z) = 1 \bmod g(z)$
 - 2.
 3. **if** $k(z) = z$ **then**
 4. $\sigma_e(z) = z$
 5. **end if**
 6. Calculer $\tau_e^2(z) = k(z) + z \bmod g(z)$
 7. Trouver $a(z)$ et $b(z)$ de petits degrés vérifiant :
 8. $\tau_e(z)b(z) = a(z) \bmod g(z)$.
 9. Faire $\sigma_e(z) = a(z)^2 + zb(z)^2$
 - 3: Déterminer l'ensemble des positions d'erreurs $\mathbf{T}_e = \{i : \sigma_e(\gamma_i) = 0\}$
 - 4: Le vecteur d'erreur $e = (e_1, \dots, e_n)$ est défini par $e_i = 1$ si $i \in \mathbf{T}_e$ et $e_i = 0$ sinon
 - 5: Le mot envoyé est $c = y - e$
 - 6: **Retourne** c
-

$$c_2 z, \text{ alors } \tau_e^2(z) = c_1^2 + c_2^2 z^2 = c_1^2 + c_2^2 z^2 + c_2^2 g(z) = (c_1^2 + c_2^2 \beta) + c_2^2 z.$$

On a donc le système

$$\begin{cases} c_2^2 = \beta^{14} \\ c_1^2 + c_2 \beta = 1 \end{cases}$$

La solution de ce système est $c_1 = 0$ et $c_2 = \beta^7$.

D'où $\tau_e(z) = \beta^7 z$.

Trouvons $a(z)$ et $b(z)$ de petits degrés tels que $S_e(z)b(z) = a(z) \pmod{g(z)}$. Posons $a(z) = \tau_e(z) = \beta^7 z$ et $b(z) = 1$. Avec ce choix, $a(z)$ et $b(z)$ vérifient bien la condition précédente.

Ainsi $\sigma_e = \beta^{14} z^2 + z = \beta^{14} z(z + \beta)$.

3. Déterminons les positions d'erreurs

Les racines du polynôme $\sigma_e(z) = \beta^{14} z(z + \beta)$ sont $\gamma_1 = 0$ et $\gamma_3 = \beta$. Par suite, on a :

$\mathbf{T}_e = \{1, 3\}$, i.e. $e = (1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

4. Le mot envoyé est donc $c = y - e = (0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0)$.

1.1.8 Code Cyclique

Les codes cycliques sont les plus utilisés dans la pratique. Leur mise en oeuvre est en effet assez simple et la richesse de leurs propriétés algébriques permet d'en faire une étude assez approfondie.

1.1.9 Définition et description

Définition 10 Un code C de longueur n est dit cyclique si :

i) C est un code linéaire.

ii) Si $x = (x_1, x_2, \dots, x_n) \in C$ alors $(x_1, x_2, \dots, x_{n-1}) \in C$ c'est-à-dire toute permutation circulaire d'un mot de C appartient à C .

Remarque :

La permutation circulaire des composantes vers la droite de la propriété ii) est aussi appelée "shift". On dit aussi que $(x_n, x_1, x_2, \dots, x_{n-1})$ est le "shift" de (x_1, x_2, \dots, x_n)

1.1.10 Représentation polynomiale

Soit C un code cyclique de longueur n sur F_p . Pour chaque mot de c du code C , on numérote les positions en commençant par 0.

Soit $c = (a_0, \dots, a_{n-1})$, on lui associe le polynôme $c(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$. Le mot "shifté" $(a_{n-1}, a_0, \dots, a_{n-1})$ est associé au polynôme $a_{n-1} + a_0x + \dots + a_{n-2}x^{n-1}$. C'est celui que l'on obtient à partir de $c(x)$ en calculant $xc(x)$ et en considérant que $x^n = 1$ autrement dit en calculant modulo $(x^n - 1)$.

Par conséquent C est cyclique si et seulement si, pour tout c de C , $xc(x)$, calculé modulo $(x^n - 1)$, est le polynôme associé à un mot c' de C .

Ceci nous conduit à introduire les définitions suivantes :

Définition 11 Soit K un corps fini et n un entier, $n \geq 1$.

1. On appelle représentation polynômiale de \mathbb{F}_p^n , l'application ϕ de \mathbb{F}_p^n dans $K[x]/(x^n - 1)$ définie par $\phi : (a_0, \dots, a_{n-1}) \rightarrow c(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$
2. Si $c = (a_0, \dots, a_{n-1})$ alors le polynôme :
 $c(x) = \phi(c) = c(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$ est la représentation polynômiale de C .
3. On appelle représentation polynomiale d'un code C , l'ensemble des représentations polynomiales des mots de C . On la note $\phi(C)$.

Grâce à l'isomorphisme ϕ , on confond, dans le langage et l'écriture, les mots d'un code avec leurs représentations polynomiales.

En effet, c'est dans le but d'utiliser la structure multiplicative de l'anneau des polynômes.

Proposition 6 Si c appartient à un code cyclique C , alors pour tout polynôme $a(x)$ de $\mathbb{F}_p[x]$, $c'(x) = a(x)c(x) \mod (x^n - 1)$ appartient à $C(x)$.

Preuve : Soit $a(x) = \sum_i a_i x^i$

$$c'(x) = \sum_i a_i x^i c(x) \mod (x^n - 1)$$

Etant donné que C est cyclique, $xc(x) \in C(x)$ et il en est de même pour $x^i c(x)$, $\forall i \in \mathbb{N}$

Par conséquent, du fait que C est linéaire, $(a_0, \dots, a_{n-1})c(x)$ est aussi dans $C(x)$.

Remarque : D'une manière générale C est cyclique si et seulement si $C(x)$ est un sous-espace vectoriel, et si tout multiple modulo $(x^n - 1)$ d'un polynôme de $C(x)$ est aussi dans $C(x)$.

On peut traduire ce résultat dans le langage de l'algèbre classique.

Théorème 3 Soit C un code linéaire sur \mathbb{F}_p , C est cyclique si et seulement si sa représentation polynomiale $C(x)$ est un idéal de $\mathbb{K}_p[x]/(x^n - 1)$.

Preuve :

Les calculs se font modulo $(x^n - 1)$

- Si C est cyclique alors sa représentation polynomiale $C(x)$ est un espace vectoriel sur \mathbb{F}_p , c'est-à-dire un sous groupe additif de

$$\mathbb{F}_p[x]/(x^n - 1).$$

De plus d'après la proposition précédente, $\forall a(x) \in \mathbb{F}_p[x]$ et $\forall c(x) \in C(x), \forall a(x)c(x) \in C(x)$ donc $C(x)$ est un idéal.

- Inversement, si $C(x)$ est un idéal, un multiple quelconque d'un élément de $C(x)$ est aussi dans $C(x)$. En particulier, $xc(x)$ étant un multiple de $c(x)$, on en déduit que C est cyclique.

Remarque :

Rechercher tous les codes cycliques de longueur n sur \mathbb{F}_p revient à rechercher tous les idéaux de $\mathbb{F}_p[X]/(X^n - 1)$.

1.1.11 Polynôme générateur

Théorème 4 *Tout idéal de $\mathbb{F}_p[X]/(X^n - 1)$ est un idéal principal.* [M.Ibrahima_Mbaye]p48

Preuve :

Soit I un idéal de $\mathbb{F}_p[X]$ non réduit à $\{0\}$.

Soit $g(x)$ un polynôme non nul de I de degré minimal. Par la définition euclidienne dans $\mathbb{F}_p[x]$, on obtient $\forall i(x) \in I, i(x) = g(x)q(x) + r(x)(*)$, avec $\deg(r) < \deg(g)$ où $r(x) = 0$

En utilisant l'homomorphisme ϕ de $\mathbb{F}_p[x]$ dans $\mathbb{F}_p[X]/(X^n - 1)$ qui associe à chaque polynôme son reste par la division euclidienne par $x^n - 1$, et puisque les degrés de $i(x), g(x), q(x)$ et $r(x)$ sont strictement inférieurs à n , l'égalité $(*)$ reste vraie dans $\mathbb{F}_p[X]/(X^n - 1)$. Etant donné que I est un idéal et $g(x) \in I$,

$$g(x)q(x) \in I \text{ et } i(x) - g(x)q(x) = r(x) \in I.$$

Or d'après le choix de $g(x), \deg(r) < \deg(g) \Rightarrow r(x) = 0$

$$\text{Donc } i(x) - g(x)q(x) = 0$$

$$\text{D'où } i(x) = g(x)q(x)$$

Si $I = \{0\}$, la propriété est évidente.

De ce théorème découle la conséquence suivante :

Théorème 5 *La représentation polynomiale dans $\mathbb{F}_p[x]/(x^n - 1)$ d'un code cyclique est formée par tous les multiples d'un même polynôme $g(x)$ unitaire de degré minimal.* [M.Ibrahima_Mbaye]p48

Définition 12 Un tel polynôme $g(x)$ s'appelle un générateur du code cyclique. Il engendre C . On note $C(x) = \langle g \rangle$.

Remarque :

Un même code peut avoir plusieurs générateurs, mais on peut privilégier l'un d'entre eux.

Théorème 6 Le polynôme générateur $g(x)$ d'un code cyclique de longueur n et de dimension k sur \mathbb{F}_p divise $x^n - 1$ dans $\mathbb{F}_p[x]$. [M.Ibrahima_Mbaye]p49

Preuve :

La division euclidienne de $x^n - 1$ par g donne :

$$x^n - 1 = a(x)g(x) + r(x) \text{ avec } \deg(r) < \deg(g) \text{ où } r(x) = 0$$

Etant donné que les calculs se font modulo $(x^n - 1)$ on a :

$$0 = a(x)g(x) + r(x) \text{ ce qui implique que } r(x) = -a(x)g(x) \text{ dans } \mathbb{F}_p[x]/(x^n - 1)$$

Or $\deg(r) < \deg(g)$ on a : $r(x) = 0$

D'où $g(x)$ divise $x^n - 1$.

Théorème 7 Unicité du polynôme générateur [M.Ibrahima_Mbaye]p49

Chaque code cyclique de longueur n sur \mathbb{F}_p , non réduit à $\{0\}$, possède un générateur et un seul dont le coefficient dominant est 1 (c'est un polynôme est unitaire).

Preuve :

Soit $f(x)$ un polynôme générateur de $C(x)$, on a :

$$g(x) = a(x)f(x) \mod (x^n - 1), \text{ avec } a(x) \in \mathbb{F}_p[x]$$

$g(x) = a(x)f(x) \mod (x^n - 1)$ signifie dans $\mathbb{F}_p[x]$ que $g(x)$ est le reste de la division euclidienne de $a(x)f(x)$ par $x^n - 1$.

C'est-à-dire $r(x) = g(x) = a(x)f(x) + b(x)(x^n - 1)$, $b(x) \in \mathbb{F}_p[x]$.

Puisque $f(x)$ divise $x^n - 1$, il divise aussi $g(x)$ dans $\mathbb{F}_p[x]$

$g(x)$ étant un générateur du code, on peut échanger les rôles de $f(x)$ et $g(x)$ et on obtient $g(x)$ divise $f(x)$. D'où $f(x) = g(x)$ puisqu'ils ont 1 pour coefficient dominant.

Conséquence :

On appelle le générateur de C ce générateur privilégié qui divise $x^n - 1$.

Ainsi la recherche de tous les codes cycliques de longueur n sur \mathbb{F}_p qui était celui de tous les idéaux de $\mathbb{F}_p[x]/(x^n - 1)$ devient plus simple dans la mesure où elle se réduit à la recherche de tous les diviseurs de $x^n - 1$.

Exemple :

La décomposition de $x^7 - 1$ en diviseurs irréductibles sur \mathbb{F}_2 est :

$$x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x + 1)$$

Les codes cycliques de longueur 7 sur \mathbb{F}_2 , différents de $\{0\}$ sont donc donnés par la décomposition de $x^7 - 1$. Ce sont :

$$C_0 : g_0(x) = x^7 - 1 = 0, C_0 = \{(0, 0, \dots, 0)\}$$

$$C_1 : g_1(x) = x - 1$$

$$C_2 : g_2(x) = x^3 + x + 1$$

$$C_3 : g_3(x) = x^3 + x^2 + 1$$

$$C_4 : g_4(x) = (x - 1)(x^3 + x + 1) = x^4 + x^3 + x^2 + 1$$

$$C_5 : g_5(x) = (x - 1)(x^3 + x^2 + 1) = x^4 + x^2 + x + 1$$

$$C_6 : g_6(x) = (x^3 + x + 1)(x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

La représentation polynomiale $C_i(x)$ est formée par tous les multiples, modulo $(x^7 - 1)$ de $g_i(x)$ et C_i est donc formé par tous les 7-uplets correspondants.

Théorème 8 Inversibilité des éléments dans $\mathbb{F}_2/(x^p + 1)$ ([8 p.12])

Soit p un nombre premier tel que $\text{ord}_p(2) = p - 1$. Soit g un polynôme binaire dans $R = \mathbb{F}_2/(x^p + 1)$ avec $\deg(g) > 0$. Alors g admet un inverse multiplicatif dans $\mathbb{F}_2/(x^p + 1)$ si et seulement si il contient un nombre impair de termes et $g \neq \Phi$ avec $\Phi(x) = x^{p-1} + x^{p-2} + \dots + x + 1$.

Preuve : Si $g \in \mathbb{F}_2/(x^p + 1)$ a un nombre impair de termes et $g \neq \Phi$. Considérons $\text{pgcd}(g(x), x_p + 1) = \text{pgcd}(g, (x + 1)\Phi)$. Puisque $x + 1 \nmid g$ lorsque $g(1) = 1$ alors $\text{pgcd}(g, x + 1) = 1$ c'est-à-dire qu'ils sont co-premiers. De plus, puisque $\text{ord}_p(2) = p - 1$, Φ est irréductible sur $\mathbb{F}_2[x]$, ce qui signifie $g \nmid \Phi$.

Supposons, par contradiction, que $\Phi \mid g$, alors $\Phi = gh$ pour un certain $h \in R$. Alors $\deg(g) + \deg(h) = p - 1$ mais $\deg(g) \leq p - 1$. En cas d'égalité, on aurait $\deg(h) = 0$ ou $h = 0$ ou 1 . Si $h = 1$ alors $gh \mid \Phi$ donc $g \cdot 1 \mid \Phi$ qui est une contradiction. Si $h = 0$ alors $g \cdot 0 = \Phi = x^{p-1} + x^{p-2} + \dots + 1$, ce qui est une autre contradiction. Ainsi $\Phi \nmid g$ donc $\text{pgcd}(g, \Phi) = 1$.

Enfin, nous savons que $\text{pgcd}(g(x), x^p + 1) = \text{pgcd}(g(x), (x + 1)\Phi) = 1$ implique que g est inversible.

Si $g \in F_2[x](xp + 1)$, avec $\deg(g) > 0$ est inversible alors

$$\text{pgcd}(g(x), x^p + 1) = \text{pgcd}(g(x), (x + 1)\Phi) = 1.$$

Ainsi

$$\text{pgcd}(g(x), x+1) = 1$$

et

$$\text{pgcd}(g, \Phi) = 1$$

impliquant $g \neq \Phi$ et $g(1) = 1$. Supposons, par contradiction, qu'il existe un nombre de termes dans g alors $g(1) = 0$ ce qui est une contradiction. D'où le théorème fonctionne dans les deux sens.

Matrice circulante et quasi-circulante

Une matrice carrée est dite circulante si elle est construite de sorte que ses lignes soient les décalages cycliques successifs de la première ligne.

Une telle matrice s'écrit comme suit :

$$M = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_n \\ a_n & a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_n & a_0 & \cdots & a_{n-2} \\ & & & \ddots & a_0 \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix}$$

La matrice circulante est donc entièrement définie par la donnée d'une seule de ses lignes.

Remarque : Un code cyclique admet au moins une matrice génératrice(ou de contrôle de parité) circulante.

Une matrice est dite quasi-circulante lorsqu'elle est formée de blocs circulants. Une telle matrice s'écrit comme suit :

$$M = \begin{bmatrix} a_0 & a_1 & a_2 & b_0 & b_1 & b_2 \\ a_2 & a_0 & a_1 & b_2 & b_0 & b_1 \\ a_1 & a_2 & a_0 & b_1 & b_2 & b_0 \end{bmatrix}$$

On remarque que cette matrice est formée de deux blocs circulants. On a donc :

$M = [A|B]$ avec

$$A = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{bmatrix} \text{ et } B = \begin{bmatrix} b_0 & b_1 & b_2 \\ b_2 & b_0 & b_1 \\ b_1 & b_2 & b_0 \end{bmatrix}$$

1.1.12 Codes quasi-cycliques

Au sein des codes correcteurs d'erreurs et de la théorie de l'information, les codes quasi-cycliques tiennent une place particulière. Ces codes sont une généralisation des codes cycliques et possèdent en plus d'excellents paramètres : ils ont une grande capacité de correction.

Définition 13 *Un code est dit **quasi-cyclique** lorsqu'il admet une matrice génératrice (ou de contrôle de parité) quasi-circulante.*

Remarque : Un code quasi-cyclique admet au moins une matrice génératrice sous forme systématique et quasi-circulante.

Définition 14 : Codes Quasi-cycliques [N.Apon], p.2]

Un code binaire quasi-cyclique (QC) d'indice n_o et d'ordre r est un code linéaire qui admet comme matrice génératrice une matrice circulante par blocs d'ordre r et d'indice n_o . Un code (n_o, k_o) -QC est un code quasi-cyclique de indice n_o , longueur $n_o r$ et dimension $k_o r$ [baldi2014], p.41

Proposition 7 : Propriétés des Codes Quasi-cycliques [baldi2020], p.23]

Un code quasi-cyclique est défini comme un bloc linéaire de dimension $k = p k_o$ et de longueur $n = p n_o$ ayant pour propriétés suivantes.

- (i) *Chaque mot de code est formé d'une série de p blocs de n_o symboles, dont chacun est formé d'une suite de p blocs de n_o symboles suivis d'une redondance $n_o - k_o$ symboles.*
- (ii) *Chaque décalage cyclique d'un mot de code de n_o symboles donne un autre mot de code valide*

Définition 15 : (Matrice Binaire Circulante)[baldi2014], p.33

Une matrice binaire circulante A est une $v \times v$ sur \mathbb{F}_2 définie comme suit :

$$A = \begin{pmatrix} a_0 & a_1 & \cdots & a_{v-1} \\ a_{v-1} & a_0 & \cdots & a_{v-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{pmatrix}$$

tel que pour tout $0 \leq i < v$, $a_i \in \mathbb{F}_2$. C'est par définition une matrice non singulière puisque chaque ligne et chaque colonne sont des décalages cycliques les uns des autres.

1.2 Les codes LDPC et MDPC

Les codes LDPC (Low Density Parity Check) ont été introduits en 1960 par Robert G. Gallager dans [R_G_Gallager]. Ces codes sont constitués d'une matrice de contrôle de parité à densité faible. Pendant presque 35 ans, les codes LDPC ont été oubliés par la communauté de la théorie des codes. Il faudra alors attendre jusqu'en 1996 pour voir David J. C. McKay et R. M. Neal redécouvrir ces codes. Ce travail sera suivi de plusieurs autres, attestant leur excellente caractéristique de correction d'erreurs et de les promouvoir à la sélection au groupe de bons codes linéaires pour les applications de télécommunication.

Les codes MDPC (Moderate Density Parity Check), quant à eux, ont été introduits [10]. Ce sont des codes LDPC de densité plus élevée que ce qui est habituellement adopté pour les applications de télécommunications. Ce qui conduit en générale à une plus grande capacité de correction d'erreurs. Cependant, dans la cryptographie à base de codes, on ne s'intéresse pas nécessairement à corriger de nombreuses erreurs, mais seulement un certain nombre d'erreurs qui assure un niveau de sécurité adéquat, une condition que satisfait les codes MDPC. Ces derniers admettent une matrice de contrôle de parité à densité modérée, c'est-à-dire que les équations de parité définies par cette matrice sont de poids modéré.

Définition 16 (Code LDPC) : Un code $[n, r, w]$ –LDPC est un code de longueur n , co-dimension r et qui admet une matrice de contrôle dont les lignes ont un poids fixe w , où $w = O(1)$.

Définition 17 (Code MDPC) : Un code $[n, r, w]$ –MDPC est un code de longueur n , co-dimension r et qui admet une matrice de contrôle dont les lignes ont un poids fixe w , où $w = O(\sqrt{n})$.

Lorsqu'on ajoute la structure quasi-cyclique à un code $[n, r, w]$ –MDPC, on notera ce code $[n, r, w]$ –QC – MDPC.

Dans ce qui suit, nous donnons quelques notions de sécurité.

1.2.1 Définitions de quelques notions de sécurité

Mécanismes d'Encapsulation et de Décapsulation

Un schéma de chiffrement asymétrique nécessite l'utilisation d'une paire de clés pour chaque entité engagée dans un protocole de communication : la clé publique qui sert de chiffrement et la clé privée qui sert de déchiffrement. Pour un schéma de chiffrement symétrique, deux entités en communication ont la même clé : la clé secrète utilisée pour le chiffrement et le déchiffrement.

Par ailleurs, les schémas de chiffrement symétriques sont beaucoup plus rapides que les schémas de chiffrement asymétriques. il est donc plus avantageux, si on se focalise sur la durée d'exécution, d'utiliser un schémas de chiffrement symétrique. Cependant, utiliser un cryptosystème à clé secrète nécessite le partage de la clé secrète entre les entités concernées. Cela les expose à des attaques sur la clé secrète lors du partage de cette dernière, notamment l'attaque de l'homme du milieu.

Pour pallier ce problème, l'idée serait de combiner les deux types de chiffrement pour donner naissance à un nouveau mécanisme qui fonctionne comme suit :

1. L'entité **A**, par exemple, en plus de sa paire de clés, choisit une clé secrète k appelée clé de session destinée à être utilisée pour le chiffrement des données envoyées par **A** ou reçues de l'entité **B**.
2. L'entité **A** chiffre cette clé k avec la clé publique pk_B de **B** en utilisant un système de chiffrement asymétrique. **A** obtient ainsi k' . Cette opération s'appelle **Encapsulation** de la clé k .
3. A la réception du chiffre k' de la clé de session, **B** utilise sa clé privée sk_B pour déchiffrer k' et retrouver la clé de session k en utilisant le même système de chiffrement asymétrique que **A**. Cette opération s'appelle **Décapsulation**.
4. A ce stade, **A** et **B** ont la même clé de session k . Ces deux entités pourront maintenant utiliser la même clé k pour chiffrer et déchiffrer leurs données.

Un tel système est appelé *Mécanisme d'Encapsulation de Clé* (Key Encapsulation Mechanism (KEM)).

Trois mesures de complexité

De nombreux algorithmes quantiques font intervenir une fonction f à laquelle ils font appel de nombreuses fois lors de leur déroulement. Nous allons nous intéresser à la complexité en moyenne et nous allons distinguer trois types de complexité :

1. On définit la **complexité en requêtes** d'un algorithme comme étant le nombre de fois où il fait appel à la fonction f .
2. La **complexité en temps** ou **complexité temporelle** est définie comme le temps d'exécution moyen de l'algorithme. Par conséquent, elle est fonction de la complexité en requête et du temps d'exécution de la fonction f .
3. La **complexité en espace** ou **complexité spatiale** est définie comme l'espace mémoire occupée par les données internes de l'algorithme. Elle est non-cumulative contrairement à la complexité en temps car on peut souvent réutiliser de l'espace mémoire.

Sécurité IND-CPA, IND-CCA

La sécurité en termes d'indiscernabilité (Indistinguishability) a de nombreuses définitions, en fonction des hypothèses faites sur les capacités de l'attaquant. Il est normalement présenté comme un jeu dans lequel le cryptosystème est considéré comme sécurisé si aucun adversaire ne peut gagner la partie avec une valeur significativement plus grande probabilité qu'un adversaire qui doit deviner au hasard.

Les définitions les plus couramment utilisées en cryptographie, il y a l'indiscernabilité sous une attaque de texte en clair choisie (en abrégé IND-CPA), l'indiscernabilité sous une attaque de texte chiffré choisi (non adaptative) (IND-CCA), et indiscernabilité sous attaque de texte chiffré adaptatif choisi (IND-CCA2). Sécurité l'une ou l'autre de ces dernières définitions implique une sécurité selon les précédentes : un système qui est sécurisé IND-CCA est également sécurisé IND-CPA, et un schéma sécurisé IND-CCA2 est IND-CCA et IND-CPA sont sécurisés. Ainsi, IND-CCA2 est le plus fort de ces trois définitions de la sécurité.

L'indiscernabilité sous une attaque de texte en clair choisie (en abrégé IND-CPA)

Un cryptosystème est indiscernable sous une attaque en clair choisie (IND-CPA) si un adversaire disposant d'une capacité de calcul en temps polynomial d'une machine de Turing a un avantage négligeable de trouver un texte clair aléatoirement choisi.

Bien que la définition ci-dessus soit spécifique à un cryptosystème à clé asymétrique, elle peut être adaptée au cas symétrique en remplaçant la fonction de chiffrement à clé publique par un « chiffrement oracle », qui conserve la clé de cryptage secrète et crypte les textes chiffrés arbitraires au demande de l'adversaire.

Indiscernabilité sous l'attaque de texte chiffré choisie/adaptative choisie attaque de texte chiffré (IND-CCA, IND-CCA2)

Indiscernabilité sous attaque de texte chiffré choisi non adaptative et adaptative (IND-CCA, IND-CCA2) utilise une définition similaire à celle de IND-CPA. Cependant, outre le public clé (ou oracle de chiffrement, dans le cas symétrique), l'adversaire a accès à un « oracle de décryptage » qui déchiffre les textes chiffrés arbitraires à la demande de l'adversaire, renvoyant le texte en clair.

Dans la définition non adaptative, l'adversaire est autorisé à interroger cet oracle seulement jusqu'à ce qu'il reçoive le texte chiffré du défi.

Dans la définition adaptative, l'adversaire peut continuer à interroger l'oracle de décryptage même après avoir reçu un texte chiffré de défi, avec la mise en garde qu'il ne peut pas passer le texte chiffré du défi pour le déchiffrement (sinon, la définition serait triviale).

Enfin un schéma est sécurisé IND-CCA/IND-CCA2 si aucun adversaire n'a un avantage non

négligeable pour deviner un texte en clair.

Non-malleability

La malléabilité est la propriété d'un algorithme cryptographique. Un algorithme de chiffrement est malléable s'il est possible pour un adversaire de transformer un texte chiffré en un autre texte chiffré qui déchiffre en un texte en clair associé.

Cela reçoit un cryptage d'un texte en clair m , c'est possible de générer un autre texte chiffré qui déchiffre en $f(m)$, pour une fonction f connue, sans nécessairement connaître ou apprendre m . La malléabilité est souvent une propriété indésirable dans un cryptosystème à usage général, car elle permet à un attaquant de modifier le contenu d'un message.

1.2.2 Les Problèmes de Base de sécurité

Définition 18 ([overbeck2020], p.9) Le problème général du décodage d'un code linéaire sur une norme $|\cdot|$ est défini comme suit :

- Soient $C \in \mathbb{F}^{k \times n}$ un (n, k) code linéaire sur \mathbb{F} et $y \in \mathbb{F}^n$.
- Trouver $x \in C$ tel que $\|y - x\|$ soit minimale.

Définition 19 ([overbeck2020], p.9) Le problème de trouver les poids du sous-espace d'un linéaire Le code est défini comme suit :

- Soit C un (n, k) code linéaire sur \mathbb{F} et $y \in \mathbb{F}^n$ avec $\|y\| = w$.
- Trouver $x \in C$ tel que $\|x\| = w$.

Le problème de trouver des poids de sous-espace d'un code linéaire a été défini à l'origine par Berlekamp, McEliece et Van Tilborg [berlekamp1978] en utilisant la norme de Hamming.

Produit tensoriel Matriciel : Le produit de Kronecker

Le produit de Kronecker est un cas particulier du produit tensoriel sur les matrices.

Soient \mathbf{A} une matrice $k \times l$ et \mathbf{B} une matrice $m \times n$.

Définition 20 Le produit de Kronecker Le produit de Kronecker de \mathbf{A} et \mathbf{B} (noté $\mathbf{A} \otimes \mathbf{B}$) est la matrice $km \times ln$ donnée par :

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1l}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2l}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1}\mathbf{B} & a_{k2}\mathbf{B} & \cdots & a_{kl}\mathbf{B} \end{pmatrix}.$$

Exemple : On donne deux matrices A et B

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Le produit de Kronecker de **A** et **B**, noté $\mathbf{A} \otimes \mathbf{B}$ est :

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} & 1 \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} & 1 \cdot \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{pmatrix}.$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Chapitre 2

LE CONCOURS DE NIST

Contents

2.1	Motivation	34
2.2	Les Algorithmes de chiffrement	35
2.3	Classic McEliece	35
2.3.1	Paramètres du système	35
2.3.2	Génération de clé	36
2.3.3	Encodage	37
2.3.4	Décodage	37
2.3.5	Encapsulation	39
2.3.6	Décapsulation	40
2.3.7	Analyse de sécurité de classic McEliece	41
2.3.8	Les Attaques sur le système de McEliece classique	41
2.4	BIKE :Bit-flipping Key Encapsulation	42
2.4.1	Problème de Base de sécurité	43
2.4.2	Les Paramètres du système de BIKE	46
2.4.3	Génération de clé	47
2.4.4	Encodage	47
2.4.5	Désencapsulation	48
2.4.6	fonction de décodage de BIKE	49
2.4.7	Les Attaques sur le système de BIKE	51
2.4.8	Analyse de Sécurité	52
2.5	Hamming Quasi - Cyclic	52
2.5.1	Problèmes de Bases de Sécurité	55
2.5.2	Les Paramètres du Système	58
2.5.3	Génération de clés	59
2.5.4	Encryption / Encapsulation	59
2.5.5	Décryptage / Décapsulation	61
2.5.6	Analyse de Sécurité	61
2.5.7	Les attaques sur HQC	62

2.1 Motivation

En parlant de NIST, Institut national des normes et de la technologie, est une agence du département du Commerce des États-Unis. Son but est de promouvoir l'économie en développant des technologies, la métrologie et des normes de concert avec l'industrie. Cette agence a pris la suite en 1988 du National Bureau of Standards fondé en 1901 avec substantiellement les mêmes missions qui sont :

- Normalisation : NIST est responsable de l'établissement, de la promotion et de la diffusion de normes de mesure, de sécurité, de qualité et de technologies pour divers secteurs de l'industrie, du gouvernement et du grand public.
- Métrologie : NIST assure la maintenance et la diffusion des unités de mesure de base pour les États-Unis, conformément au Système international d'unités (SI). Il joue un rôle essentiel dans l'établissement d'étalons de mesure précis pour les industries et les laboratoires de recherche.
- Recherche et développement technologique : NIST effectue des recherches avancées dans différents domaines technologiques tels que l'informatique, l'intelligence artificielle, la cybersécurité, les matériaux, l'électronique, les télécommunications, les sciences des données, etc. Leurs travaux de recherche visent à promouvoir l'innovation et à soutenir le développement technologique aux États-Unis.
- Cybersécurité : NIST joue un rôle important dans l'élaboration de normes et de bonnes pratiques en matière de cybersécurité, y compris le développement de cadres de cybersécurité tels que le Framework NIST pour l'amélioration de la cybersécurité (NIST Cybersecurity Framework).
- Promotion de l'industrie et de l'innovation : NIST collabore avec l'industrie, les universités et d'autres partenaires pour soutenir l'innovation technologique et favoriser la compétitivité économique des États-Unis.[ANSSI]

En 2016 le NIST(National Institute of Standards and Technology) a lancé un concours international dans le but de standardiser les algorithmes post-quantiques. Le concours a également pour but de stimuler la recherche et le développement dans ce domaine, et de promouvoir la collaboration entre les chercheurs et les entreprises travaillant sur les technologies de chiffrement post-quantique. Il s'est déroulé sur 4 rounds :

1. Round 1 : Ce tour a été lancé en novembre 2017. Dans ce tour, NIST a reçu 82 propositions d'algorithmes de chiffrement post-quantique. Parmi eux, 69 ont été retenus après une première évaluation.
2. Round 2 : Ce tour a débuté en janvier 2019. Dans ce tour, les 69 propositions retenues lors du Round 1 ont été examinées plus en détail. Les candidats ont été évalués sur la base de critères tels que la sécurité, la performance et la flexibilité. À l'issue de ce

- tour, 26 candidats ont été retenus pour participer au Round 3.
3. Round 3 : Ce tour a commencé en janvier 2020. Dans ce tour, les 26 candidats retenus lors du Round 2 ont été évalués plus en profondeur. Les candidats ont été évalués sur la base de critères tels que la sécurité, la performance, la flexibilité et la simplicité de mise en œuvre. À l'issue de ce tour, 15 candidats ont été sélectionnés pour participer au Round 4.
 4. Round 4 : Ce tour a débuté en juillet 2021. Dans ce tour, les 15 candidats retenus lors du Round 3 ont été soumis à une évaluation plus rigoureuse. Les candidats ont été évalués sur la base de critères tels que la sécurité, la performance, la flexibilité, la simplicité de mise en œuvre et la qualité des implémentations proposées. Les résultats de ce tour n'ont pas encore été annoncés.

2.2 Les Algorithmes de chiffrement

Suivant les critères :

- la sécurité
- la performance
- la flexibilité
- la simplicité de mise en œuvre
- la qualité des implémentations proposées

Il a été retenu 3 algorithmes Classic McEliece, BIKE et HQC . Ceux-ci viennent en complément aux algorithmes déjà sélectionnés(fondés sur les réseaux euclidiens structurés, des constructions en arbres de hachage).

2.3 Classic McEliece

2.3.1 Paramètres du système

Pour créer un système KEM basé sur les codes Goppa, les paramètres système suivants doit être choisis. Cela se fait en pesant la sécurité par rapport à la taille des clés publiques et l'efficacité des opérations de chiffrement/déchiffrement.

Les paramètres de CM sont définis comme suit :

1. Un entier positif m tel que $q = 2^m$
2. n un entier positif tel que $n \leq q$
3. t entier positif $t \geq 2$ tel que $mt < n$. On définit aussi un paramètre $k = n - mt$.

4. Un polynôme minimal $f \in \mathbb{F}_2[x]$ de degré m . Ceci est défini une représentation $\mathbb{F}_2[x]/(f)$ dans le corps \mathbb{F}_q .
5. un entier positif l et une fonction cryptographique de hashage H qui renvoie l bits.
Il y'a trois types de inputs pour la fonction de hashage : $(2, v)$; $(1, v, C)$ et $(0, v, C)$ avec $v \in \mathbb{F}_2^n$, et C le text chiffré. Initialement on définit les bits 0, 1 ou 2 comme les données de la fonction de hashage.

2.3.2 Génération de clé

Afin de générer des clés, les informations suivantes doivent être choisies avec suffisamment d'aléa pour être cryptographiquement sécurisées. C'est-à-dire, en utilisant un vrai nombre aléatoire générateur ou, au moins un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé.

Étant donné un ensemble de paramètres CM, un utilisateur génère une paire de clés CM.

Algorithme : Generation de paire de clé CM

Entrées : (m, n, t, f, l) : clé public

Sortie : (A) : la clé publique de CM

(s, Γ) : clé privée de CM

1. On génère aléatoirement un polynôme minimal irréductible $g \in \mathbb{F}_q[x]$ de degré t .
2. On sélectionne une séquence aléatoire uniforme $(\alpha_1, \alpha_2, \dots, \alpha_n)$ de n éléments deux à deux distincts de \mathbb{F}_q tel que $g(\alpha_i) \neq 0$
3. On calcule la matrice $t \times n$ $\vec{H} = \{h_{i,j}\}$ sur \mathbb{F}_q , tel que $h_{i,j} = \alpha_j^{i-1} / g(\alpha_j)$ pour $i = 1, \dots, t$ et $j = 1, \dots, n$.
4. On forme une matrice $mt \times n$ \hat{H} sur \mathbb{F}_2 par remplacement de chaque entrée $c_0 + c_1x + \dots + c_{m-1}x^{m-1}$ de \vec{H} avec une colonne de t bits c_0, c_1, \dots, c_{m-1} .
5. On réduit la matrice \hat{H} à sa forme systématique $(I_{n-k} | A)$ avec I_{n-k} une matrice identité. Si on ne parvient pas à cette opération on retourne à l'étape 1.
6. on génère aléatoirement une chaîne s de n -bit.
7. L'ensemble $\Gamma = (g, \alpha_1, \alpha_2, \dots, \alpha_n)$ et la sortie (s, Γ) sont la clé privée et A est la clé publique.



La seconde partie de la clé privée $\Gamma = (g, \alpha_1, \alpha_2, \dots, \alpha_n)$ décrit un code de Goppa binaire de longueur n et de dimension $k = n - mt$. La clé publique \mathbf{A} est une $(n-k) \times k$ matrice tel que $[I_{n-k} | \mathbf{A}]$ avec I_{n-k} est une matrice de parité du code de Goppa.



La clé publique \mathbf{A} est une matrice de parité systématique du code de Goppa binaire en opposition avec la clé publique du chiffrement de McEliece original avec la production des matrices (S, G, P) . Ce qui nous permet de supprimer $(n-k)^2$ bits sur la taille de la clé publique.

2.3.3 Encodage

Le sous-programme de l'encodage prend deux entrées, un vecteur de poids t $e \in \mathbb{F}_2^n$ et la clé publique \mathbf{A} et retourne $\mathbf{c}_0 \in \mathbb{F}_2^{n-k}$.

Algorithme : Encodage CM[3,p.9]

Entrées :

(\mathbf{A}) : clé publique CM

(e) : un vecteur colonne de poids t

Sortie : ($\mathbf{c}_0 \in \mathbb{F}_2^{n-k}$: Un vecteur chiffré

1. On définit $\mathbf{H} = [I_{n-k} | \mathbf{A}]$

2. Calcul $\mathbf{c}_0 = \mathbf{H}e \in \mathbb{F}_2^{n-k}$.

3. on retourne \mathbf{c}_0 .



Connaissant la forme de la matrice de contrôle de parité, celle-ci est reconstituée à partir de la clé publique comme $[I_{n-k} | \mathbf{A}]$ puis utilisé pour calculer le codage du vecteur $\mathbf{c}_0 = \mathbf{H}e$.

2.3.4 Décodage

Le sous programme de décodage prend deux entrées un vecteur $\mathbf{c}_0 \in \mathbb{F}_2^{n-k}$ et une clé privée (s, Γ) et renvoie deux valeurs de retour possible soit un vecteur colonne de poids t $e \in \mathbb{F}_2^n$ ou une valeur d'erreur \perp .

Algorithme : Décodage de CM[3,p.9]

Entrées :

 (c_0) : un vecteur codé $c_0 \in \mathbb{F}_2^{n-k}$. (s, Γ) : la clé privée de CM.Sortie : $(c_0 \in \mathbb{F}_2^{n-k}$: un vecteur encodé1 On étend c_0 à $v = (c_0, 0, \dots, 0) \in \mathbb{F}_2^n$ en complétant k zéros.2 Trouver l'unique mot chiffré c du code de Goppa défini par Γ tel que sa distance par rapport à v est moins t . En cas d'échec retourner \perp (une valeur erronée).3 Calcul $e = v + c$.4 **Si** $\omega(e) = t$ et $c_0 = He$ **alors**5 retourne e 6 **sinon**7 retourne \perp .7 **fin if**

Remarque : En ce qui concerne l'étape 2, il existe de nombreuses façons différentes de décoder un code Goppa. L'algorithme de Patterson fournit une méthode efficace. le triplet d'algorithmes (Key Generation, Encoding, Décodage) est essentiellement la version "dual" de Niederreiter du cryptosystème McEliece.

Explication détaillée [3 section 2.5]

Pour voir pourquoi le sous-programme fonctionne, notez d'abord que le "syndrome" Hv est c_0 , car le premier $n-k$ positions de v sont multipliées par la matrice identité et les positions restantes sont zéro. Si c_0 a la forme He où e a un poids t alors $Hv = He$, donc $c = v + e$ est un mot de code.

Ce mot de code a une distance exactement t de v , et c'est le mot de code unique à distance $\leq t$ de v puisque la distance minimale de Γ est au moins $2t + 1$. Donc l'étape 2 trouve c , l'étape 3 trouve e , et l'étape 4 renvoie e .

Inversement, si le sous-programme renvoie e à l'étape 4, alors e a été vérifié d'avoir un poids t et d'avoir $c_0 = He$, donc si c_0 n'a pas cette forme alors le sous-programme doit retourner \perp . La logique repose ici sur l'étape 2 qui trouve toujours un mot de code à la distance t s'il en existe un. Cela fait ne pas se fier à l'étape 2 échouant dans les cas où un mot de code n'existe pas : le sous-programme reste corriger si, au lieu de retourner \perp , l'étape 2 choisit un vecteur $c \in \mathbb{F}_2^n$ et continue jusqu'à l'étape 3.



Les implémenteurs sont avertis qu'il est important d'éviter de divulguer des informations secrètes via canaux secondaires, et que la distinction entre le succès et l'échec dans ce sous-programme est secret dans le cadre du Classic McEliece KEM. En particulier, arrêter immédiatement le calcul lorsque l'étape 2 revient \perp révélerait cette distinction par le biais du temps, il est donc recommandé pour les implémenteurs d'avoir l'étape 2 toujours choisir certains $c \in \mathbb{F}_2^n$.



Comme autre note d'implémentation : Afin de tester $c_o = He$, le sous-programme de décodage n'a pas besoin de recalculer H à partir de Γ comme dans la génération de clé. Au lieu de cela, il peut utiliser n'importe quelle matrice de contrôle de parité H_o pour le même code. Le calcul utilise $v = (c_o, 0, \dots, 0)$ et compare $H_o v$ à $H_o e$. Les résultats sont égaux si et seulement si $v + e = c$ est un mot de code, ce qui implique $He = H(v + c) = Hv + Hc = Hv = c_o$. Il existe différents choix standard de H_o en rapport à \hat{H} qui sont facilement récupérées à partir de Γ , et qui peuvent être appliquées à des vecteurs sans utiliser espace quadratique.

2.3.5 Encapsulation

L'algorithme d'encapsulation **Encap** utilise la clé publique pk pour produire une encapsulation c et une clé $k \in K$

L'expéditeur génère une clé de session k et un texte chiffré C comme suit.

Algorithme : Encap CM[2,p.10]

Entrées :

(m, n, t, f, l)

(H) : Une fonction de hashage

Sortie :

(k) : La clé de session à partager.

$(C = c_o, c_1)$: Le ciphertext.

1. On génère uniformément un vecteur aléatoire $e \in \mathbb{F}_2^n$ de poids t .
2. On calcule c_o par l'encodage sur e et de clé publique A .
3. On calcule $c_1 = H(2, e)$.
4. On pose $C = (c_o, c_1)$
5. on calcule $k = H(1, e, C)$.
6. On retourne la clé de session k et le ciphertext C

L'algorithme d'encapsulation est une variante de la conversion Fujisaki-Okamoto dans laquelle aucun message n'est envoyé. Le but du KEM est de générer une clé secrète partagée $k \in \mathbb{F}_2^l$.

2.3.6 Décapsulation

Le sous programme **Decap** utilise la clé privée sk et l'encapsulation c pour retrouver la clé k ou retourner une erreur notée \perp .

Algorithme : DECAP CM[2,p.52]

Entrées :

(m, n, t, f, l) : Les paramètres de CM

(s, Γ) : la clé secrète sk

(H) : Une fonction de hashage

Sortie : (k) : La clé de session à partager.

1. Diviser le ciphertext C en (c_0, c_1) avec $c_0 \in \mathbb{F}_2^{n-k}$ et $c_1 \in \mathbb{F}_2^l$.
2. $b \leftarrow 1$.
3. On calcule e au sous programme décodage de paramètres c_0 et (s, Γ) . Si le sous programme retourne \perp alors $e \leftarrow s$ et $b \leftarrow 0$.
4. On calcule $c'_1 = H(2, e)$.
5. Si $c'_1 \neq c_1$ alors $e \leftarrow s$ et $b \leftarrow 0$.
6. On calcule $k = H(b, e, C)$.
7. On retourne la clé de session k .



Si C est le vrai texte chiffré alors $C = (C_0, C_1)$ avec $C = He$ pour $e \in \mathbb{F}_2^n$ de poids t et $c_1 = H(2, e)$. L'algorithme de décodage retournera e comme unique vecteur de poids t et si $c'_1 = c_1$ alors $b \leftarrow 1$ et on renvoie k la clé de session calculée dans l'encapsulation.



En tant que note d'implémentation, la sortie de décapsulation est inchangée si $e \leftarrow s$ à l'étape 3 est modifié pour affecter autre chose à e . Les implémenteurs peuvent préférer, par exemple, définir e à une valeur fixe chaîne de n bits, ou une chaîne aléatoire de n bits autre que s . Cependant, la définition de la décapsulation dépend du fait que e soit défini sur s à l'étape 5.

ID	m	n	t	l	field polynomial
mceliece348864	12	3488	64	256	$f(x) = x^{12} + x^3 + 1$
mceliece460896	13	4608	96	256	$f(x) = x^{13} + x^4 + x^3 + x + 1$
mceliece460896	13	6688	128	256	$f(x) = x^{13} + x^4 + x^3 + x + 1$

TABLE 2.1 – Les paramètres de classic Mc Eliece

2.3.7 Analyse de sécurité de classic McEliece

Certains des ensembles de paramètres recommandés et mis en œuvre sont les suivants.

Par exemple, dans **mceliece34886**, nous avons $m = 12$, $n = 3488 \leq 2^m$, $t = 64$ et $l = 256$. Le nombre de vecteurs possibles de poids t dans un message de longueur n est $\binom{n}{t} = \binom{3488}{64} \approx 2.3 \times 10^{137}$, qui est supérieur au nombre de partages possibles clés $k = 2^{256} \approx 1.2 \times 10^{77}$. La fonction de hachage dans l'implémentation recommandée est *SHAKE*₂₅₆, qui produit une valeur pseudo-aléatoire de 256 bits à partir du hachage de l'un de ces $\binom{3488}{64}$ vecteurs d'erreurs.

En cas d'échec lors du décodage, un hachage est effectué sur certains "garbage" données, cela empêche les attaques par canal latéral basées sur la synchronisation et la puissance puisque les mêmes opérations ont lieu même en cas de défaillance du KEM

2.3.8 Les Attaques sur le système de McEliece classique

La soumission de Classic McEliece au comité de normalisation du NIST fournit une longue histoire d'attaques sur ce système de 1981 à 2016. Chacune de ces attaques utilise une stratégie, développée par Prange en 1962, le décodage d'ensembles d'informations (ISD). [prange62] Tous, notamment, traitent la structure de la clé publique comme une matrice aléatoire uniforme. Aucune des attaques actuellement connues n'utilise la structure des codes Goppa pour récupérer information clé. En d'autres termes, ils s'attaquent au problème général de décodage et non spécifiquement le problème de McEliece. Dans l'article de Bernstein de 2010 "Grover vs McEliece" [Bergrover3], l'auteur fournit une description de l'ISD et l'utilisation de l'algorithme de recherche quantique de Grover afin d'attaquer le McEliece Cryptosystem. L'algorithme de décodage de l'ensemble d'informations est le suivant :

Algorithme :Information Set Decoding

Entrées :

(**G**) : Une matrice génératrice k colonnes et n lignes

($c = mG \oplus e$) : Le ciphertext avec m le message et e est le vecteur error de poids t

(j) : un entier positif tel que $t \leq j$

Sortie : (**m**) : le texte clair.

1. on choisit uniformément un vecteur aléatoire de taille k dans le sous ensemble $S \subseteq \{1, 2, \dots, n\}$.
2. On sélectionne k colonnes de **G** et **c** correspondant respectivement à G_s et c_s
3. Si G_s est inversible alors
4. $Q_1 \leftarrow G_s^{-1}$ et $Q_2 \leftarrow Q_1 G$
5. $m' \leftarrow c \oplus c_s Q_2$
6. si $\omega(m') = t$ alors
7. retourne m'
8. fin if
9. fin if
10. On recommence avec une nouvelle sélection de s .

Cet algorithme retourne deux valeurs :

- Le vecteur $e_s = 0$ c'est à dire aucune des k valeurs de S n'est indexé le vecteur erreur.
- La matrice G_s à k colonnes et k lignes est inversibles.

Sur ce, nous avons $\binom{n-t}{k}$ d'occurrence sur lesquelles $\binom{n}{k}$ sont ensembles de S .

En moyenne, cela se produit après $\binom{n}{k} / \binom{n-t}{k}$ itérations.

Aussi , pour $R = k/n$ et $t \approx (1 - R)n / \log_2(n)$ alors

$$\frac{\binom{n}{k}}{\binom{n-t}{k}} = \frac{n \cdots (n-t+1)}{(n-k) \cdots (n-k-t+1)} \approx \left(\frac{n}{n-k}\right)^t = \left(\frac{1}{1-R}\right)^t \approx C^{\frac{m}{\log_2 n}}.$$

Où $C = (1/(1-R))^{1-R}$.

2.4 BIKE :Bit-flipping Key Encapsulation

Le chiffrement BIKE a été proposé comme l'un des candidats potentiels pour les algorithmes post-quantiques et a été soumis à l'évaluation dans le cadre du processus de sélection de la norme de chiffrement post-quantique par le NIST (National Institute of Standards and Technology). Il a été sélectionné comme finaliste pour la norme de chiffrement post-quantique dans le cadre de la deuxième ronde d'évaluation du NIST.

BIKE (Bit Flipping Key Encapsulation) est un KEM basé sur une modulation quasi-cyclique

linéaire binaire. codes de vérification de la parité de la densité (QC-MDPC). Le cryptosystème BIKE a été initialement conçu pour une utilisation de clé éphémère, mais il a maintenant été revendiqué qu'il prend également en charge l'utilisation de clé statique.

BIKE est l'un des principaux concurrents pour les mécanismes de chiffrement de clé basés sur le code dans le cadre de la troisième round de la compétition de normalisation post-quantique du National Institute of Standards and Technology (NIST) qui est basé sur le problème de l'apprentissage de la parité avec du bruit (LPN). Il utilise une variante du chiffrement McEliece avec des codes binaires de Goppa et sa sécurité est basée sur la dureté de LPN. Le niveau de sécurité de BIKE peut être supérieur à la plupart des autres systèmes de cryptographie à base de code avec des paramètres comparables.

La norme BIKE est un mécanisme d'encapsulation de clé **IND-CCA** (KEM) utilisant **QC-MDPC** pour la création, l'encapsulation et la décapsulation d'une clé secrète partagée de 256 octets.

Il repose sur des codes basés sur la théorie des codes algébriques, qui peuvent résister aux attaques des ordinateurs quantiques. La proposition du NIST intitulée Bit Flipping Key Encapsulation (BIKE) combine les matrices circulantes avec l'idée de matrices de contrôle de parité à densité modérée.

L'utilisation de matrices circulantes permet de réduire la taille des clés, tandis que l'utilisation de matrices de contrôle de parité à densité modérée permet un décodage efficace avec une clé de chiffrement à l'aide d'un algorithme de retournement de bits. Au cours de cette section nous évoquerons les problèmes sur lesquels reposent la sécurité de Bike, l'encodage et le décodage puis nous allons une analyse de sécurité en parlant des attaques du système.

2.4.1 Problème de Base de sécurité

Les difficultés sur lesquelles repose la sécurité de BIKE

sont analogues à ceux des définitions 2.3.2 utilisant une norme de Hamming. Les deux ont été montrés d'être NP-Complet par Berlekamp, McEliece et Van Tilborg dans [14] pour générer (autres codes QC)

Soient $R = \mathbb{F}_2[x]/(x^r - 1)$ anneau des polynômes à coefficient dans \mathbb{F}_2 Tout élément $a \in R$ peut être représenté sous forme de polynômes de degré inférieur ou égal à $r - 1$ et peut s'écrire de manière unique comme une combinaison linéaire de la forme

$$a = \sum_{i=0}^{r-1} \lambda_i x^i$$

où les $\lambda_i \in \mathbb{F}_2 \quad \forall i \in \{0, 1, \dots, r-1\}$. Cela nous donne une notion naturelle du poids de a , que nous désignons par $wt(a)$, i.e.,

$$wt(a) = |\{i \in \{0, 1, \dots, r-1\} \mid \lambda_i \neq 0\}|.$$

Remarque : La matrice de contrôle de parité d'un code de BIKE peut également être décrit de la manière suivante : $a = \sum_{i=0}^{r-1} a_i x^i \in R$ et $b = \sum_{i=0}^{r-1} b_i x^i \in R$

$$\mathbf{H} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{r-2} & a_{r-1} & b_0 & b_1 & \cdots & b_{r-2} & b_{r-1} \\ a_{r-1} & a_0 & \cdots & a_{r-3} & a_{r-2} & b_{r-1} & b_0 & \cdots & b_{r-3} & b_{r-2} \\ \vdots & & \ddots & & \vdots & \vdots & & \ddots & & \vdots \\ a_2 & a_3 & \cdots & a_0 & a_1 & b_2 & b_3 & \cdots & b_0 & b_1 \\ a_1 & a_2 & \cdots & a_{r-1} & a_0 & b_1 & b_2 & \cdots & b_{r-1} & b_0 \end{pmatrix}$$

Dans ce cas, les erreurs $e_0 = \sum_{i=0}^{r-1} e_{0,i} x^i$ et $e_1 = \sum_{i=0}^{r-1} e_{1,i} x^i$ peuvent être considérées comme des vecteurs

$$\mathbf{e}_j = (e_{j,0}, e_{j,r-1}, e_{j,r-2}, \dots, e_{j,1}) \quad \text{pour tout } j \in \{1, 2\}$$

On calcule le syndrome comme suit :

$$\mathbf{H}(\mathbf{e}_1 | \mathbf{e}_2)^\top$$

Définition : Problème 1- Décodage par syndrome (SD) [3, p.48]

Le problème de décodage de syndrome (SD) pour les codes QC-MDPC est défini comme suit :

- Soit C un (n, k) code linéaire corrigeant t erreurs sur \mathbb{F} , Soit $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$ sa matrice de contrôle de parité et soit $s \in \mathbb{F}_2^n$.
- Trouver $e \in \mathbb{F}_2^n$ telque $|e| \leq t$ et $e\mathbf{H}^T = s$.

Dans le cas générique (non QC), ceci est analogue au problème de trouver des poids décrit dans la définition suivant

Définition : (Problème 2-Codeword Finding) ([11, p.49])

Le problème de trouver un mot codé pour les QC-MDCP est défini comme suit :

- Soit C (n, k) un code linéaire corrigeant t erreurs sur \mathbb{F} , et soit $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$ sa matrice de contrôle de parité.
- Trouver $c \in \mathbb{F}_2^n$ telque $|c| = t$ et $c\mathbf{H}^T = 0$.

Dans le cas générique, ceci est analogue au problème général de décodage de la définition 3.2.1

Définition : (Problème 3 - (2,1)-QC Syndrome Decoding- (2,1)-QCSD [12 p18]

Le problème du décodage du syndrome (2, 1)-QC (QCSD) pour les codes QC-MDPC est défini comme suit :

- Soit $Cun(n,k)$ code linéaire corrigeant t erreurs sur R , l'anneau polynomial cyclique $\mathbb{F}_2[X]/(X^r - 1)$ prenons $s, h \in R$.
- Trouver $e_0, e_1 \in R$ telque $|e_0| + |e_1| \leq t$ et $e_0 + e_1 h = s$.

La notion de sécurité des messages (la non-malléabilité de la définition 3.1.13) capturée par le BIKE-KEM s'appuie sur une variante décisionnelle du problème 3 dans laquelle une attaque doit être en mesure de décider, pour un (s, h) approprié (s) , s'il existe un vecteur d'erreur qui allumettes. Les principales différences étant les contrôles de parité sur $w(h)$ et $w(c)$ et l'égalité du poids de l'erreur. Par construction, $w(h)$ est toujours impair comme le poids d'un public clés est étrange. Les problèmes 3 et 3a ci-dessous sont d'une difficulté similaire, que l'égalité soit ou non est utilisé.

Définition : Problème 3a-(2,1)-QC Décisional Parity-(2,1)-QCSD [12 p18]

Le problème de parité décisionnelle (2, 1)-QC (QCSD) pour les codes QC-MDPC est défini comme suit : Suit :

- Soit $Cun(n,k)$ code linéaire corrigeant t erreurs sur R et prenons $s, h \in R$ avec $w(h)$ impair et $w(c) + t$ pair.
- Il existe $e_0, e_1 \in R$ tel que $|e_0| + |e_1| = t$ et $e_0 + e_1 h = c$?

Définition : Problème 4-(2,1)-QC Codeword Finding-(2,1)-QCCF [12 p18]

La recherche de mots de code (2, 1)-QC (FCQC) pour les codes QC-MDCP est définie comme suit :

- Soit C un (n,k) code linéaire corrigeant t erreurs sur R et prenons $h \in R$.
- Trouver $c_0, c_1 \in R$ tel que $|c_0| + |c_1| h = 0$.

La notion d'indiscernabilité est capturée par une variante du problème 4.

Problème 4a-(2,1)-QC Decisional Balanced - (2,1)-QCCF [12 p19].

Le (2, 1)-QC Decisional Balanced Problem (QCCF) pour les codes QC-MDCP est défini comme suit :

- Soit C un (n,k) code linéaire corrigeant t erreurs sur R et prenons $h \in R$ avec $w(h)$ impair et un entier $w > 0$ tel que w soit pair et $w/2$ soit pair.
- Il existe $h_0, h_1 \in R$ tels que $|h_0| + |h_1| = w/2$ et $h_0 + h_1 h = 0$?

Selon les auteurs de la spécification, aucune des variations décrites ont un impact de la dureté des problèmes [12,p.19].

2.4.2 Les Paramètres du système de BIKE

La sélection des paramètres du système est basée sur la définition d'un niveau de sécurité λ , puis Utilisation de λ pour générer le système.

Algorithm 2 BIKE-Parameter [12, p.3]

Require: (λ) : Le niveau de sécurité

Ensure: (r) : La longueur de bloc

```

1:  $(w)$  : Le poids des lignes
2:  $(t)$  : Le rayon de décodage
3:  $(l)$  : Taille du secret partagé.
4:  $(H, K, L)$  : Un ensemble de fonctions de hachage également requises pour une utilisation
   dans le KEM).
5: On donne  $\lambda$ , Soient  $r, w, t$  et  $l$  des paramètres à valeur recommandée.  $l \leftarrow 256$ 
6: if  $\lambda == 1$  then
7:    $r = 12323, w = 142, t = 134$ 
8: end if
9: if  $\lambda == 3$  then
10:   $r = 24659, w = 206, t = 199$ 
11: else
12:   End ▷ (si  $\lambda$  n'est pas défini)
13:
14: end if
15: On sélectionne fonction de hashage :  $K : \mathbb{F}_2^{r+2l} \rightarrow \mathbb{F}_2^l$ 
16: On sélectionne de hashage  $L : \mathbb{F}_2^{2r} \rightarrow \mathbb{F}_2^l$  et on retourne  $H, K, L, r, w, t, l$ 
```

Cela définit intrinsèquement $n = 2r$ et $d = w/2$. La fonction H utilise l'AES 256 bits en mode compteur (AES256-CTR) pour étendre une graine donnée de longueur l dans une chaîne pseudo-aléatoire de bits d'une longueur $2r$ de poids st . Les fonctions K et L utilisent des fonctions de hachage SHA384 standard pour rejeter les bits au-dessus de L bits, réduisant la sortie à 256 bits.

BIKE cible les niveaux de sécurité 1 et 3 de l'appel à propositions du NIST. Ceci est essentiellement défini comme la sécurité de AES-128 et AES-192, respectivement. Pour tous λ , la taille de la clé secrète partagée (L) est définie sur 256. Les autres valeurs sont données dans le tableau des paramètres BIKE suggérés à partir de la spécification et répétés ici [12,p.9].

Security	r	w	t	DFR
Level 1	12323	142	134	2^{-128}
Level 3	24659	206	199	2^{-192}

Les algorithmes décrits ci-dessous nécessitent des fonctions pour générer uniformément et aléatoirement des vecteurs et des clés, parfois de poids spécifiques.

2.4.3 Génération de clé

Algorithm 3 Algorithme de génération de clé BIKE

3, p.4

Require: (n, w, t, l) : Les paramètres de BIKE

Ensure: (h_0, h_1, σ) : La clé privée

- 1: (h) : La clé publique
 - 2: On génère $h_0, h_1 \leftarrow R^2$ tels qu'ils soient de poids impair $|h_0| = |h_1| = w/2$
 - 3: On génère $\sigma \leftarrow \mathbb{F}_2^l$ uniformément aléatoire
 - 4: $h \leftarrow h_1 h_0^{-1}$
 - 5: on retourne $(h_0, h_1, \sigma), h$
-



L'un des défis majeurs de BIKE Key Generation est de produire h_0 tel qu'il soit inversible. L'algorithme de génération de polynômes inversibles dans BIKE est basé sur les travaux de 2020 de *Gueron et Kostic* [13]. Dans cet article, les auteurs généralisent l'algorithme k-carré rapide d'Itoh et Tsujii (algorithme ITI) [14] pour montrer que l'élévation d'un élément a à la puissance $2k$ (ou k-carré) peut être faite efficacement dans l'anneau de polynômes $R = \mathbb{F}_2[x]/(x^r - 1)$.

L'inverse de a peut aussi être calculé en utilisant **Le petit Théorème de Fermat**

2.4.4 Encodage

Algorithme 3 : L'Encapsulation de la clé de BIKE

Input : (h) : La clé publique

Sortie : (k) : L'Encapsulation de la clé

$(C = (c_0, c_1))$: Le texte chiffré

- 1 $m \leftarrow \mathbb{F}_2^l$ ▷ (m est choisi aléatoirement et uniformément)
 - 2 $(e_0, e_1) \leftarrow H(m)$
 - 3 $(c_0, c_1) \leftarrow (e_0 + e_1 h, m \oplus L(e_0, e_1))$
 - 4 $k \leftarrow K(m, C)$
 - 5 **return** (C, K)
-

La fonction d'encapsulation suit la forme de la conversion Fujisaki-Okamoto de l'algorithme 3.1.18 sans message associé pour passer la clé secrète partagée. 55 A l'étape (1) nous générons cette clé secrète. Cette clé secrète est utilisée pour produire une erreur vecteurs e_0 et e_1 ,

qui sont codés à l'aide de la clé publique pour produire la première moitié du texte chiffré $c_o = e_o + e_1 h$. La seconde moitié hache le vecteur d'erreur entier (e_o, e_1) et XORs ceci avec le message pour produire c_1 . La clé secrète est alors calculée par hachage $K = K(m, C)$.

2.4.5 Désencapsulation

algorithme 4 : Décapsulation de la clé de BIKE [3 ,p.14]

Input :

(h_o, h_1, σ) :la clé privée

$(C = (c_o, c_1))$:le texte chiffré

Output : (k) :La décapsulation de la clé

1. $(e'_o, e'_1) \leftarrow R^2$ ▷ (on fait un random aléatoire et uniform)
2. $s \leftarrow c_o h_o$ ▷ (s est le syndrome)
3. $\{(e''_o, e''_1), \perp\} \leftarrow \text{decoder}(s, h_o, h_1)$
4. **if** $(e''_o, e''_1) \leftarrow \text{decoder}(s, h_o, h_1)$ and $|(e''_o, e''_1)| = t$ **then**
5. $(e'_o, e'_1) \leftarrow (e''_o, e''_1)$
6. **end if**
7. $m' \leftarrow c_1 \oplus L(e'_o, e'_1)$
8. **if** $H(m') \neq (e'_o, e'_1)$ **then**
9. $k \leftarrow K(\sigma, C)$
10. **else** $k \leftarrow K(m', C)$
11. **end if**
12. **return** k

L'algorithme de décapsulation est basé sur la conversion **Fujisaki-Okamoto Algorithme 3.1.19**. Le calcul en (6) retrouve le message initial m qui a été encodé dans l'étape d'encapsulation (3). Nous savons que $H(m) = (e_o, e_1)$ d'Encapsulation, donc si, à l'étape (7) ceux-ci ne sont pas égaux, nous calculerons des données parasites afin que les informations ne fuient pas via des différences de temps. Dans le cas où nous atteignons l'étape (10), nous aurons une clé valide.

L'algorithme dépend d'une fonction décodeur associée qui renvoie la vecteur d'erreur (e_o, e_1) ou un jeton d'erreur (\perp) La probabilité de produire cette erreur pendant une session d'échange

de clés est le taux d'échec de décodage ou *Decoding Failure Rate* (DFR) et cela peut être vu dans le tableau des paramètres. Elle ne dépend que de l'algorithme choisi pour la fonction décodeur. Cela permet d'améliorer ultérieurement le décodeur, soit pour la vitesse, la sécurité ou pour réduire les défaillances sans affecter le schéma de sécurité périphérique.

2.4.6 fonction de décodage de BIKE

La fonction décodeur actuellement recommandée pour BIKE est le **Black-Gray-Flip** décodeur de l'algorithme 3.3.13, une amélioration de Drucker, Gueron et Kostick [drucker2019] sur l'algorithme de retournement de bits inventé à l'origine en 1962 par Gallager [R_G_Gallager]. Drucker, Gueron et Kostic montrent que ce décodeur a un taux de panne inférieur à $\frac{1}{2}^{-128}$ dans [drucker2019] c'est ce qui est requis pour être IND-CCA. C'est un algorithme à temps constant pour éviter les fuites d'informations à cause des différences de temps.

Le décodeur Black-Gray-Flip nécessite des fonctions de seuil **threshold** et **ctr**. la fonction de seuil (S, i) est pour la sélection de seuil. En général, cela dépend du syndrome S , et du numéro d'itération courant i . La fonction dans la spécification BIKE est indépendant de i et est actuellement un paramètre du niveau de sécurité λ . Il est défini pour les deux niveau de sécurité $\lambda = 1$, $\lambda = 3$ [[NGK], p.5-6].

Pour le niveau de sécurité 1, $threshold(S, i) = \max([0.0069722 \cdot S] + 13.530, 36)$

Pour le niveau de sécurité 3, $threshold(S, i) = \max([0.005265 \cdot S] + 15.2588, 52)$

Ces valeurs seuils sont basées sur la thèse de doctorat de Chaulet [chaulet2017], qui a montré que la meilleure valeur de seuil est le plus petit entier T tel que pour

$$\pi_1 = \frac{|s|+X}{td} \quad \text{et} \quad \pi_0 = \frac{w|s|-X}{(n-t)d}$$

avec

$$X = \sum_{l_{\text{impair}}} (l-1) \frac{r_l^{(w)} \binom{n-w}{t-l}}{\binom{n}{t}}$$

$$\text{alors } T = \left\lceil \frac{\log \frac{n-t}{t} + d \log \frac{1-\pi_0}{1-\pi_1}}{\log \frac{\pi_1}{\pi_0} + \log \frac{1-\pi_0}{1-\pi_1}} \right\rceil$$

qui dépend des valeurs $n = 2r$, w et t définies pour un niveau de sécurité $\lambda = 1$ ou $\lambda = 3$ et S , le poids du syndrome [aragon2020]. La fonction $ctr(H, s, j)$ compte le nombre de nombres de contrôle de parité non satisfaits de j . C'est le nombre de bits qui apparaissent dans la même position dans le syndrome s et la jème colonne de la matrice H .

Le décodeur **Black-Gray-Flip** décrit ci-dessous repose sur les deux procédures $BFIter(s, e, T, H)$ et $BFMaskedIter(s, e, mask, T, H)$. Le $BFIter$ calcule une itération de Bit-Flipping suivie de la définition du tableau de masques noir et gris. Puis dans $BFMaskedIter$, les masques noirs et gris sont utilisés pour effectuer une deuxième et une troisième itération de Bit-Flipping dans laquelle l'erreur e n'est mise à jour qu'en fonction du contenu de ces masques. Le résultat est que le Black-Gray-Flip n'a besoin que de 7 étapes pour un DFR acceptable (c'est-à-dire moins de $\frac{1}{2}^{-128}$ et un DFR inférieur en comparaison au décodeur noir-gris de Drucker, Gueron et Kostic [drucker2019] peut être réalisé en 9 étapes [drucker2019].

Algorithme 4 :Black-Gray-Flip(BGF) [drucker2019] ,p.6

Input :

- $(s \in \mathbb{F}_2^r)$:Le message reçu
- $(H \in \mathbb{F}_2^{r \times n})$:Matrice de contrôle de parité
- $(r, w, t, d = w/2, n = 2r)$:Les paramètres de BIKE
- $(NbIter)$:Le nombre d'itérations
- (τ) :La valeur de seuil **threshold**

Output :

- (e) :Le vecteur d'erreur récupéré
- (\perp) :Un jeton d'erreur possible si l'erreur ne peut pas être récupérée

1. **function** $BFIter(s, e, T, H)$
2. **for** $j = 0$ **to** $n - 1$ **do**
3. **if** $ctr(H, s, j) \geq T$ **then**
4. $e_j \leftarrow e_j \oplus 1$
5. $black_j \leftarrow 1$
6. **else if** $ctr(H, s, j) \geq T - \tau$ **then**
7. $gray_j \leftarrow 1$
8. **end if**
9. **end for**
10. **return** $e, black, gray$
11. **end function**
12. **function** $BFMaskedIter(s, e, mask, T, H)$
13. **for** $j = 0$ **to** $n - 1$ **do**
14. **if** $ctr(H, s, j) \geq T$ **then**

```

15.      $e_j \leftarrow e_j \oplus \text{mask}_j$ 
16.   end if
17. end for
18. return  $e$ 
19. end function
20.  $e \leftarrow \{0\}^n$ 
21. for  $i = 0, \dots, NbIter$  do
22.    $T \leftarrow \text{threshold}(|s + eH^T|, i)$ 
23.    $e, \text{black}, \text{gray} \leftarrow BFIter(s + eH^T, e, T, H)$ 
24.   if  $i = 1$  then
25.      $e \leftarrow BFMaskedIter(s + eH^T, e, \text{black}, (d+1)/2 + 1, H)$ 
26.      $e \leftarrow BFMaskedIter(s + eH^T, e, \text{gray}, (d+1)/2 + 1, H)$ 
27.   end if
28. end for
29. if  $s = eH^T$  then
30.   return  $e$ 
31. else
32.   return  $\perp$ 
33. end if

```

2.4.7 Les Attaques sur le système de BIKE

Comme dans le cas du McEliece Cryptosystem, la meilleure attaque sur le système BIKE est une variante de la DSI de Prange [prange23]. Soit $WFA(n, k, t)$ la quantité de travail nécessaire pour la meilleure variante ISD A qui décode une longueur n , dimension k , t -correction d'erreur code binaire. Le facteur travail peut alors s'écrire $WFA(n, k, t) = 2^{ct(1+o(1))}$, où c dépend de l'algorithme, du taux de code $R = k/n$ et du taux d'erreur t/N . Torres et Sendrier dans [70] prouvent que pour le poids $t = o(n)$, spécifiquement où $w \approx t \approx \sqrt{n}$, on a

$c = \log_2 \frac{1}{1-R}$ qui implique $WFA(n, k, t) \approx 2^{l \log_2 \frac{n}{n-k}}$ pour tous les ISD variantes [N.Aragon], p.20.

En raison de la nature quasi-cyclique de BIKE, il est plus facile d'effectuer la recherche de mots de code et décodage du syndrome par un facteur r . Pour atteindre λ_{in128} , 256 bits de classique sécurité, avec $WFA(n, k, t) = 2^{l \log_2 \frac{n}{n-k}}$, on choisit w , t et r tels que

Security	r	w	t	DFR
Niveau 1	12,323	142	134	2^{-128}
Niveau 2	24,659	206	199	2^{-192}

TABLE 2.2 – Tableau récapitulatif du taux d’échec de décodage de suivant le niveau de sécurité 1 et 3

$$\lambda \approx t - \frac{1}{2} \log_2 r \approx w - \log_2 r$$

afin que les problèmes 3a et 4a des Définitions définies dans la section 3.2 soient « assez durs » [aragon20], p.21]. Il faut aussi que r ait les propriétés suivantes :

- 2 est primitif modulo r
- r est premeir

Ce choix déjoue l’attaque au carré de [londahl2016] et évite les tentatives d’exploitation, en factorisant, la structure de $\mathbb{F}_2[x]/(x^r - 1)$ en s’assurant que $x^r - 1$ n’a que deux facteurs dont l’un est $x - 1$. Comme discuté lors de l’analyse du système Classic McEliece, toute accélération quantique connue est basée sur l’algorithme de recherche de Grover appliqué à l’ISD et est au mieux quadratique [aragon20].

2.4.8 Analyse de Sécurité

Le chiffrement BIKE utilise deux types de codes différents pour le chiffrement et le déchiffrement, ce qui lui permet d’être résistant aux attaques basées sur les failles cryptographiques courantes.

Ces limitations des valeurs de r , w et t conduisent aux résultats qui étaient en paramètre configuration de l’algorithme 1 de la sous section 3.2.10.

Une autre considération, en plus des limites de sécurité, était que le bloc longueur r est choisie telle que $|r - 2|$ (le poids de Hamming du développement binaire de $r - 2$) est petit pour que l’algorithme d’inversion (Algorithme 3.3.10) [25] soit aussi efficace que possible. Pour $r = 12,323$, $|r - 2| = 4$ et pour $r = 24659$, $|r - 2| = 5$.

2.5 Hamming Quasi - Cyclic

Le cryptosystème quasi-cyclique de Hamming est un code basé sur le IND-CPA et fonfé sur la dureté du problème du décodage de syndrome. La transformation Fujisaki-Okomoto permet la conversion en un cryptosystème de type hybride IND-CCA2. Il existe deux versions du KEM basées toutes deux sur la construction d’un KEM sécurisé à l’aide d’une combinaison de deux codes moins sécurisés.

Dans le cas de HQC, les codes sont des répétitions codes et des codes BCH qui sont combinés à l'aide du produit tensoriel et dans le cas de HQC-RMRS les codes sont les codes Reed-Muller et Reed-Solomon qui sont combinés à l'aide d'une opération de concaténation. HQC a des clés publiques de petite taille et des algorithmes de décodage efficaces basés sur des codes bien connus.

Les notations suivantes sont utilisées lors des descriptions de HQC.

Soit

$$S_w^n(\mathbb{F}_2) = \{v \in \mathbb{F}_2^n | w(v) = w\}$$

l'ensemble des vecteurs binaires de longueur n et de poids w . Soit $R = \mathbb{F}_2[x]/(x^n - 1)$ un anneau polynomial. Soit V un espace vectoriel de dimension n sur \mathbb{F}_2 alors V peut être représenté en tant que polynômes dans R ou vecteurs lignes d'une matrice par le théorème 2.5.6. Un entier premier n est primitif si le polynôme $x^n - 1/(x - 1)$ est irréductible dans R . Pour $u, v \in V$ le produit $u \cdot v$ est défini comme suit :

$$u \cdot v = w \in V,$$

Pour tout $0 \leq k < n$

$$w_k = \sum_{i+j=k \pmod n} u_i v_j,$$

qui est la multiplication polynomiale standard dans R

Définition 3.6.1 : La matrix circulante $\text{rot}(v)$ [melchor2020] Soit $v = (v_0, \dots, v_{n-1}) \in \mathbb{F}_2^n$

La matrice circulante induite par v est définie et notée comme suit :

$$\text{rot}(v) = \begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \cdots & v_2 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n-1} & v_{n-2} & \cdots & v_0 \end{pmatrix}$$

Le produit de deux éléments quelconques $u, v \in R$ peut être exprimé comme un produit vecteur matriciel usuel (ou matrice-vecteur) en utilisant l'opérateur $\text{rot}(\cdot)$ comme :

$$u \cdot v = u \times \text{rot}(v)^\top = (\text{rot}(u) \times v^\top)^\top = v \times \text{rot}(u)^\top = v \cdot u$$

C'est-à-dire que la multiplication de ces vecteurs commute.

Définition 3.6.2 : Code quasi cyclique systématique Un code quasi-cyclique systématique $[sn, n]$ d'indice s et de taux $1/s$ est un code quasi-cyclique avec une $(s-1)n \times sn$ Matrice de contrôle de parité de forme [melchor2020], p.9 :

$$\mathbf{H} = \begin{pmatrix} I_n & 0 & \cdots & 0 & A_0 \\ 0 & I_n & \cdots & 0 & A_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_n & A_{n-2} \end{pmatrix}$$

Où les A_0, \dots, A_{n-2} sont des $n \times n$ matrices circulantes.

Cette forme systématique peut être généralisée pour tenir compte des taux de différentes fractions mais les codes HQC utilisent des taux de $1/2$ et $1/3$ seulement.

Pour les références HQC à codes quasi cycliques désignent uniquement les codes des taux $1/s$. Deux formes de HQC sont à l'étude, HQC et HQC-RMRS. Le formulaire HQC se compose d'un code créé en combinant un code de répétition et un BCH à l'aide du produit Tensor. La forme HQC-RMRS consiste en la concaténation d'un Reed-Muller et un code Reed-Solomon.

Définition 3.6.3 : Code de produit tensoriel [melchor2020]

Soit C_1 et C_2 $[n_1, k_1, d_1]$ et $[n_2, k_2, d_2]$ codes linéaires sur \mathbb{F}_2 . Le code de produit tensoriel de C_1 et C_2 noté $C_1 \otimes C_2$ est défini comme l'ensemble de toutes les matrices $n_2 \times n_1$ dont les lignes sont des mots de code de C_1 et dont les colonnes sont des mots de code de C_2 . Si G_1 et G_2 sont les générateurs de C_1 et C_2 respectivement, puis

$$C_1 \otimes C_2 = \{G_1 \times G_2 \mid X \in \mathbb{F}_2^{k_1 \times k_2}\}$$

Le produit tensoriel de deux codes linéaires est un $[n_1 n_2, k_1 k_2, d_1 d_2]$ code linéaire

Définition 3.6.4 : Codes BCH utilisés dans HQC

Pour tous les entiers positifs $m \geq 3$ et $t \leq 2^{m-1}$, il existe un code BCH binaire avec les paramètres suivants [melchor2020], p.22 :

- longueur des blocs $n = 2^{m-1}$
- Nombre de chiffres de contrôle de parité $n - k \leq m\delta$, avec δ , la capacité de correction de le code et k le nombre de bits d'information, et
- la distance minimale est $d_{min} \leq 2\delta + 1$

Code	n	k	S-code	diff	n'	k'	δ
RS-1	255	207	RS-S1	175	80	32	24
RS-2	255	211	RS-S2	179	76	32	22
RS-3	255	209	RS-S3	177	78	32	23

TABLE 2.3 – Tableau des paramètres des codes de Reed-Solomon

Nous notons ce code par $RS[n, k, \delta]$. Soit α un élément primitif dans \mathbb{F}_q^m , le générateur polynômial $g(x)$ du code $RS[n, k, \delta]$ est donné par

$$g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2\delta}).$$

Les codes RS raccourcis sont ensuite construits en soustrayant les valeurs constantes de Paramètres n et k .

Dans ce tableau, les codes $RS-i[n, k, \delta]$ d'origine sont donnés avec leurs codes $RSi[n', k', \delta]$ associés $1 \leq i \leq 3$. Les valeurs de n' et k' sont générées en soustrayant diff de n et k , tandis que la valeur δ reste inchangée par ce raccourcissement.

2.5.1 Problèmes de Bases de Sécurité

La sécurité de HQC est basée sur les problèmes difficiles qui sont des variations des problèmes nous l'avons vu précédemment, à savoir le problème de décodage du syndrome (SD). Tous les problèmes difficiles décrits ci-dessous sont des variantes de ce décodage du syndrome Problème pour les codes quasi-cycliques de l'indice $s = 2, 3$ avec un taux de $1/s = 1/2, 1/3$.

Définition 3.6.5 (Distribution s-QCSD)

Pour les entiers positifs n, w et s , la distribution s-QCSD(n, w) choisit uniformément au hasard une matrice de contrôle de parité $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{(sn-n) \times sn}$ d'un code QC systématique C de l'indice s et du taux $1/s$ (voir définition 3.5.2) avec un vecteur $x = (x_0, \dots, x_{s-1}) \xleftarrow{\$} \mathbb{F}_2^{sn}$ tel que $w(x_i) = w, 0 \leq i < s$ et sorties $(\mathbf{H}, \mathbf{H}x^\top)$. Rappelez-vous de la définition 2.1.7 que w représente une norme sur R .

Après avoir mis en place la distribution, et donné $\mathbf{H}, \mathbf{H}x$, le problème est de trouver le x qui l'a généré.[melchor2020], p.10

Définition 3.6.6 : (Problème du calcul s-QCSD

Pour les entiers positifs n, w et s , la distribution s-QCSD(n, w) choisit uniformément au hasard une matrice de contrôle de parité $\mathbf{H} \xleftarrow{\$} \mathbb{F}_2^{sn-n}$ d'un code QC systématique C de l'indice s et $y \xleftarrow{\$} \mathbb{F}_2^{sn-n}$ le problème de calcul du $s - Quasi - CyclicSD(s - QCSD)(n, w)$ est de trouver $x = \{x_0, \dots, x_{s-1}\} \in \mathbb{F}_2^{sn}$ tel que $w(x_i) = w, 0 \leq i < s$ et $y = x\mathbf{H}^\top$.[melchor2020], p.10

Plus précisément, HQC utilise des codes double et triple circulaires avec une matrice génératrice systématique comme définie dans 3.6.2. Les problèmes spécifiques sont écrites ci-dessous, la matrice \mathbf{H} ci-dessus est spécifiée comme étant sous forme systématique, ceci afin de faciliter l'analyse de la sécurité. Il existe une certaine probabilité que tout code quasi-cyclique généré uniformément puisse être réduit à la forme systématique. L'analyse de sécurité est effectuée avec cette forme systématique car il s'agit de la forme d'attaque la moins gourmande en ressources informatiques.

Il existe des variantes décisionnelles des problèmes ci-dessus qui tentent de décrire les attaques basées sur l'indiscernabilité du système. La forme systématique de \mathbf{H} utilisée pour les variantes décisionnelles n'est pas celle de la définition 3.6.2. Des contraintes supplémentaires (ci-dessous) sont introduites pour éviter l'attaque de Liu, Pan et Xie selon laquelle une clé publique peut être distinguée d'une distribution uniforme en évaluant h et s à 1.

Définition 3.6.7 : (Additional HQC constraints)

Pour $b \in \{0, 1\}$ on définit un ensemble fini $\mathbb{F}_{2,b}^n = \{h \in \mathbb{F}_2^{2n \times 3n} / h(1) = b \pmod{2}\}$, i.e. un vecteur binaire de longueur n et parity b et les matrices,

$$\mathbb{F}_{2,b}^{n \times 2n} = \left\{ \mathbf{H} = (\mathbf{I}_n \text{ } rot(h)) \in \mathbb{F}_2^{n \times 2n} / h \in \mathbb{F}_{2,b}^n \right\}$$

et

$$\mathbb{F}_{2,b_1,b_2}^{2n \times 3n} = \left\{ \mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & rot(h_1) \\ \mathbf{0} & \mathbf{I}_n & rot(h_2) \end{pmatrix} \in \mathbb{F}_2^{2n \times 3n} / h_1 \in \mathbb{F}_{2,b_1}^n, h_2 \in \mathbb{F}_{2,b_2}^n \right\} [\text{melchor2020}], \text{ p.11}$$

Définition 3.6.8 : La distribution 2-QCSD (avec Parité)

Pour des entiers positifs n , w et b , la distribution 2-QCSD(n , w , b) avec parité choisit uniformément au hasard une matrice de contrôle de parité $\mathbf{H} \in \mathbb{F}_{2,b}^{n \times 2n}$. ainsi qu'un vecteur $x = (x_1, x_2)$ tel que $w(x_1) = w(x_2) = w$ et les sorties $(\mathbf{H}, \mathbf{H}x^\top)$ [melchor2020], p.11

Définition 3.6.9 : Problème Décisionnel 2-QCSD (avec Parité)

Soit $h \in \mathbb{F}_{2,b}^n$, $\mathbf{H} = (\mathbf{I}_n \text{ } rot(h))$ et $b' \equiv w + b \times w \pmod{2}$. Pour $y \in \mathbb{F}_{2,b'}^n$, le problème décisionnel 2-Quasi-Cyclic SD Problem with parity 2-DQCSD(n, w, b) demande de décider avec un avantage non négligeable si (\mathbf{H}, y) provient de la distribution 2 - QCSD(n , w , b) avec parité ou de la distribution uniforme sur $\mathbb{F}_{2,b}^{n \times 2n} \times \mathbb{F}_{2,b'}^n$. [melchor2020], p.11

Définition 3.6.10 : 3-QCSD Distribution (avec parité)

Pour les entiers positifs n , w , b_1 et b_2 , la distribution 3-QCSD(n , w , b_1 , b_2) avec parité choisit uniformément au hasard une matrice de contrôle de parité $\mathbf{H} \in \mathbb{F}_{2,b_1,b_2}^{2n \times 3n}$, ainsi qu'un vec-

teur $x = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{F}_2^{3n}$ tel que $w(x_1) = w(x_2) = w$ et les sorties $(\mathbf{H}, \mathbf{H}x^\top)$. [melchor2020], p.11

Définition 3.6.11 : Le problème Décisionnel de 3-QCSD(avec parité)

Soit $h_1 \in \mathbb{F}_{2,b_1}^n$, $h_2 \in \mathbb{F}_{2,b_2}^n$, $\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \text{rot}(h_1) \\ \mathbf{0} & \mathbf{I}_n & \text{rot}(h_2) \end{pmatrix}$, et $b'_1 \equiv w + b_1 \times w \pmod{2}$ et $b'_2 \equiv w + b_2 \times w \pmod{2}$. Pour $y \in \mathbb{F}_{2,b'}^n$, le problème décisionnel de la distribution 3-Quasi-cyclic SD avec parité 3-DQCSD(n, w, b_1, b_2) demande de décider avec un avantage non négligeable si $(\mathbf{H}, (y_1, y_2))$ provient de la distribution 3 - QCSD(n, w, b_1, b_2) avec parité ou de la distribution distribution uniforme sur $\mathbb{F}_{2,b_1,b_2}^{2n \times 3n} \times \left(\mathbb{F}_{2,b_1}^n \times \mathbb{F}_{2,b_2}^n \right)$

La sécurité IND-CPA du système HQC est basée sur la dureté supposée des problèmes 2 et 3-DQCSD des définitions 3-6-9 et 3-6-10. [melchor2020], p.12

Les codes définis pour le HQC sont des codes de Reed-Muller/Reed-Solomon concaténés et des codes BCH/répétition tensoriels. Les longueurs de ces codes $n_1 n_2$ ne sont pas premières. Cela introduit une certaine structure dans le code linéaire qui peut être exploitée par un attaquant, le code est donc étendu au-delà de la taille $(n_1 n_2)$ jusqu'au nombre premier primitif n suivant. Ensuite, les derniers $l = n - n_1 n_2$ bits du code peuvent être tronqués, si nécessaire. Cela introduit le problème de décodage suivant, sur lequel repose la sécurité du système.

Définition 3.6.12 : Décodage avec l effacements

Soit $C[n, k]$ un code QC-code généré par \mathbf{G} et $\mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e}$ pour un vecteur d'erreur aléatoire $\mathbf{e} \xleftarrow{\$} S_w^n(\mathbb{F}_2)$ (S_w^n est l'ensemble de vecteurs de longueur n de poids w). On considère la matrice $\mathbf{G}' \in \mathbb{F}_2^{k \times n'}$ et un vecteur d'erreur $\mathbf{e}' \in \mathbb{F}_2^{n'}$ obtenu par suppression des $l = n - n' \geq 1$ colonnes de \mathbf{G} et \mathbf{e} . Le problème de décodage avec l effacements consiste à récupérer $\mathbf{m} \in \mathbb{F}_2^k$ de $\mathbf{c}' = \mathbf{m}\mathbf{G}' + \mathbf{e}' \in \mathbb{F}_2^{n'}$ et $\mathbf{G}' \in \mathbb{F}_2^{k \times n'}$. [melchor2020], p.12

Puisqu'il s'agit de décoder le message codé avec moins d'informations que dans les problèmes de 2- DQCSD et 3-DQCSD des définitions 3.6.9 et 3.6.11, il est soutenu [[melchor2020], p.12] qu'il est encore plus difficile que les problèmes originaux.

La sécurité du système est IND-CPA et repose sur l'hypothèse que les problèmes 2-DQCSD et 3-DQCSD des définitions 3.5.9 et 3.5.11 avec effacements sont NP difficiles. Il existe un algorithme connu pour convertir un système IND-CPA en un système IND-CCA2. système

Instance	n_1	n_2	n	k	δ	w	$w_r = w_e$	λ	P_{fail}
hqc-128	766	31	23,869	256	57	67	77	128	$< 2^{-128}$
hqc-192	766	59	45,197	256	57	101	117	192	$< 2^{-192}$
hqc-256	766	87	69,259	256	60	133	153	256	$< 2^{-256}$

TABLE 2.4 – Les paramètres des codes hqc pour les codes BCH tensoriels avec codes à répétition

IND-CCA2 mais, en raison des erreurs de décryptage qui seraient introduites, ces algorithmes ne peuvent pas être utilisés dans le cadre de la HQC.

2.5.2 Les Paramètres du Système

Les paramètres pour le HQC sont donnés sous forme de tableau de sorte que le facteur de travail impliqué dans la meilleure attaque connue contre l'algorithme soit minimalement supérieur au niveau de sécurité souhaité.

Étant donné que le HQC est spécifié pour les codes BCH tensoriels avec des codes de répétition ainsi que pour les codes concaténés de Reed-Muller et de Reed-Muller. ainsi que pour les codes concaténés de Reed-Muller et Reed-Solomon, il existe deux tableaux différents.

Pour les codes de produits tensoriels, les paramètres suggérés sont les suivants [[melchor2020], p.37] :

Dans le cas du tenseur, nous utilisons des codes BCH $[n_1, k, \delta_1]$ abrégés et un code de répétition $\mathbf{1}_{n_2}$ combinés conformément à la définition 3.6.3 pour produire un code de produit tensoriel $C = BCH[n_1, k_1, \delta] \otimes \mathbf{1}_{n_2}$.

Les paramètres sont les suivants :

- n_1 la longueur du code BCH abrégé,
- n_2 la longueur du code de répétition,
- n le plus petit nombre premier supérieur au produit $n_1 n_2$,
- k la longueur de la clé échangée ou du message à chiffrer,
- δ la capacité de correction du code,
- w le poids de la clé secrète,
- w_r, w_e le poids des erreurs introduites lors du chiffrement ou de l'encapsulation
- λ le niveau de sécurité souhaité, et
- P_{fail} la probabilité d'un échec de décodage.

Pour les codes concaténés, les paramètres proposés sont les suivants :

Instance	n_1	n_2	n	w	$w_r = w_e$	λ	P_{fail}
hqc-RMS-128	80	256	23,533	67	77	128	$< 2^{-128}$
hqc-RMS-192	76	512	38,923	101	117	192	$< 2^{-192}$
hqc-RMS-256	78	768	59,957	133	153	256	$< 2^{-256}$

TABLE 2.5 – Tableau récapitulatif des paramètres de hqc-RMS

2.5.3 Génération de clés

Lors de la génération des clés, les paramètres ci-dessus sont utilisés pour générer des paires de clés secrètes et publiques comme suit. La génération de clés pour le KEM et le PKA est identique.

Algorithme 3.9.1 : Génération de clés HQC

[melchor2020], p.15

Input : (λ) : Le niveau de sécurité cible.

Output : (x, y) : La clé secrète

$(h, s = x + h \cdot y)$: La clé publique

1. A partir du niveau de sécurité cible λ , rechercher les paramètres associés $n, k, \delta, w, w_r, w_e$.
2. $h \xleftarrow{\$} R$ \triangleright on génère aléatoirement et uniformément h dans l'anneau des polynômes R
3. La matrice génératrice $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ de C
4. On génère $sk = (x, y) \xleftarrow{\$} R$ tel que $w(x) = w(y) = w$.
5. $pk = (h, s = x + h \cdot y)$
6. **return** (pk, sk)

2.5.4 Encryption / Encapsulation

Les opérations de cryptage et d'encapsulation sont similaires, mais les niveaux de sécurité décrits sont différents. L'algorithme de chiffrement met en place un système IND-CPA alors que le système d'encapsulation est entièrement IND-CCA2. Pour un système entièrement IND-CCA2, le système doit être indiscernable pour un attaquant ayant accès à un oracle de décryptage auquel il peut soumettre tout ce qui n'est pas le système auquel il peut soumettre tout autre chose que le texte chiffré. Plusieurs techniques existent pour convertir IND-CPA en IND-CCA2, mais comme dans le cas de BIKE, beaucoup d'entre elles ne peuvent pas être utilisées avec un taux d'échec de décodage non nul. La transformation de Fujisaki-Okamoto peut et est utilisée pour convertir le cryptage IND-CPA en KEM IND-CCA2.

Algorithme 3.9.2 :Encryption-IND-CPA HQC

[melchor2020], p.15

Input : $(h, s = x + h \cdot y)$: La clé publique

(m) : Le message

Output : $(c = u \cdot v)$: Le message chiffré

1. $\mathbf{e} \xleftarrow{\$} R$ tel que $w(e) = w_e$.
2. $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2) \xleftarrow{\$} R^2$ tel que $w(\mathbf{r}_1) = w(\mathbf{r}_2) = w_r$
3. $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$ et $\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$
4. **return** $\mathbf{c} = (\mathbf{u}, \mathbf{v})$.

L'algorithme d'encapsulation nécessite une fonction de hachage \mathcal{G} , et \mathcal{H} il s'agit généralement d'un hachage SHA ou SHAKE. Dans le cas du HQC, il est recommandé que \mathcal{G} utilise SHA3-512 et que \mathcal{H} utilise SHA-512. Il s'agit de fonctions de hachage normalisées par le NIST et les spécifications peuvent être disponibles en ligne.

Algorithme 3.9.3 : Encryption- IND-CCA2

[melchor2020], p.15

Input : $(h, s = x + h \cdot y)$: La clé publique

(l) : La longueur de la clé souhaitée

Output : $(c = u \cdot v)$: La clé encapsulée

1. $\mathbf{m} \xleftarrow{\$} \mathbb{F}_2^l$
2. $\theta \leftarrow \mathcal{G}(m)$.
3. On génère le texte chiffré $\mathbf{c} \leftarrow (u, v) = \text{Encrypt}(pk, \mathbf{m}, \theta)$ ▷ Algorithme 3.9.2
4. On détermine la clé symétrique $k \leftarrow \mathcal{H}(\mathbf{m}, c)$.
5. $d \leftarrow \mathcal{H}(m)$
6. **return** (c, \mathbf{m})

Il s'agit à nouveau d'une version simplifiée de la conversion Fujisaki-Okamoto, sans message à transmettre. Le seul problème que nous rencontrons est le partage de la clé symétrique avec l'autre système. Le système génère un message aléatoire \mathbf{m} , utilise une fonction de hachage \mathcal{G} pour le convertir en une variable θ , crypte le message à l'aide de celle-ci pour produire un vecteur d'erreur, puis utilise le texte chiffré et le message aléatoire \mathbf{m} pour produire la clé partagée. Le message est haché à l'aide d'un autre algorithme de hachage et transmis en tant qu'élément du texte chiffré afin de garantir que la clé partagée est s'assurer que la valeur de \mathbf{m} est décodée correctement.

2.5.5 Décryptage / Décapsulation

Le décryptage dans le PKE est une simple question de décodage du message reçu

Algorithme 3.9.4 : Decryption - IND-CPA HQC

[melchor2020], p.15

Input : (x, y) : La clé secrète
 $(c = u \cdot v)$: Le message chiffré
Output : (m) : Le message

1. **return** $\mathbf{m} = \text{Decode}(v - u \cdot y)$

L'algorithme de décodage à utiliser est déterminé par le type de code HQC que nous utilisons, qu'il s'agisse de codes concaténés ou de codes tensoriels.

Algorithme 3.9.5 : Décapsulation - IND-CCA2 HQC

[melchor2020], p.15

Input : (x, y) : La clé secrète
 (pk) : La clé publique
 (c, d) : Les données de la clé cryptée
Output : (k) : La clé secrète partagée

1. Decrypt $\mathbf{m}' \leftarrow \text{Decrypt}(sk, c)$ ▷ Algorithme 3.9.5
2. $\theta' \leftarrow \mathcal{G}(m')$
3. Réencryption $\mathbf{c}' = \text{Encrypt}(pk, m', \theta')$ ▷ Algorithme 3.5.2 pour vérification
4. **if** $\mathbf{c} \neq \mathbf{c}'$ or $\mathbf{d} \neq \mathcal{H}(m')$ **then**
5. abort ▷ échec
6. **end if**
7. **return** $k \leftarrow \mathcal{K}(m, c)$ ▷ la clé secrète partagée

Si l'algorithme échoue, il doit déclencher l'algorithme d'encapsulation pour produire une nouvelle clé et retransmettre.

2.5.6 Analyse de Sécurité

La sécurité IND-CPA de HQC repose essentiellement sur la dureté des problèmes 2 et 3-DQCS décrits ci-dessus (Déf. 2.1.15 et 2.1.17). Cependant, afin de contrecarrer les attaques structurelles, l'équipe de NIST a travaillé avec un code de longueur primitive n ,

de sorte que $X^n - 1$ n'ait que deux facteurs irréductibles modulo q . Mais pour les paramètres et les codes considérés dans les proposés (codes de Reed-Muller et de Reed-Solomon concaténés), le codage d'un message m a une taille de $n_1 n_2$, qui n'est évidemment pas première. Par conséquent, nous utilisons comme longueur ambiante n qui est un premier nombre premier primitif supérieur à $n_1 n_2$, et nous tronquons les derniers $' = n - n_1 n_2$ là où c'est nécessaire. Il en résulte une version légèrement modifiée du problème DQCSD, qui, selon nous, est au moins aussi difficile que les problèmes originaux.

Sur la base du résumé ci-dessus, les différents ensembles de paramètres proposés à la section 3.5.2 ont été calculés en utilisant les meilleures attaques connues afin de produire des mesures de sécurité dans la section 3.5.2 recommandées par le NIST pour $\lambda = 1, 3$ ou 5 . C'est-à-dire d'une complexité de calcul supérieure que celle de l'AES-128, de l'AES-192 et de l'AES-256, respectivement.

La meilleure attaque contre HQC a une complexité de $2^{-t \log(1-R)(1+o(1))}$ où $t = 2w$, $R = 1/2$, $o(1) = \log\left(\left(\binom{n}{w}\right)^2 / \binom{2n}{2w}\right)$ pour $2 - DQCSD$ et $t = 3w$, $R = 1/3$, $o(1) = \log\left(\left(\binom{n}{w_r}\right)^3 / \binom{3n}{3w_r}\right)$ pour $3 - DQCSD$. Pour tenir compte de l'attaque DOOM, ce facteur doit être divisé par \sqrt{n} [48, p.49]

2.5.7 Les attaques sur HQC

Outre les attaques de ISD qui ont été examinées dans les sections précédentes, les attaques contre le HQC se divisent en deux catégories : les attaques structurelles et les attaques par canaux latéraux ont une complexité de $2^{cw(1+O(1))}$ pour une certaine constante c . Une amélioration de Sendrier [61] appelée "Decoding One of Many" de Sendrier [61] appelée "Decoding One of Many" (DOOM) doit être prise en compte. Il s'agit du cas où l'attaquant dispose de plusieurs algorithmes de chiffrement mais ne doit en décoder qu'un seul. Cette traduction du problème produit un gain de $O(\sqrt{n})$ par rapport au problème standard.

Les attaques structurelles sont basées sur des attaques sur la forme du polynôme définissant l'anneau R . De telles attaques ont été étudiées par Guo, Johansson et Loundahl [34] et al pour les codes de longueur et de dimension paires et par Sendrier [61] pour réduire l'espace de recherche lorsqu'un attaquant a accès à de multiples mais ne se contente que d'en décoder un seul [61]. Ces attaques sont contrecarrées en choisissant R de telle sorte qu'il n'ait que deux facteurs irréductibles, ce qui, pour de bons choix de n , est le cas ici, puisque nous ne sommes pas en présence d'un code irréductible.

Chapitre 3

Les algorithmes d'attaques de Décodage par ensemble d'information

Contents

3.1 L'informatique quantique	65
3.1.1 Etats quantiques -Normes- Mesure -Les portes - circuits quantiques	65
3.1.2 Les opérations sur les qubits	69
3.1.3 Les n-qubits	69
3.1.4 Oracle	70
3.1.5 Transformation quantique	72
3.1.6 L'algorithme de Grover	75
3.1.7 Transformation de Grover	78
3.1.8 Etapes de l'algorithme de Grover	81
3.2 Les outils mathématiques des ISD	85
3.2.1 Le coefficient binomial et la fonction d'entropie	85
3.2.2 Codes poinçonnés	85
3.2.3 Dénombrement	87
3.2.4 Distance de Gilbert-Varshamov	87
3.2.5 Combinatoire et probabilités	88
3.2.6 Un algorithme de jointure :Merge-Join	89
3.2.7 Théorie des graphes	90
3.2.8 Produits de graphe	92
3.3 Les Algorithmes quantiques	93
3.3.1 Algorithme de Grover	93
3.3.2 Marche aléatoire sur un graphe	95
3.3.3 Algorithme de Recherche de collisions	98
3.4 Décodage par ensemble d'information (ISD)	99
3.4.1 Algorithme de Prange	99
3.5 Squelette d'un algorithme ISD	102
3.5.1 Décodage par ensemble d'information quantique	104
3.6 Les algorithmes d'attaques ISD et ses dérivés	105

3.6.1	Les Algorithmes de Prange et de Bernstein	106
3.6.2	Les algorithmes ISD avancés avec technique de représentation	108
3.6.3	ISD avancé 2 avec technique de représentation étendue	118
3.6.4	Algorithme MMT* classique	118
3.6.5	Algorithme de $MMT^{\#}$	120
3.6.6	Les recherches de voisins proches : Algorithme de May-Ozerov	121
3.6.7	Version classique	123

Nous avons vu précédemment que les cryptosystèmes post-quantiques PQCs admis au 3ème tour de NIST ont leur sécurité basé principalement sur le problème du décodage par syndrome SDP qui est un problème NP-complet ce qui leur permet de résister à la puissance de calcul de la machine quantique. Mais cette résistance est faillible devant l'algorithme classique du décodage par ensemble d'information "Information Set Decoding algorithm"(ISD algorithm) proposé par Prange [Prange23] et ses dérivés ultérieurs [Over22],[Mo19],[Berst5]. Dans ce chapitre nous évoquerons les algorithmes décodage par ensemble d'information et leur dérivés, nous allons décrire les meilleures attaques et donner des tableaux comparatifs suivant les cryptosystèmes étudiés.

Avant d'entrer dans le vif du sujet nous allons définir dans la section suivante les outils mathématiques utilisés dans les algorithmes ISD.

3.1 L'informatique quantique

Définition 21 *L'informatique quantique est un domaine multidisciplinaire comprenant des aspects de l'informatique, de la physique et des mathématiques qui utilise la mécanique quantique pour résoudre des problèmes complexes plus rapidement que sur des ordinateurs classiques. Le domaine de l'informatique quantique comprend la recherche sur le matériel et le développement d'applications. Les ordinateurs quantiques sont capables de résoudre certains types de problèmes plus rapidement que les ordinateurs classiques en tirant parti des effets de la mécanique quantique comme la superposition et l'interférence quantique.[ama]*

Parmi les applications pour lesquelles les ordinateurs quantiques peuvent fournir un tel gain de vitesse figurent le machine learning (ML) l'optimisation et la simulation de systèmes physiques. Les cas d'utilisation éventuels pourraient être l'optimisation de portefeuilles dans le domaine financier ou la simulation de systèmes chimiques, ce qui permettrait de résoudre des problèmes actuellement impossibles à résoudre, même par les superordinateurs les plus puissants du marché.

On se place dans l'espace de Hilbert complexe H de dimension 2 et on note $|0\rangle$ et $|1\rangle$ les vecteurs de la base canonique.[Arnaud]

3.1.1 Etats quantiques -Normes- Mesure -Les portes - circuits quantiques

Etats quantiques

Le bit est l'unité de base de l'information classique. Il peut être dans deux états différents : 0 ou 1. **L'unité de base de l'information quantique s'appelle le qubit.** Le propre des états quantiques est que toute combinaison linéaire de deux états quantiques est encore un état quantique, ce que l'on appelle principe de superposition.

Ainsi, un qubit peut être $|0\rangle$ on lit "ket 0", $|1\rangle$ on lit "ket 1" ou bien l'état superposé $\alpha|0\rangle + \beta|1\rangle$ avec $\|\psi\| = |\alpha|^2 + |\beta|^2 = 1$ avec $\alpha, \beta \in \mathbb{C}$ s'appellent respectivement les **amplitudes** ou **phases** de $|0\rangle$ et de $|1\rangle$

Représentation vectorielle

Sur le plan $|0\rangle$ correspond au vecteur de coordonnées $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $|1\rangle$ correspond au vecteur de coordonnées $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Norme

Soit le qubit $\psi = \alpha|0\rangle + \beta|1\rangle$ est de norme 1 si $|\alpha|^2 + |\beta|^2 = 1$.

Exemple : $\psi = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ est de norme 1 car $|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} + \frac{1}{2} = 1$.

Mesure et Probabilité

Soit l'état quantique $\psi = \alpha|0\rangle + \beta|1\rangle$ de norme 1.

On définit la mesure de ψ la fonction noté $measure(|\psi\rangle)$ qui renvoie 0 avec une probabilité de $|\alpha|^2$ et 1 avec une probabilité de $|\beta|^2$.

Exemple : Donnons une mesure de l'état quantique $|\psi\rangle = \frac{1-i}{\sqrt{3}}|0\rangle + \frac{1+2i}{\sqrt{15}}|1\rangle$

Vérifions s'il est de norme 1

On a $|\alpha|^2 = |\frac{1-i}{\sqrt{3}}|^2 = \frac{2}{3}$ et $|\beta|^2 = |\frac{1+2i}{\sqrt{15}}|^2 = \frac{1}{3} \Rightarrow |\alpha|^2 + |\beta|^2 = 1$ alors $|\psi\rangle$ est de norme 1 donc on a

$$Measure(\psi) = \begin{cases} 0 & \text{avec probabilité de } \frac{2}{3}. \\ 1 & \text{avec probabilité de } \frac{1}{3}. \end{cases}$$

Les portes quantiques

Un ordinateur quantique est fait à base de qubits qui subissent des transformations dans un circuit quantique et celles-ci sont réalisées par des portes quantiques.

Une porte quantique est la transformation $|\psi\rangle \rightarrow A|\psi\rangle$ où A est une matrice unitaire.

On distingue des portes quantiques élémentaires agissant sur un seul qubit :

Les portes de Pauli sont les suivantes X/ Not, Y et Z. Leur action sur un qubit est décrit dans le tableau 5.2.

Exemple : Soit $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ alors la transformation de la porte X sur ψ est $X|\psi\rangle =$

$X: \begin{cases} 0\rangle \mapsto 1\rangle \\ 1\rangle \mapsto 0\rangle \end{cases}$	$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
$Y: \begin{cases} 0\rangle \mapsto i 1\rangle \\ 1\rangle \mapsto -i 0\rangle \end{cases}$	$Y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
$Y: \begin{cases} 0\rangle \mapsto i 1\rangle \\ 1\rangle \mapsto -i 0\rangle \end{cases}$	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
$Z: \begin{cases} 0\rangle \mapsto 0\rangle \\ 1\rangle \mapsto - 1\rangle \end{cases}$	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

TABLE 3.1 – Les portes de Pauli
[Rap]

$H: \begin{cases} 0\rangle \mapsto 1\rangle \\ 1\rangle \mapsto 0\rangle \end{cases}$	$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
---	---

TABLE 3.2 – Les porte de Hadamard
[Rap]

$$\frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle.$$

La porte de Hadamard est appelée aussi la porte **H** table 5.3.

$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ et $H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ soit l'état quantique $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ l'action de la porte H sur $|\psi\rangle$ est notée $H|\psi\rangle = \alpha(H|0\rangle) + \beta(H|1\rangle)$.

Les 2-qubits

En informatique quantique un 2-qubits est la superposition de quatre états fondamentaux $|\psi\rangle = \alpha|0 \cdot 0\rangle + \beta|0 \cdot 1\rangle + \delta|1 \cdot 0\rangle + \lambda|1 \cdot 1\rangle$ avec $\alpha, \beta, \delta, \lambda \in \mathbb{C}$ tel que $|\alpha|^2 + |\beta|^2 + |\delta|^2 + |\lambda|^2 = 1$.

La mesure de 2-qubits renvoie deux bits classiques

$$Measurement(\psi) = \begin{cases} 0.0 & \text{avec proba de } |\alpha|^2. \\ 0.1 & \dots |\beta|^2. \\ 1.0 & \dots |\delta|^2. \\ 1.1 & \dots |\lambda|^2. \end{cases}$$

Notations :

$$|0.0\rangle = |0\rangle \cdot |0\rangle.$$

$$|0.1\rangle = |0\rangle \cdot |1\rangle.$$

$$|1.0\rangle = |1\rangle \cdot |0\rangle.$$

$$|1.1\rangle = |1\rangle \cdot |1\rangle.$$

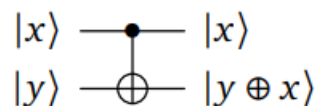
$$|0.1\rangle = |0\rangle \cdot |1\rangle = |0\rangle \otimes |1\rangle.$$

Porte quantiques à deux entrées

La porte C-NOT est une superposition d'un contrôleur et d'une porte NOT dont la sortie de la première ligne reste inchangée et celle de la seconde ligne change si sa valeur d'entrée est $|1\rangle$

$$\begin{cases} |0.0\rangle \mapsto |0.0\rangle \\ |0.1\rangle \mapsto |0.1\rangle \\ |1.0\rangle \mapsto |1.1\rangle \\ |1.1\rangle \mapsto |1.0\rangle \end{cases}$$

Cependant pour un état quantique quelconque $|\psi\rangle = \alpha|0.0\rangle + \beta|0.1\rangle + \delta|1.0\rangle + \lambda|1.1\rangle$ l'action de la porte C-NOT est notée : $C-NOT|\psi\rangle = \alpha|0.0\rangle + \beta|0.1\rangle + \lambda|1.0\rangle + \delta|1.1\rangle$



La porte SWAP échange deux qubits d'entrée



La porte FANOUT transforme un 1-qubit en 2-qubit. Dans un circuit quantique, cela permet d'augmenter le nombre de lignes quantiques



3.1.2 Les opérations sur les qubits

Addition sur 1-qubit

Soient $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ et $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$

On a $|\psi_1\rangle + |\psi_2\rangle = (\alpha_1 + \alpha_2)|0\rangle + (\beta_1 + \beta_2)|1\rangle$

Addition sur 2-qubit

Soient $|\psi_1\rangle = \alpha_1|0.0\rangle + \beta_1|0.1\rangle + \delta_1|1.0\rangle + \lambda_1|1.1\rangle$ et $|\psi_2\rangle = \alpha_2|0.0\rangle + \beta_2|0.1\rangle + \delta_2|1.0\rangle + \lambda_2|1.1\rangle$

On a $|\psi_1\rangle + |\psi_2\rangle = (\alpha_1 + \alpha_2)|0.0\rangle + (\beta_1 + \beta_2)|0.1\rangle + (\delta_1 + \delta_2)|1.0\rangle + (\lambda_1 + \lambda_2)|1.1\rangle$

Multiplication

Soit les qubits $|i\rangle$ et $|j\rangle$ la multiplication de deux 1-qubit donne 2-qubit

$$|i\rangle \cdot |j\rangle = |i \cdot j\rangle$$

3.1.3 Les n-qubits

Un n-qubit est la superposition de n états quantiques de base.

Soit $|\psi\rangle$ un n-qubit alors il s'écrit

$$|\psi\rangle = \alpha_0|0 \cdots 0\rangle + \alpha_1|0 \cdots 1\rangle + \cdots + \alpha_{2^n-1}|1 \cdots 1\rangle$$

Exemple : 3-qubit

$$|\psi\rangle = \alpha_0|0.0.0\rangle + \alpha_1|0.0.1\rangle + \alpha_2|0.1.0\rangle + \alpha_3|1.0.0\rangle + \alpha_4|0.1.1\rangle + \alpha_5|1.0.1\rangle + \alpha_6|1.1.0\rangle + \alpha_7|1.1.1\rangle$$

On constate nous avons 8 termes pour 3-qubits pour un état quantique alors qu'informatique classique travailler avec 3 bits revient à faire 8 entrées possibles d'où la puissance de la machine quantique.

Mesure d'un n-qubit

Soit $|\psi\rangle = \alpha_0|0 \cdots 0\rangle + \alpha_1|0 \cdots 1\rangle + \cdots + \alpha_{2^n-1}|1 \cdots 1\rangle$ un n-qubits telle que

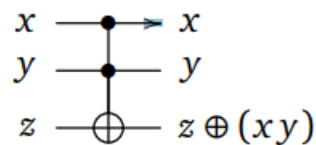
$$\|\psi\| = \sqrt{|\alpha_0|^2 + \cdots + |\alpha_{2^n-1}|^2} = 1$$

$$measure(\psi) \left\{ \begin{array}{lll} 0.0 \cdots 0 & \text{avec une proba} & |\alpha_0|^2 \\ 0.0 \cdots 1 & \text{avec une proba} & |\alpha_1|^2 \\ & \vdots & \dots \\ 1.1 \cdots 1 & \text{avec une proba} & |\alpha_{2^n-1}|^2 \end{array} \right.$$

Les portes à Trois entrées

Porte de Toffoli (CCNOT) : La porte de Toffoli est similaire à une porte CNOT mais avec trois lignes. Si les deux premiers qubits sont $|1\rangle$, alors on applique une porte **X** (c'est -à-dire NOT) au troisième qubit.

Voici l'action d'une porte de Toffoli lorsque x, y, z sont des bits 0 ou 1 (noter que $xy = 1$ si et seulement si $x = 1$ et $y = 1$ et alors $1 \oplus z = \text{NOT}(z)$)

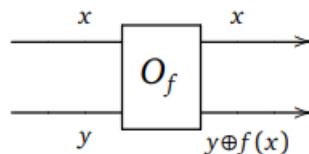


3.1.4 Oracle

Soit le groupe $(\mathbb{Z}/n\mathbb{Z}, +)$ correspond à l'ensemble des entiers modulo n représenté par $\{0, 1, \dots, n-1\}$ pour $n = 2$ on a le groupe $(\mathbb{Z}/2\mathbb{Z})$ qui est l'ensemble $\{0, 1\}$, muni de l'addition binaire notée \oplus qui vérifie $1 \oplus 1 = 0$, ce qui est cohérent car $1 + 1 \equiv 0 \pmod{2}$.

Définition 22 Nous allons associer à une fonction f un oracle. L'oracle d'une fonction f est un circuit quantique dont on explicite seulement l'entrée et la sortie (qui dépend de f). C'est une sorte de boîte noire, car nous n'avons pas besoin de connaître les détails du circuit qui réalise un oracle.

Cas $f : \mathbb{Z}/2\mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$: la transformation effectuée par un oracle, lorsque les entrées sont des bits classiques 0 ou 1.



Il y a deux lignes pour l'entrée de l'oracle et deux lignes pour la sortie. La première sortie laisse la première entrée inchangée. Pour la seconde sortie : si x et y sont 0 ou 1 alors la seconde sortie est $y \oplus f(x)$; c'est donc y si $f(x) = 0$ et $\text{NOT}(y)$ si $f(x) = 1$.

Ainsi l'oracle associé à f fournit une fonction

$$\begin{aligned} F : \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} &\rightarrow \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \\ (x, y) &\mapsto (x, y \oplus f(x)) \end{aligned}$$

Nous verrons plus tard comment cela définit naturellement une transformation quantique sur les 2-qubits. Pour l'instant nous généralisons l'oracle au cas d'autres fonctions.

Cas $f : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$:

Cette situation correspondra à l'algorithme de Grover. On fixe $n \geq 2$ et on considère une fonction quelconque $f : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$ que l'on peut aussi voir comme une fonction $f : \{0, 1, \dots, n-1\} \rightarrow \{0, 1\}$. La transformation de l'oracle, pour $x \in \mathbb{Z}/n\mathbb{Z}$ et $y \in \mathbb{Z}/2\mathbb{Z}$, renvoie une nouvelle fois x (élément de $\mathbb{Z}/n\mathbb{Z}$) et $y \oplus f(x)$ (élément de $\mathbb{Z}/2\mathbb{Z}$)

On obtient ainsi :

$$\begin{aligned} F : \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} &\rightarrow \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \\ (x, y) &\mapsto (x, y \oplus f(x)) \end{aligned}$$

Exemple

Fixons $l \in \{0, \dots, n-1\}$ un entier et $f : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$ tel que $f(x) = 0$ pour tout x , sauf $f(l) = 1$.

Alors :

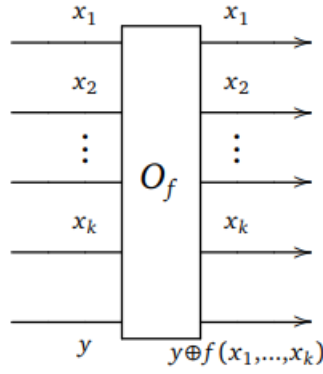
- pour $x \neq l$ et $y = 0$ on a $y \oplus f(x) = 0$,
- pour $x \neq l$ et $y = 1$ on a $y \oplus f(x) = 1$,
- pour $x = l$ et $y = 0$ on a $y \oplus f(x) = 1$,
- pour $x = l$ et $y = 1$ on a $y \oplus f(x) = 1 \oplus 1 = 0$

Cas $f : (\mathbb{Z}/n\mathbb{Z})^k \rightarrow \mathbb{Z}/2\mathbb{Z}$

Cette situation correspondra à l'algorithme de **Deutsch-Jozsa**. On fixe $k \geq 1$ et on considère une fonction quelconque $f : (\mathbb{Z}/n\mathbb{Z})^k \rightarrow \mathbb{Z}/2\mathbb{Z}$ que l'on peut aussi voir comme une fonction $f : \{0, 1\}^k \rightarrow \{0, 1\}$

que l'on peut aussi voir comme une fonction $f : \{0, 1\}^k \rightarrow \{0, 1\}$.

. La transformation de l'oracle, pour $x = (x_1, \dots, x_k) \in (\mathbb{Z}/n\mathbb{Z})^k$ et $y \in \mathbb{Z}/2\mathbb{Z}$, renvoie $x = (x_1, \dots, x_k)$ (élément de $(\mathbb{Z}/n\mathbb{Z})^k$) et $y \oplus f(x)$ (élément de $\mathbb{Z}/2\mathbb{Z}$)



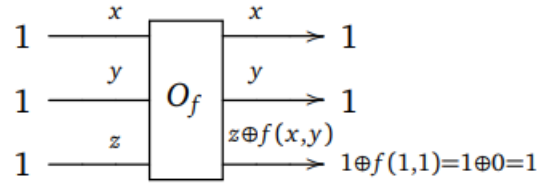
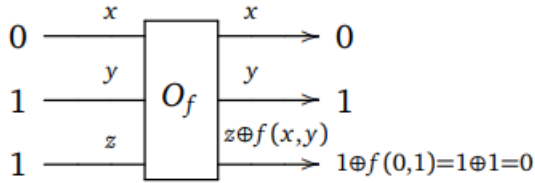
On obtient ainsi :

$$\mathbf{F} : (\mathbb{Z}/2\mathbb{Z})^k \times \mathbb{Z}/2\mathbb{Z} \rightarrow (\mathbb{Z}/2\mathbb{Z})^k \times \mathbb{Z}/2\mathbb{Z}$$

$$(x_1, \dots, x_k, y) \mapsto (x_1, \dots, x_k, y \oplus f(x_1, \dots, x_k))$$

Exemple :

On considère $f : (\mathbb{Z}/2\mathbb{Z})^2 \rightarrow (\mathbb{Z}/2\mathbb{Z})$ définie par $f(x, y) = x \text{ XOR } y$. Voici quelques exemples d'action de l'oracle :



Autrement dit $\mathbf{F}(0, 1, 1) = (0, 1, 0)$ et $\mathbf{F}(1, 1, 1) = (1, 1, 1)$

On pourrait généraliser l'oracle au cas d'une fonction $f : E \rightarrow E'$ pour lequel l'oracle associé serait une fonction $F : E \times E' \rightarrow E \times E'$ défini par $F(x, y) = (x, y \oplus f(x))$ où « \oplus » est une addition dans E'

3.1.5 Transformation quantique

Considérons le cas d'une fonction $f : (\mathbb{Z}/2\mathbb{Z})^2 \rightarrow (\mathbb{Z}/2\mathbb{Z})$. L'oracle fournit une fonction $\mathbf{F} : (\mathbb{Z}/2\mathbb{Z})^{k+1} \rightarrow (\mathbb{Z}/2\mathbb{Z})^{k+1}$ en ayant considéré $(\mathbb{Z}/2\mathbb{Z})^k \times \mathbb{Z}/2\mathbb{Z} = (\mathbb{Z}/2\mathbb{Z})^{k+1}$. Voyons la transformation quantique associée sur les $(k+1)$ -qubits.

Les $(k+1)$ -qubits sont engendrés par la base canonique formée des 2^{k+1} qubits de base :

$$|0.0 \dots 0\rangle \quad |0.0 \dots 1\rangle \quad \dots \quad |1.1 \dots 1\rangle$$

La fonction F (définie sur des $(k+1)$ -bits) s'étend naturellement en une fonction \tilde{F} sur les vecteurs de la base des $(k+1)$ -qubits :

$$\begin{aligned} |e_0\rangle = |0.0 \cdots 0\rangle & \xrightarrow{\tilde{F}} |F(0,0,\dots,1)\rangle = |f_0\rangle \\ |e_1\rangle = |0.0 \cdots 1\rangle & \xrightarrow{\tilde{F}} |F(0,0,\dots,1)\rangle = |f_1\rangle \\ \dots & \xrightarrow{\tilde{F}} \\ |e_{2^{k+1}-1}\rangle = |1.1 \cdots 1\rangle & \xrightarrow{\tilde{F}} |F(1,1,\dots,1)\rangle = |f_{2^{k+1}-1}\rangle \end{aligned}$$

Maintenant que \tilde{F} est définie sur les vecteurs de la base par la relation $\tilde{F}(|e_i\rangle) = |F(e_i)\rangle = |f_i\rangle$, elle s'étend par linéarité à tous les $(k+1)$ -qubits. Ainsi on obtient

$$\tilde{F} : \mathbb{C}^{2^{k+1}} \rightarrow \mathbb{C}^{2^{k+1}}$$

et pour un $(k+1)$ -qubits

$$|\psi\rangle = \sum_{i=0}^{2^{k+1}-1} \alpha_i |e_i\rangle \text{ avec } \alpha_i \in \mathbb{C}, \text{ on obtient le } (k+1)\text{-qubit :}$$

$$\tilde{F}(|\psi\rangle) = \alpha_i |f_i\rangle$$

Comme F est bijective alors \tilde{F} envoie l'ensemble des vecteurs de la base canonique sur ces mêmes vecteurs de la base canonique (autrement dit \tilde{F} permute les vecteurs de la base).

Notation : On fixe un entier $n \geq 1$. Soit $0 \leq k \leq 2^n - 1$. Notons \underline{k} l'écriture binaire de l'entier k sur n bits. L'écriture entière \underline{k} désigne le n -qubit de la base canonique associée à l'écriture binaire k .

Exemple : Pour $n = 1$ il y a seulement deux qubits de base $|\underline{0}\rangle = |0\rangle$ et $|\underline{1}\rangle = |1\rangle$

Pour $n = 2$ $|\underline{0}\rangle = |0.0\rangle$, $|\underline{1}\rangle = |0.1\rangle$, $|\underline{2}\rangle = |1.0\rangle$, $|\underline{3}\rangle = |1.1\rangle$

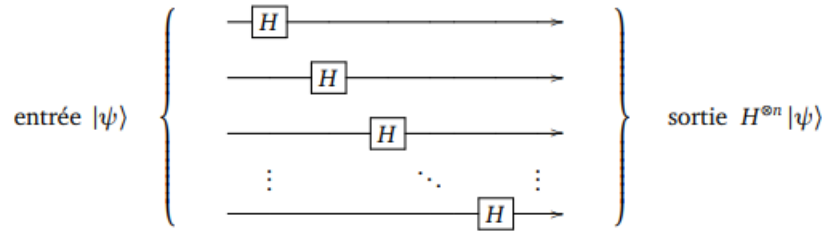
La transformation de Hadamard

On rappelle que la porte de Hadamard est définie pour les 1-qubits par la formule :

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ et } H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

La transformation de Hadamard d'un n -qubit $|\psi\rangle$ est l'application d'une porte de Hadamard sur chacun des 1-qubits le constituant. On note cette transformation $H^{\otimes n}$.

Le circuit est simplement composé de n lignes, avec une porte de Hadamard par ligne (l'ordre de ces portes n'a pas d'importance).



Exemple : pour $n=2$

$$|\underline{0}\rangle = |0.0\rangle \xrightarrow{H^{\otimes 2}} \frac{1}{2}(|0+1\rangle|0+1\rangle) = \frac{1}{2}(|0.0\rangle + |0.1\rangle + |1.0\rangle + |1.1\rangle) = \frac{1}{2}(|\underline{0}\rangle + |\underline{1}\rangle + |\underline{2}\rangle + |\underline{3}\rangle)$$

$$|\underline{1}\rangle = |0.1\rangle \xrightarrow{H^{\otimes 2}} \frac{1}{2}(|0+1\rangle|0-1\rangle) = \frac{1}{2}(|0.0\rangle - |0.1\rangle + |1.0\rangle - |1.1\rangle) = \frac{1}{2}(|\underline{0}\rangle - |\underline{1}\rangle + |\underline{2}\rangle - |\underline{3}\rangle)$$

Formule de la transformation de Hadamard

Soit $n \geq 1$, en se basant de ce qui précède on a :

$$\text{pour } n = 2, H|\underline{0}\rangle = \frac{1}{2}(|\underline{0}\rangle + |\underline{1}\rangle + |\underline{2}\rangle + |\underline{3}\rangle)$$

$$H|\underline{0}\rangle = \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} |\underline{l}\rangle$$

D'une autre manière avec on a :

$$\mathbf{H}|\underline{0}\rangle = \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} |\underline{l}\rangle$$

La formule générale est donnée par la proposition suivante :

Proposition 8 Pour $0 \leq k \leq 2^n - 1$, on a :

$$\mathbf{H}^{\otimes n}|\underline{k}\rangle = \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} (-1)^{\underline{k} \odot \underline{l}} |\underline{l}\rangle$$

Notation : Pour l'écriture binaire $\underline{k} = k_1.k_2 \cdots k_n$ et l'écriture binaire $\underline{l} = l_1.l_2 \cdots l_n$ (avec $k_i, l_i \in \{0, 1\}$)

$$\underline{k} \odot \underline{l} = k_1 l_1 \oplus k_2 l_2 \oplus \cdots \oplus k_n l_n \in \{0, 1\}.$$

C'est comme un produit scalaire modulo 2

Exemple :

Soit $n = 3$ et $|\underline{5}\rangle = |1.0.1\rangle$, alors un calcul direct donne :

$$\begin{aligned}
\mathbf{H}^{\oplus 3}|\underline{5}\rangle &= \mathbf{H}^{\oplus 3}|1.0.1\rangle \\
&= \frac{1}{2\sqrt{2}}|(0-1).(0+1).(0-1)\rangle \\
&= \frac{1}{2\sqrt{2}}(|(0.0.0)\rangle - |(0.0.1)\rangle + |(0.1.0)\rangle - |(0.1.1)\rangle - |(1.0.0)\rangle + |(1.0.1)\rangle - |(1.1.0)\rangle + |(1.1.1)\rangle)
\end{aligned}$$

Dans ce qui précède nous avons vu les éléments de base de l'informatique quantique et qui forment les principaux constituants du circuit de l'algorithme de Grover.

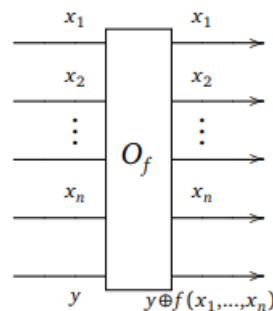
3.1.6 L'algorithme de Grover

L'algorithme de Grover est basé sur la recherche d'un élément sur une liste de taille N . Il est d'une complexité \sqrt{N} et d'une probabilité $1 - \frac{4}{N}$ de donner le bon élément. Parler d'un circuit de Grover revient à décrire, les entrées, l'oracle et la transformation de Grover et l'ensemble des portes quantiques qui le composent.

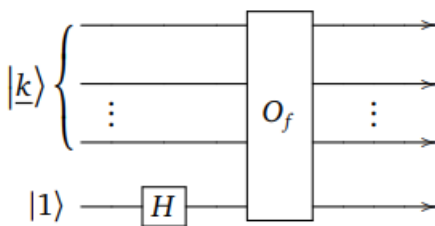
Oracle de Grover

On se place dans le cas où N est une puissance de 2 : $N = 2^n$. On se rappelle que pour $0 \leq k \leq N-1$, alors \underline{k} est l'écriture binaire de k sur n bits. Ainsi $|k\rangle$, pour $k = 0, \dots, 2^n - 1$, désigne les n -qubits de la base canonique : $|\underline{0}\rangle = |0.0 \dots 0\rangle, |\underline{1}\rangle = |0.0 \dots 1\rangle, \dots, |2^n - 1\rangle = |1.1 \dots 1\rangle$.

Nous allons utiliser l'oracle O_f associé à la fonction f . Pour $x \in \mathbb{Z}/N\mathbb{Z}$ et $y \in \mathbb{Z}/2\mathbb{Z}$, l'oracle réalise une fonction $\mathbf{F}(x, y) = (x, y \oplus f(x))$. On préfère écrire l'entier x à l'aide de son écriture binaire $\underline{x} = x_1.x_2 \dots x_n$, ce qui permet de récrire la fonction \mathbf{F} sous la forme $\mathbf{F}(x, y) = (x_1, \dots, x_n, y \oplus f(x_1, \dots, x_n))$ [Arnaud]-p4.



En appliquant $|\underline{x}\rangle = |\underline{0}\rangle$ et posant $y \oplus f(x_1, \dots, x_n) = 1 \Rightarrow y = 1$



L'entrée $|1\rangle$ est d'une porte de hadamard et de l'oracle d'où la sortie $|s\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \oplus f(k) = (-1)^{f(k)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

$$|s\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{si } k \neq k_o \\ -\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{si } k = k_o \end{cases}$$

L'oracle nous permet de faire une remarque : le signe du terme au rang k_o est négatif "-"

Ainsi nous allons procéder à la détection du rang k_o .

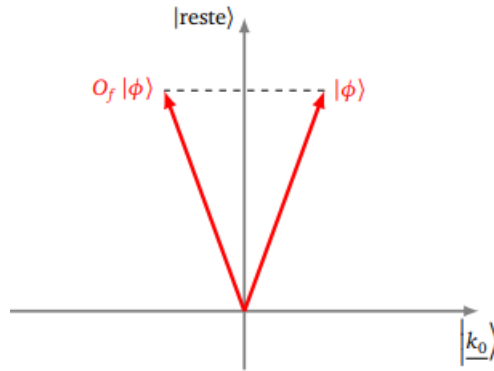
Mais celle-ci se fera par des transformations géométriques.

Symétrie de l'oracle

L'oracle transforme $|k_o\rangle$ en $-|k_o\rangle$ et laissant inchangé les $k \neq k_o$. Soit $|\phi\rangle$ un n-qubit en isolant k_o on a

$$|\phi\rangle = \alpha|\underline{k_o}\rangle + \sum_{k \neq k_o} \alpha_k |\underline{k}\rangle$$

L'action de l'oracle O_f sur $|\phi\rangle$ est : $O_f|\phi\rangle = -\alpha|\underline{k_o}\rangle + \sum_{k \neq k_o} \alpha_k |\underline{k}\rangle$



Ainsi $O_f|\phi\rangle$ est le symétrique de $|\phi\rangle$ par rapport à l'axe $|reste\rangle$ qui sont l'ensemble des termes autres que $|k_o\rangle$.

Nous allons définir cette symétrie axiale par récurrence induite par l'oracle

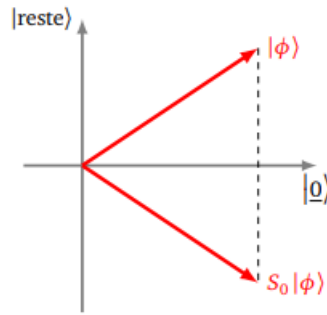
Soit S_o , la transformation définie sur les n-qubits de la base canonique par :

$$\begin{cases} |\underline{0}\rangle \xrightarrow{S_o} |\underline{0}\rangle \\ |\underline{k}\rangle \xrightarrow{S_o} -|\underline{k}\rangle \text{ si } k \neq 0 \end{cases}$$

On voit c'est seulement le qubit $|\underline{0}\rangle$ qui ne change pas. Par exemple $n = 2$, on a $S_o|0.0\rangle = |0.0\rangle, S_o|0.1\rangle = -|0.1\rangle, S_o|1.0\rangle = -|1.0\rangle, S_o|1.1\rangle = -|1.1\rangle$.

Comme on étend S_o par linéarité à tous les n-qubits.

$$S_o(\alpha|0.0\rangle + \beta|0.1\rangle + \gamma|1.0\rangle + \delta|1.1\rangle) = \alpha|0.0\rangle - \beta|0.1\rangle - \gamma|1.0\rangle - \delta|1.1\rangle$$



Lemme suivant donne l'expression algébrique de S_0

Lemme 1 $S_0 = 2|\underline{0}\rangle\langle\underline{0}| - I$

I désigne l'application identité. Ainsi cette formule signifie que pour un qubit $|\phi\rangle$ on a :

$$S_0|\phi\rangle = 2|\underline{0}\rangle\langle\underline{0}|\phi\rangle - |\phi\rangle$$

L'écriture $|\underline{0}\rangle\langle\underline{0}|\phi\rangle$ est bien qubit car $\langle\underline{0}|\phi\rangle$ est un scalaire (i.e un nombre complexe)

Soit la transformation S_ψ avec $|\psi\rangle$ un n-qubit de norme 1. Nous allons montrer que

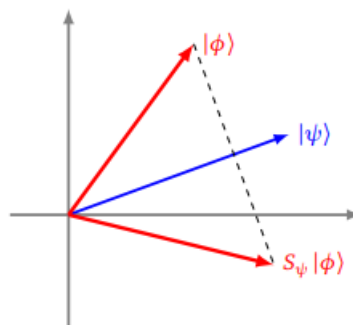
$$S_\psi = 2|\psi\rangle\langle\psi| - I$$

Soit $|\phi\rangle$ un n-qubit quelconque on a $S_\psi|\phi\rangle = (2|\psi\rangle\langle\psi| - I)|\phi\rangle$

$$S_\psi|\phi\rangle = 2|\psi\rangle\langle\psi|\phi\rangle - |\phi\rangle$$

Cette transformation vérifie :

$$\begin{cases} |\psi\rangle \xrightarrow{S_\psi} |\psi\rangle \\ |\phi\rangle \xrightarrow{S_\psi} -|\phi\rangle \text{ si } |\phi\rangle \text{ est orthogonal à } |\psi\rangle \end{cases}$$



Géométriquement S_ψ est une symétrie par rapport à l'axe dirigé par $|\psi\rangle$.

Transformation S_{ψ_H}

Notons $|\psi_H\rangle$ le n-qubit formé par la somme de tous les qubits de la base canonique :

$$|\psi_H\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle$$

ce qubit $|\psi_H\rangle$ est aussi l'image du qubit $|0\dots 0\rangle$ par transformation de Hadamard :

$$|\psi_H\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{0}\rangle$$

Proposition 9 La transformation de $|\psi_H\rangle$ est définie par l'une des caractérisations équivalentes suivantes :

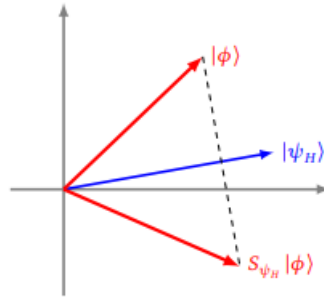
(i) $S_{\psi_H} = 2|\psi_H\rangle\langle\psi_H| - I$, c'est-à-dire $S_{\psi_H}|\phi\rangle = 2|\psi_H\rangle\langle\psi_H|\phi\rangle - |\phi\rangle$ pour tout qubit $|\phi\rangle$

(ii) $\begin{cases} |\psi_H\rangle \xrightarrow{S_{\psi_H}} |\psi_H\rangle \\ |\phi\rangle \xrightarrow{S_{\psi_H}} -|\phi\rangle \text{ si } |\phi\rangle \text{ est orthogonal à } |\psi_H\rangle \end{cases}$

(iii) $S_{\psi_H} = H^{\otimes n} \cdot S_0 \cdot H^{\otimes n}$

(iv) S_{ψ_H} a pour matrice

$$\frac{2}{2^n}U - I \text{ où } U = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \ddots & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & \dots & 1 & 1 \end{pmatrix} \in \mathbf{M}_{2^n}$$

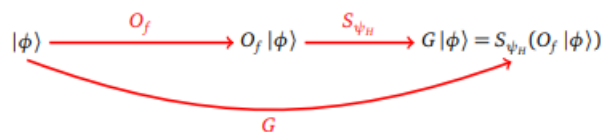


Géométriquement on voit que la transformation S_{ψ_H} d'un n-qubit $|\phi\rangle$ est le symétrique de $|\phi\rangle$ par rapport à l'axe de $|\psi_H\rangle$

3.1.7 Transformation de Grover

La transformation de Grover est l'application [Arnaud]p9

$$G = S_{\psi_H} \circ O_f$$



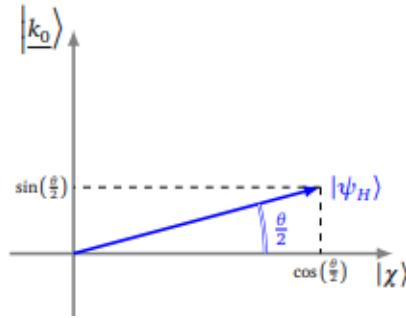
Essayons d'expliquer cette transformation.

Soit le qubit $|\psi_H\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle$ obtenu comme la somme de tous les qubits de base isolant le terme de rang k_o on a :

$$\begin{aligned} |\psi_H\rangle &= \frac{1}{\sqrt{N}} |\underline{k_o}\rangle + \frac{1}{\sqrt{N}} \sum_{k \neq k_o}^{N-2} |\underline{k}\rangle \text{ avec } N = 2^n \\ |\psi_H\rangle &= \frac{1}{\sqrt{N}} |\underline{k_o}\rangle + \sqrt{N-1} \times \frac{1}{\sqrt{N-1}} \frac{1}{\sqrt{N}} \sum_{k \neq k_o}^{N-2} |\underline{k}\rangle \\ |\psi_H\rangle &= \frac{1}{\sqrt{N}} |\underline{k_o}\rangle + \frac{\sqrt{N-1}}{\sqrt{N}} \times \frac{1}{\sqrt{N-1}} \sum_{k \neq k_o}^{N-2} |\underline{k}\rangle \\ |\psi_H\rangle &= \frac{1}{\sqrt{N}} |\underline{k_o}\rangle + \sqrt{\frac{N-1}{N}} \frac{1}{\sqrt{N-1}} \sum_{k \neq k_o}^{N-2} |\underline{k}\rangle \end{aligned}$$

En posant $|\mathcal{X}\rangle = \frac{1}{\sqrt{N-1}} \sum_{k \neq k_o}^{N-2} |\underline{k}\rangle$ on obtient $|\psi_H\rangle = \frac{1}{\sqrt{N}} |\underline{k_o}\rangle + \sqrt{\frac{N-1}{N}} |\mathcal{X}\rangle$
Avec une écriture trigonométrique :

$$|\psi_H\rangle = \cos\left(\frac{\theta}{2}\right) |\mathcal{X}\rangle + \sin\left(\frac{\theta}{2}\right) |\underline{k_o}\rangle \text{ où } \frac{\theta}{2} \text{ est l'angle entre } |\mathcal{X}\rangle \text{ et } |\psi_H\rangle$$

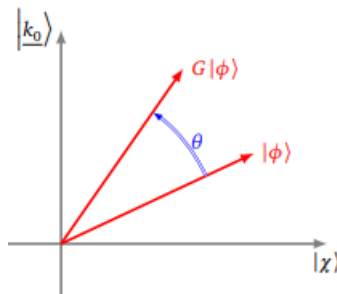


L'angle $\frac{\theta}{2}$ est défini par :

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{N-1}{N}} \text{ et } \sin\left(\frac{\theta}{2}\right) = \frac{1}{\sqrt{N}}$$

La proposition suivante donne une définition de la transformation grover

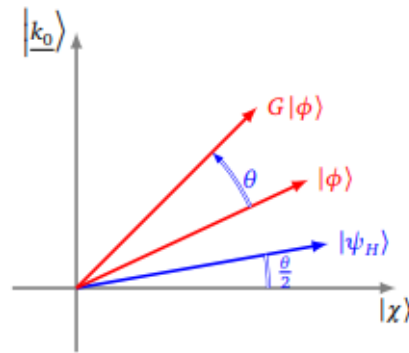
Proposition 10 *La transformation de Grover est une rotation d'angle θ (centrée à l'origine). [Arnaud]p9*



Démonstration : Un résultat géométrique dit que la composition de deux symétries axiales est une rotation d'angle θ de cette rotation étant le double de l'angle entre les axes. Ici G est la composition de deux symétries :

- la symétrie O_f d'axe $|\mathcal{X}\rangle$,
- la symétrie S_{ψ_H} d'axe $|\psi_H\rangle$
- l'angle entre $|\mathcal{X}\rangle$ et $|\psi_H\rangle$ est $\frac{\theta}{2}$

Ainsi $G = S_{\psi_H} \circ O_f$ est la rotation d'angle θ (centrée à l'origine).



Idée de l'algorithme

Le but de l'algorithme de Grover est de déterminer le rang k_o . Ce rang est repérable après l'application de l'oracle O_f . [Arnaud]-p12

$$|\psi_H\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle = \sqrt{\frac{N-1}{N}} |\mathcal{X}\rangle + \frac{1}{\sqrt{N}} |\underline{k_o}\rangle$$

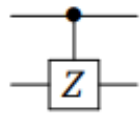
L'action de l'oracle donne $O_f|\psi_H\rangle = \sqrt{\frac{N-1}{N}} |\mathcal{X}\rangle - \frac{1}{\sqrt{N}} |\underline{k_o}\rangle$. Sur ce, nous avons les étapes suivantes :

- La transformation de Hadamard envoie l'état initial $|0.0\dots 0\rangle$ sur $|\psi_H\rangle$.
- On part du qubit ψ_H qui est la superposition de tous les qubits de base
- Ce qubit forme un angle $\frac{\theta}{2}$ avec l'axe $|\mathcal{X}\rangle$. (L'angle $\frac{\theta}{2}$ est petit car $N = 2^n$ est grand)
- La transformation de Grover est une rotation d'angle θ et conduit donc au qubit $G|\psi_H\rangle$ forme un angle $\frac{\theta}{2} + \theta$ avec l'axe $|\mathcal{X}\rangle$.
- On itère la transformation de Grover jusqu'à obtenir un qubit $G^l|\psi_H\rangle$ qui forme un angle d'environ $\frac{\pi}{2}$ avec l'axe $|\mathcal{X}\rangle$. (Ce nombre d'itérations l est environ $\frac{\pi}{2\theta}$)
- Le qubit $G^l|\psi_H\rangle$ obtenu est proche de $|\underline{k_o}\rangle$
- La mesure de ce qubit conduit très probablement à $\underline{k_o}$ (avec une probabilité d'erreur très petite, d'ordre $\frac{4}{N}$)

Portes quantiques

La transformation de Grover est la composition de l'oracle O_f et de la transformation S_{ψ_H} . On a déjà vu le circuit quantique de l'oracle.

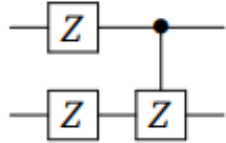
- la porte **Z** est la porte principalement utilisée déjà vue
- **Porte CZ**.est composée d'un contrôleur et d'une porte **Z** fonctionne comme une porte **CNOT** : si l'entrée de la première ligne est $|0\rangle$, alors la seconde ligne est inchangée, par contre si l'entrée de la première ligne est $|1\rangle$, alors on fait agir une porte **Z** sur la seconde ligne.



$$\begin{cases} |0.0\rangle \mapsto |0.0\rangle \\ |0.1\rangle \mapsto |0.1\rangle \\ |1.0\rangle \mapsto |1.0\rangle \\ |1.1\rangle \mapsto -|1.1\rangle \end{cases} \quad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

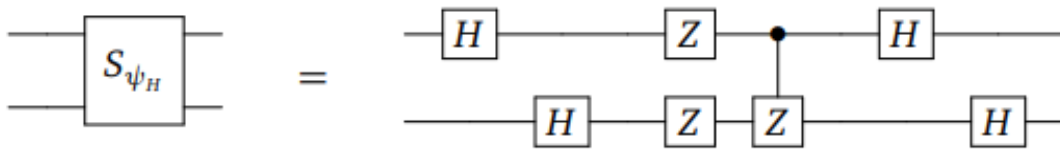
Circuit quantique

Circuit pour S_0 Voici un circuit qui permet de réaliser la transformation S_0 , dans le cas des 2-qubits. Noter bien que la partie droite du circuit est une porte **CZ**.



$$\begin{cases} |0.0\rangle \mapsto |0.0\rangle \\ |0.1\rangle \mapsto -|0.1\rangle \\ |1.0\rangle \mapsto -|1.0\rangle \\ |1.1\rangle \mapsto -|1.1\rangle \end{cases} \quad S_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Circuit pour S_{ψ_H} On sait que $S_{\psi_H} = H^{\otimes n} \cdot S_0 \cdot H^{\otimes n}$, il suffit juste d'appliquer la transformation de Hadamard avant et après le circuit S_0



3.1.8 Etapes de l'algorithme de Grover

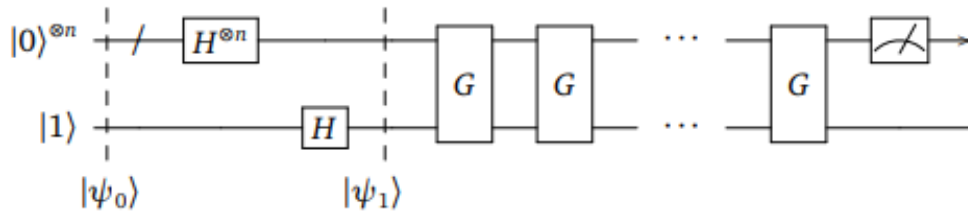
Circuit

On note **G** la transformation de Grover, elle prend en entrée un $(n+1)$ -qubit, et est formée par la porte O_f de l'oracle, suivie d'une porte associée à la transformation S_{ψ_H} . On repré-

sente cette porte **G** avec 2 lignes seulement, la première ligne correspond à un n -qubit (ligne symbolisée avec /), la seconde à un 1 -qubit.



Voici le circuit de l'algorithme de Grover.



La porte **G** est itérée l fois avec $l \approx \frac{\pi}{4} \sqrt{N}$ où $N = 2^n$. La complexité de l'algorithme est d'ordre l , donc d'ordre $O(\sqrt{N})$. La mesure finale est la mesure d'un n -qubit (et correspond donc à n mesures de 1 -qubits). Le circuit renvoie donc un n -bit classique \underline{k} avec $0 \leq k < 2^n$. Nous allons justifier que cet entier est très probablement le rang k_0 cherché. [Arnaud]-14

Données

Soit $n \geq 1$ et $N = 2^n$. Supposons donné un entier k_0 vérifiant $0 \leq k_0 \leq N$. On considère la fonction $f : \mathbb{Z}/N\mathbb{Z} \rightarrow \mathbb{Z}/2\mathbb{Z}$, avec $f(k_0) = 1$ et $f(k) = 0$ pour tout $k \neq k_0$.

Initialisation et transformation de Hadamard

Le circuit quantique est initialisé par $(n+1)$ -qubit [Arnaud]

$$|\psi\rangle = |0\dots 0\rangle \cdot |1\rangle = |\underline{0}\rangle \cdot |1\rangle$$

Ensuite on applique la transformation de Hadamard pour obtenir le qubit

$$\begin{aligned} |\psi_1\rangle &= \mathbf{H}^{\otimes n+1} |\psi_0\rangle \\ &= \mathbf{H}^{\otimes n} |\underline{0}\rangle \cdot \mathbf{H} |1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |\underline{k}\rangle \cdot \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\ &= |\psi_H\rangle \cdot \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \end{aligned}$$

Dans la suite on oublie le dernier qubit et on s'intéresse seulement au n -qubit formé par les n premières lignes.

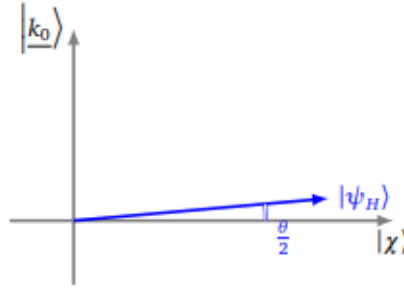
Dans $|\psi_H\rangle$ distinguons la qubit de base $|k_0\rangle$:

$$|\psi_H\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle = \sqrt{\frac{N-1}{N}} |\mathcal{X}\rangle + \frac{1}{\sqrt{N}} |k_0\rangle$$

que l'on récrit sous forme trigonométrique :

$$|\psi_H\rangle = \cos\left(\frac{\theta}{2}\right) |\mathcal{X}\rangle + \sin\left(\frac{\theta}{2}\right) |k_0\rangle$$

où $\frac{\theta}{2}$ est l'angle entre $|\mathcal{X}\rangle$ et $|\psi_H\rangle$, également défini par la relation $\sin\frac{\theta}{2} = \frac{1}{\sqrt{N}}$

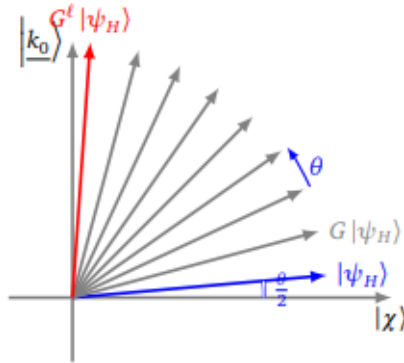


Itérations de la transformation de Grover

la transformation de Grover $G = S_{\psi_H} \circ O_f$, est une rotation d'angle θ . Donc après l itérations on obtient le n-qubit.

$$G^l |\psi_H\rangle = \cos(\theta_l) |\mathcal{X}\rangle + \sin(\theta_l) |k_0\rangle \text{ avec } \theta_l = \frac{\theta}{2} + l\theta.$$

On veut $\theta_l \approx \frac{\pi}{2}$, c'est-à-dire $\frac{\theta}{2} + l\theta \approx \frac{\pi}{2}$. Ainsi l est défini comme l'entier le plus proche de $\frac{\pi}{2\theta} - \frac{1}{2}$.



Donnons une approximation du nombre l d'itérations nécessaires. Pour cela nous considérons que $N = 2^n$ est grand, et donc θ est petit. Comme $\sin\left(\frac{\theta}{2}\right) = \frac{1}{\sqrt{N}}$ alors $\frac{\theta}{2} \approx \frac{1}{\sqrt{N}}$ (car pour x proche de 0, $\sin(x) \approx x$). On veut $l\theta \approx \frac{\pi}{2}$ donc $l \approx \frac{\pi}{2\theta}$ et ainsi

$$l \approx \frac{\pi}{4} \sqrt{N}.$$

Mesure

Après ces l itérations nous avons $\theta_l \approx \frac{\pi}{2}$, donc

$$G^l|\psi_H\rangle = \cos(\theta_l)|\mathcal{X}\rangle + \sin(\theta_l)|\underline{k_o}\rangle \approx |\underline{k_o}\rangle.$$

La mesure de ce n-qubit conduit donc probablement au n-bit $\underline{k_o}$ et permet alors d'identifier le rang k_o .

Probabilité de Grover

Proposition 11 *L'algorithme de Grover renvoie le rang correct k_o avec une probabilité supérieure à $1 - \frac{4}{N}$.* [Arnaud]-p16

Démonstration

- Le qubit final obtenu par l'algorithme de Grover est

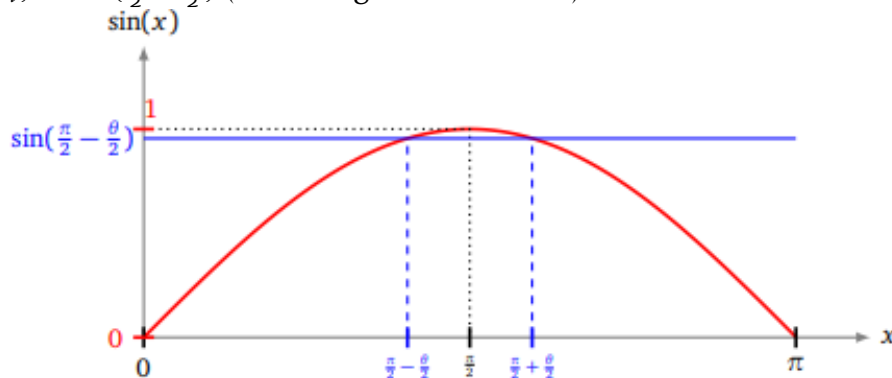
$$G^l|\psi_H\rangle = \cos(\theta_l)|\mathcal{X}\rangle + \sin(\theta_l)|\underline{k_o}\rangle$$

Donc, lors de la mesure, la probabilité d'obtenir la bonne réponse $\underline{k_o}$ est $p = |\sin(\theta_l)|^2$.

- Nous savons que la transformation de Grover G est une rotation d'angle θ et nous avons itéré cette transformation l fois de façon à construire un angle $\theta_l = \frac{\theta}{2} + l\theta$ le plus proche possible de l'angle $\frac{\pi}{2}$. Ainsi l'angle θ_l est dans un intervalle d'amplitude θ centré en $\frac{\pi}{2}$

$$\frac{\pi}{2} - \frac{\theta}{2} < \theta_l \leq \frac{\pi}{2} + \frac{\theta}{2}$$

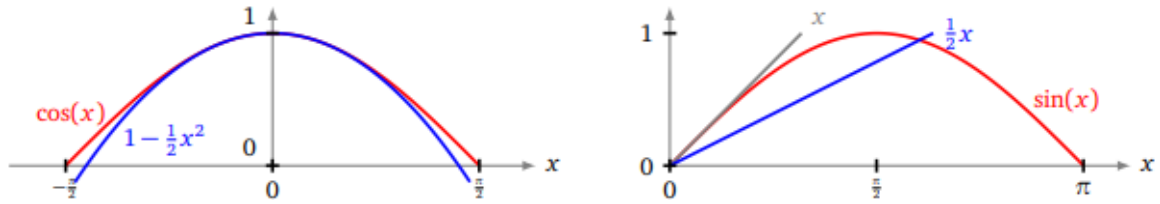
- Ainsi $\sin(\theta_l) \geq \sin(\frac{\pi}{2} - \frac{\theta}{2})$ (voir la figure ci-dessous)



Donc :

$$\sin(\theta_l) \geq \sin(\frac{\pi}{2} - \frac{\theta}{2}) = \cos(\frac{\theta}{2}) \geq 1 - \frac{1}{2}(\frac{\theta}{2})^2$$

. Pour la dernière inégalité on connaît le développement limité $\cos(x) \approx 1 - \frac{x^2}{2}$ (pour x proche de 0) mais on a en plus l'inégalité $\cos(x) \geq 1 - \frac{x^2}{2}$ (figure de gauche ci-dessous).



- L'angle θ est défini avec la relation $\sin(\frac{\theta}{2}) = \frac{1}{\sqrt{N}}$. On sait, pour x proche de 0, on a $\sin(x) \approx x$, mais on a en plus l'inégalité $\sin(x) \geq \frac{x}{2}$ (voir figure de droite ci-dessous). Ainsi, comme $\sin(\frac{\theta}{2}) = \frac{1}{\sqrt{N}}$, alors $\frac{1}{\sqrt{N}} \geq \frac{\theta}{4}$ donc $\frac{4}{N} \geq (\frac{\theta}{2})^2$ et alors en reprenant les inégalités ci-dessus :

$$\sin(\frac{\theta_l}{2}) \geq 1 - \frac{1}{2}(\frac{\theta}{2})^2 \geq 1 - \frac{2}{N}$$

Enfin on a $(1-x)^2 = 1 - 2x + x^2 \geq 1 - 2x$ quel que soit x , donc

$$p = |\sin(\theta_l)|^2 \geq \left(1 - \frac{2}{N}\right)^2 \geq 1 - \frac{4}{N}. \text{ [Arnaud]-p17}$$

3.2 Les outils mathématiques des ISD

3.2.1 Le coefficient binomial et la fonction d'entropie

Définition 23 On définit la fonction d'entropie binaire par :

$$f \mapsto \begin{matrix} [-1, 1] \longrightarrow [0, \frac{1}{2}] \\ -(1-x) \log_2(1-x) - x \log_2(x) \end{matrix}$$

Théorème 9 Si H désigne la fonction d'entropie de Shannon, on a : $\frac{2^{nH(\frac{t}{n})}}{n+t} \leq \binom{n}{t} \leq 2^{nH(\frac{t}{n})}$

Nous écrirons pour cela $\binom{n}{t} \approx 2^{nH(\frac{t}{n})}$. [Ghazal]

3.2.2 Codes poinçonnés

Définition 24 Étant donné un code C de matrice génératrice \mathbf{G} et de paramètres $[n, k]$, le poinçonnage consiste à supprimer des colonnes de la matrice génératrice. [Ghazal]-p1

Soit I un ensemble d'indice, $|I| = i$. On note C_I le code poinçonné dont la matrice génératrice G_I est la matrice de G dont on a enlevé les colonnes indicées par I . Alors C_I est de paramètres $[n-i, k]$. On notera H_I la matrice de parité de C_I .

Théorème 10 Soit m un message de longueur n et $m_I := m|I|$, $|I| = i$. Soit $c_I = m_I G_I$. Alors s'il y a une sous-matrice carrée inversible de G_I , alors il est possible de dériver $c := mG$ de manière unique à partir de c_I . [Ghazal]

Démonstration On supposera pour simplifier que l'on a mis G_I est sous forme systématique.

On suppose aussi que $i = 1$ et que $I = n$. Alors G s'écrit (à des permutations de colonnes et d'opérations de pivot de Gauss près) $(G_I | g_n)$, plus précisément :

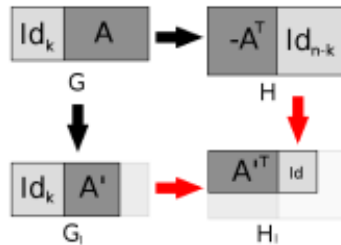
$$\left[\begin{array}{ccc|ccc} 1 & \cdots & 0 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 1 & \cdots & \cdots & \cdots \end{array} \middle| \begin{array}{c} g_{1,n} \\ \cdots \\ g_{n,k} \end{array} \right]$$

Par conséquent si l'on écrit $m = (m_1, \dots, m_n)$, $m_I = (m_1, \dots, m_{n-1})$, $cI = (c_{I_1}, \dots, c_{I_k})$, c s'écrit $c = (c_{I_1} + g_{n,1}m_n, \dots, c_{I_k} + g_{n,k}m_n)$. Il suffit donc de connaître m_n et la colonne g_n de la matrice pour pouvoir calculer c à partir de c_I .

Ce résultat se généralise facilement au cas où $i > 2$, il faudrait alors connaître les $(m_j)_{j \in I}$ et les colonnes $(g_j)_{j \in I}$

Théorème 11 Soit le code C ayant pour matrice génératrice $G = [Id | A]$ et donc pour matrice de parité $H = [-A^T | Id]$ et soit C_I le code obtenu en poinçonnant les $i < k$ dernières de G , alors une matrice de parité H_I de C_I est donnée par la sous-matrice $(n-i) \times (n-k-i)$ "en haut à gauche" de H . [Ghazal]

Illustration de ce théorème :



Démonstration : On peut écrire $G = [Id_k | A] = [Id_k | A_1 | A_2]$ où $A_1 \in M_{k, n-k-i}(\mathbb{F}_2)$ et $A_2 \in M_{k, i}(\mathbb{F}_2)$ correspond à la partie qui sera poinçonnée.

Alors, d'un côté $H = [-A^T | Id_{n-k}]$ avec $A^T = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix}$ donc au final, H s'écrit de manière plus

détaillée :

$$\left[\begin{array}{ccc|cc} -A_1^T & Id_{n-k-i} & 0_{n-k-i, n-i} & & \\ A_2^T & 0_{i, n-k-i} & Id_i & & \end{array} \right]$$

D'un autre côté, $G_I = [Id_k | A_1] \in M_{k, n-i}(\mathbb{F}_2)$ donc $H_I = [A_1^T | Id_{n-k-i}]$, ce qui correspond à la sous-matrice de taille $(n-i) \times (n-k-i)$ "en haut à gauche" de H .

3.2.3 Dénombrement

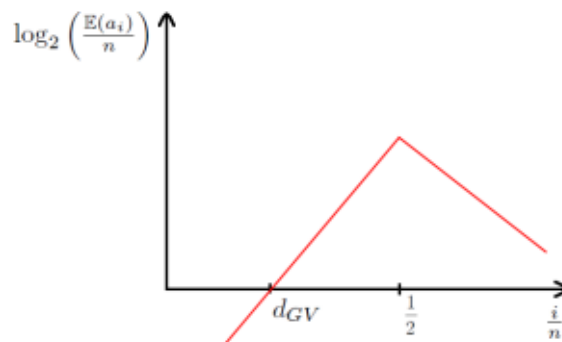
Théorème 12 Soit C un code linéaire de paramètres $[n, k]$. Alors pour un syndrome $s \neq 0$ donné, le nombre moyen de mots $e \in C$ de poids p ayant pour syndrome s est $\frac{\binom{n}{p}}{2^{n-k}}$.

Démonstration : admis [Ghazal]-p3

3.2.4 Distance de Gilbert-Varshamov

Théorème 13 Soit C un code linéaire de paramètres $[n, k]$. Soit a_i le nombre moyen de mots de poids i dans C . Alors :

1. $\mathbb{E}(a_i) = 1$ ssi $H\left(\frac{i}{n}\right) = 1 - R \Leftrightarrow \frac{i}{n} = H^{-1}(1 - R)$. On appelle cette valeur de i distance de Gilbert-Varshamov et on note cette fraction $d_{GV} := H^{-1}(1 - R)$. [Ghazal]-p4
2. Plus généralement, le graphique suivant montre l'évolution du rapport $\log_2\left(\frac{\mathbb{E}(a_i)}{n}\right)$ en fonction de $\frac{i}{n}$:



Démonstration : On a $a_i = \sum_{x, w_H(x)=i} \mathbf{1}_{x \in C}$

Par conséquent :

$$\begin{aligned}
 \mathbb{E}(a_i) &= \sum_{x, w_H(x)=i} \mathbf{1}_{x \in C} \\
 &= \sum_{x, w_H(x)=i} \mathbb{P}(x \in C) \\
 &= \sum_{x, w_H(x)=i} \frac{1}{2^{n-k}} \\
 &= \frac{\binom{n}{i}}{2^{n-k}} \\
 &= \approx 2^{nH\left(\frac{i}{n}\right) - (n-k)} \\
 &= 2^{n\left(H\left(\frac{i}{n}\right) - 1 + R\right)}
 \end{aligned}$$

On pose $f(x) = H(x) - 1 + R$, on a alors $f(\frac{i}{n}) = \log_2 \left(\frac{\mathbb{E}(a_i)}{n} \right)$ et :

1. $f(x) = 0 \Leftrightarrow x = H^{-1}(1 - R)$ donc $\mathbb{E}(a_i) = 1$ ssi $\frac{i}{n} = d_{GV}$
2. $\max_{x \in [0,1]} f(x) = R$ pour l'argument $x = \frac{1}{2}$.

3.2.5 Combinatoire et probabilités

Combinatoire

Théorème 14 Soient S un ensemble, M un ensemble de données, $f : S \rightarrow M$ une fonction aléatoire ou pseudo-aléatoire. On considère $m \in M$, $P_m : S \rightarrow \{0, 1\}$ tel que $P_m(s) = 1$ si $f(s) = m$ et $P_m(s) = 0$ sinon. Alors le nombre moyen d'éléments s de S tels que $P_m(s) = 1$ est $\frac{|S|}{|M|}$. [Ghazal]

Démonstratin : On numérote les éléments de S et on définit pour s_i , le $i^{\text{ème}}$ éléments, la variable aléatoire suivante :

$$X_i = P_m(s_i)$$

Comme f est aléatoire ou pseudo-aléatoire, on a $\mathbb{P}(X_i = 1) = M^{-1}$ pour tout i . La variable $X := \sum_i X_i$ donne alors le nombre d'éléments de S tels que $P_m(s) = 1$. On calcule $\mathbb{E}(X)$:

$$\mathbb{E}(X) = \sum_i \mathbb{E}(X_i) = |S| \mathbb{P}(X_i = 1) = \frac{|S|}{|M|}$$

Théorème 15 Soient S_1 et S_2 des ensembles, M un ensemble de données, $f_1 : S_1 \rightarrow M$ et $f_2 : S_2 \rightarrow M$ des fonctions aléatoires ou pseudo-aléatoires. Alors la probabilité de collision des valeurs de f_1 et f_2 est $\frac{|S_1||S_2|}{|M|}$. [Ghazal]

Démonstration : On considère le produit cartésien $S = S_1 \times S_2$ et on numérote ses $|S_1| \parallel |S_2|$ éléments. On définit pour son $i^{\text{ème}}$ élément $(x, y)_i$ la variable aléatoire suivante :

$X_i = 1$ si $f_1(x) = f_2(y)$, $X_i = 0$ sinon. Comme f_1 et f_2 sont aléatoires ou pseudo-aléatoires, on a $\mathbb{P}(X_i = 1) = M^{-1}$ pour tout i . La variable aléatoire $X := \sum_i X_i$ donne alors le nombre d'éléments (x, y) de S tels que $f_1(x) = f_2(y)$ i.e. le nombre de collisions. On calcule $\mathbb{E}(X)$:

$$\mathbb{E}(X) = \sum_i \mathbb{E}(X_i) = |S_1| \parallel |S_2| \mathbb{P}(X_i = 1) = \frac{|S_1||S_2|}{|M|}$$

Théorème 16 Soient $X \sim B(p)$, $Y \sim B(q)$ des variables aléatoires. Alors $X + Y \sim B(p + q - 2pq)$. En particulier, si $p = q$, $X + Y \sim B(2p(1 - p))$. [Ghazal]

Démonstration

$$\begin{aligned}\mathbb{P}(X \oplus Y = 1) &= \mathbb{P}(X = 0, Y = 1) + \mathbb{P}(X = 1, Y = 0) \\ &= (1-p)q + (1-q)p \\ &= p + q - 2pq\end{aligned}$$

3.2.6 Un algorithme de jointure : Merge-Join

Soient E_1, E_2, E des ensembles quelconques, $|E_i| \leq M \in \mathbb{N}$ et $f_i : E_i \rightarrow E$ des fonctions pour $i = 1, 2$. Un algorithme appelé *Merge-Join*, qui permet de résoudre le problème suivant :

Problème 3 : (*Merge-Join*) Etant donné deux listes $L_1 = ((x_i, f(x_i)))_{i \in I_1}$ et $L_2 = ((y_i, f_2(y_i)))_{i \in I_2}$, donner la liste L des éléments $(x_i, x_j)_{i \in L_1 \times L_2}$ qui vérifient $f_1(x_i) = f_2(x_j)$ et éventuellement d'autres conditions.

En pratique, les E_i seront des ensembles de vecteurs binaires de même longueur n , donc $M = 2^n$.

Cet algorithme renvoie une liste dont les éléments sont obtenus en joignant ceux des listes d'entrée et en gardant ceux qui vérifient certaines conditions, tout en prenant soin d'éviter les doublons.

Algorithme : Merge-Join

Entrées : $L_1 = ((x_i, f(x_i)))_{i \in I_1}$ et $L_2 = ((y_i, f_2(y_i)))_{i \in I_2}$

Sorties : Une liste $L := \{((x_i, y_j), \mathbf{F}(f_1(x_i), f_2(y_j))) | f_1(x_i) = f_2(y_j)\}$

1. Trier L_1 et L_2 dans l'ordre croissant ou lexicographique en fonction de leurs premiers entrées
2. \leftarrow Nouvelle liste
3. $i \leftarrow 0, j \leftarrow |L_2| - 1$
4. **tant que** $i > |L_1|$ et $j \geq 0$ **faire**
5. **si** $f_1(x_i) < f_2(y_j)$ **faire**
6. $i \leftarrow i + 1$
7. **si** $f_1(x_i) > f_2(y_j)$ **faire**
8. $j \leftarrow j - 1$
9. **si** $f_1(x_i) = f_2(y_j)$ **faire**
10. $L \cup ((x_i, y_j), \mathbf{F}(f_1(x_i), f_2(y_j)))$

La complexité de l'étape de tri est $O(|L_1| \log(|L_1|) + |L_2| \log(|L_2|)) = \tilde{O}(|L_1| + |L_2|)$. Le nombre attendu de tours de boucles est égal au nombre de collisions, donc il y en a $O(\frac{|L_1||L_2|}{M})$. On obtient une complexité en $\tilde{O}(|L_1| + |L_2| + \frac{|L_1||L_2|}{M}) = \tilde{O}(\max(|L_1|, |L_2|, \frac{|L_1||L_2|}{M}))$ en temps et $\tilde{O}(|L_1| + |L_2| + |L|) = \tilde{O}(\max(|L_1|, |L_2|, |L|))$ en espace.

NOTATION : On utilisera la notation $L_1 \bowtie L_2$ pour indiquer un Merge-Join. À chaque utilisation, on ajoutera néanmoins quelques autres symboles en-dessous et au-dessus du papillon \bowtie qui résument les spécificités du cas en question.

3.2.7 Théorie des graphes

Graphes d-réguliers

Définition 25 un graphe $G = (E, V)$ est dit d -régulier lorsque tous les sommets ont le même degré, i.e. le même nombre de voisins.

Valeurs propres de la matrice d'adjacence d'un graphe

Soit $G = (E, V)$ un graphe à N sommets et M arêtes. et on considère sa matrice d'adjacence, modifiée de la manière suivante : on divise chaque ligne par le nombre d'entrées non-nulles (par conséquent, par le degré sortant du sommet en question) pour la transformer en une matrice stochastique. On obtient ainsi une matrice de transition d'une chaîne de Markov. En effet, rappelons que pour définir une chaîne de Markov, il faut une distribution de probabilité initiale ν sur un ensemble (de taille n) ainsi qu'une matrice carrée de transition P contenant les probabilités conditionnelles de transition :

$$P = \begin{bmatrix} P(1|1) & \cdots & P(1|n) \\ \cdots & \cdots & \cdots \\ P(n|1) & \cdots & P(n|n) \end{bmatrix}$$

On a $P_{t+1}(j) = \sum_i P(j|i)P_t(i)$

Cette matrice d'adjacence a une valeur propre 1 correspondant au vecteur uniforme $u = (\frac{1}{N}, \dots, \frac{1}{N})$.

Si le graphe est connexe, il y a un unique vecteur propre associé à la valeur propre 1.

Trou spectral Soit $G = (E, V)$ un graphe d -régulier, non-dirigé et connexe à N sommets et P la chaîne de Markov donnée par sa matrice d'adjacence. Grâce à cela nous pouvons effectuer une marche aléatoire sur le graphe.

Nous savons qu'il y a un vecteur propre $v_1 = u$ de valeur propre $\lambda_1 = 1$ et tous les $N - 1$ autres vecteurs propres v_i auront pour valeur propre $\lambda_i \in]-1, 1[$. On définit le trou spectral δ du graphe/de la chaîne : $\delta := 1 - \max_{i \geq 2} |\lambda_i|$. On peut écrire $v = \sum_{i=1}^n \alpha_i v_i$ où v est la distribution de probabilité initiale.

$$\begin{aligned} \|P^k v - u\|^2 &= \|\sum_i \alpha_i P^k v_i - u\|^2 \\ &= \|\sum_i \alpha_i \lambda_i^k v_i + u - u\|^2 \\ &\leq \sum_{i \geq 2} \alpha_i^2 \lambda_i^{2k} \|v_i\|^2 \\ &\leq (1 - \delta)^{2k} \sum_i \alpha_i^2 \\ &\leq (1 - \delta)^{2k} \|v\|^2 \\ &\leq (1 - \delta)^{2k} \end{aligned}$$

Par conséquent, en prenant $k = \frac{\ln(\frac{1}{\eta})}{\ln(\frac{1}{\delta})}$ on aura $\|P^k v - u\| \leq (1 - \delta)^{\frac{\ln(\frac{1}{\eta})}{\ln(\frac{1}{\delta})}} \leq \eta$. Or lorsque δ est petit, $\ln(\frac{1}{\delta}) \approx \delta$. Par conséquent, pour avoir $P^k v$ proche de u il faut $k = O(\frac{1}{\delta})$ étapes de la marche aléatoire.

En général, il est rare que $\delta \approx 1$. Mais pour un graphe d -régulier aléatoire, cela arrive très souvent : dès que le degré du graphe est supérieur à 2, lorsque $d \rightarrow n$, $\delta \rightarrow 1$.

Graphes de Johnson

On a vu dans la section précédente que pour connaître le nombre d'étapes de marche aléatoire à faire pour arriver à une distribution uniforme, il faut connaître le trou spectral du graphe/de la chaîne de Markov.

Or il y a peu de graphes pour lesquels on peut connaître facilement les valeurs propres et donc le trou spectral. Quelques exemples seraient les graphes de Johnson, qui nous intéresseront particulièrement, et les n -cubes ou hypercubes.

Définition 26 Un graphe de Johnson $J(n, r)$ est un graphe non-dirigé dont les sommets sont les sous-ensembles de taille r d'un ensemble de taille n , avec une arête entre deux sommets S et S' ssi $|S \cap S'| = r - 1$, autrement dit si pour obtenir S' à partir de S il suffit de supprimer un élément et ajouter un nouvel élément à sa place. [Ghazal]

Les graphes de Johnson ont les propriétés suivantes :

1. Le nombre de sommets est clairement $\binom{n}{r}$.
2. Pour chaque sous-ensemble S de taille r , il y a r éléments qui peuvent être supprimés et chacun peut être remplacé de $(n - r)$ manières. Par conséquent il y a $d = r(n - r)$ sous-ensembles de taille r qui diffèrent d'un élément par rapport à S . Par conséquent $J(n, r)$ est d -régulier.

3. Les graphes de Johnson sont un cas particulier des schémas de Johnson. Sans rentrer dans les détails, il est possible de calculer les valeurs propres d'un schéma de Johnson à l'aide de polynômes d'Eberlein. Pour un graphe de Johnson, ces valeurs propres sont :

$$d\lambda_i = (r-i)(n-r-i) - i = rn - r^2 + i^2 - in - i \text{ pour } (0 \leq i \leq n-1)$$

Ainsi $\lambda_0 = 1$ et

$$\begin{aligned} d \max_{i \geq 1} |\lambda_i| &= \max_{i \geq 1} |r(n-r) + i(i-n-1)| \\ &= \max_{i \geq 1} |d - i(n-i+1)| \\ &= d - \min_{i \geq 1} |i(n-i+1)| \\ &= d - n. \end{aligned}$$

Par conséquent $\delta = 1 - \max_{i \geq 1} |\lambda_i| = \frac{n}{d} = \frac{n}{r(n-r)}$.

3.2.8 Produits de graphe

Définition 27 (*Produit cartésien de graphes*). Etant donné deux graphes $G_1 = (E_1, V_1)$ et $G_2 = (E_2, V_2)$, On définit leur produit cartésien $G_1 \times G_2 = G = (E, V)$:

1. $V = V_1 \times V_2 \Leftrightarrow \{v_1 v_2 | v_1 \in V_1, v_2 \in V_2\}$
2. $E = \{(u_1 u_2, v_1 v_2) | (u_1 = v_1 \wedge (u_2, v_2) \in E_2) \vee ((u_1, v_1) \in E_1 \wedge u_2 = v_2)\}$

On notera dorénavant $n = |V|$, $m = |E|$ et pour $i = 1, 2$, $n_i = |V_i|$, $m_i = |E_i|$.

On va également supposer que G_i est d_i -régulier.

Théorème 17 (*Matrice d'adjacence d'un produit cartésien de graphes*). Soit $A_i \in \mathcal{M}_{n_i, n_i}(\mathbb{Z})$ la matrice d'adjacence du graphe G_i pour $i = 1, 2$. Alors la matrice d'adjacence A du graphe $G = G_1 \times G_2$ est :

$$A = I_{n_1} \otimes A_2 + A_1 \otimes I_{n_2}$$

Démonstration : Admis

Théorème 18 (*Vecteurs propres et valeurs propres d'un produit cartésien de graphes*)[Ghazal]. Soient, pour $i = 1, \dots, k$, $k_1 \geq n_1$, λ_i la valeur propre (non-normalisée) du graphe G_1 associée au vecteur propre f_i .

Soient, pour $j = 1, \dots, k_2$, $k_2 \geq n_2$, μ_j la valeur propre (non-normalisée) du graphe G_2 associée au vecteur propre g_j .

Alors le graphe $G = G_1 \times G_2$ possède $k_1 k_2$ vecteurs propres. Ces vecteurs propres sont : $h_{i,j} = f_i \otimes g_j$ de valeur propre $v_{i,j} = \lambda_i + \mu_j$, pour $i = 1, \dots, k_1$, $j = 1, \dots, k_2$.

Démonstration : Admis [Ghazal]

Dorénavant, on supposera que les valeurs propres des graphes G_1 et G_2 sont indicés du plus grand au plus petit, c'est-à-dire :

$$d_1 = \lambda_1 \leq \dots \leq \lambda_{k_1}$$

$$d_2 = \mu_1 \leq \dots \leq \mu_{k_2}$$

On a alors clairement $\max_{i,j} v_{i,j} = \lambda_1 + \mu_1 = d_1 + d_2$

Théorème 19 (Trou spectral d'un produit cartésien de graphes). Soient $\delta_1 = \frac{d_1 - \max_{i=2, \dots, k_1} |\lambda_i|}{d_1}$ le trou spectral de G_1 , et $\delta_2 = \frac{d_2 - \max_{j=2, \dots, k_2} |\mu_j|}{d_2}$ le trou spectral de G_2 .

Alors, si on note $\delta = \frac{d_1 + d_2 - \max_{i,j} |\lambda_i + \mu_j|}{d_1 + d_2}$ le trou spectral du graphe G , on a :

$$\delta \geq \frac{d_1 \delta_1 + d_2 \delta_2}{d_1 + d_2} \text{ [Ghazal]-10}$$

Démonstration :

$$\begin{aligned} \delta &= \frac{d_1 + d_2 - \max_{i,j} |\lambda_i + \mu_j|}{d_1 + d_2} \geq \frac{d_1 + d_2 - \max_i |\lambda_i| - \max_j |\mu_j|}{d_1 + d_2} \\ &= \frac{d_1 - \max_i |\lambda_i|}{d_1} + \frac{d_2 - \max_j |\mu_j|}{d_2} \\ &= \frac{d_1 \delta_1 + d_2 \delta_2}{d_1 + d_2} \end{aligned}$$



Sans perdre notre fil conducteur on verra dans la suite que les algorithmes d'attaques seront en deux catégories : ceux utilisant les notions de mathématiques telles que la théorie des graphes et ceux implémentant l'algorithme de Grover avec usage des circuits et des transformations quantiques. Ces algorithmes donnent aussi une réponse suivant une probabilité. [Grenet]

3.3 Les Algorithmes quantiques

3.3.1 Algorithme de Grover

L'algorithme de Grover résout le problème suivant : étant donné une fonction $f : E \rightarrow \{0, 1\}$, trouver un argument x tel que $f(x) = 1$. La technique à la base de l'algorithme de Grover

s'appelle l'amplification de l'amplitude. Nous allons considérer le cas où il y a plusieurs solutions au problème.[Ghazal]-p20

L'algorithme de Grover nécessite les ingrédients suivants :

Soit $N = 2^n$

1. Étant donné une fonction $f : \mathbb{Z} \rightarrow \{0, 1\}$ (concrètement, une fonction indicatrice ou un oracle de vérification d'une propriété), on définit pour $|z\rangle$ de n qubits $O_f : |z\rangle \mapsto (-1)^{f(z)}|z\rangle$. Si l'on nomme "bons" les $|z\rangle$ tels que $f(z) = 1$ et "mauvais" ceux qui vérifient $f(z) = 0$, l'action de O_f consiste à laisser fixes les "mauvais" éléments et changer le signe de l'amplitude (de la phase) des "bons" éléments. O_f a le même temps de calcul que f .

S'il y a t bonnes valeurs de z et que l'on note $|B\rangle := \frac{1}{\sqrt{t}} \sum_{z, f(z)=1} |z\rangle$ et $|M\rangle := \frac{1}{\sqrt{N-t}} \sum_{z, f(z)=0} |z\rangle$, alors on a $|B\rangle \perp |M\rangle$ et $O_f = 2|M\rangle\langle M| - Id$.

Par conséquent, cette opération consiste en une réflexion à travers $|M\rangle$ qui laisse fixe la phase des mauvais éléments et inverse la phase des bons éléments. Ainsi, on peut encore l'appeler inversion de la phase.

2. On considère la porte de Hadamard $H^{\otimes n}$, qui renvoie $|o^n\rangle$ sur $\frac{1}{\sqrt{N}} \sum_{z, f(z)=1} |z\rangle$, une superposition uniforme de tous les éléments de la base. Par conséquent, s'il y a t bonnes valeurs de z la probabilité de tomber sur un bon z en mesurant devient $\frac{t}{N}$.

On note $|U\rangle := H^{\otimes n}|o^n\rangle$ et $R := 2|o^n\rangle\langle o^n| - Id$.

On a alors :

$H^{\otimes n}R(H^{\otimes n})^* = 2H^{\otimes n}|o^n\rangle\langle o^n|(H^{\otimes n})^* - H^{\otimes n}(H^{\otimes n})^* = 2|U\rangle\langle U| - Id$. Cette opération est par conséquent une réflexion par $|U\rangle$, on peut la nommer inversion par rapport à la moyenne.

Nous notons enfin qu'il est possible d'appliquer, à la place de $H^{\otimes n}$, n'importe quel algorithme A qui, appliqué à $|o^n\rangle$, rend un état superposé où en mesurant on obtiendra un bon z avec probabilité p .

L'algorithme de Grover se déroule ainsi, avec $\epsilon := \frac{t}{N}$:

Algorithme 4 : Grover [Ghazal]

Entrées : ϵ, f

Sorties : une superposition $|Z\rangle$ des états $|z\rangle$ tel que $f(z) = 1$

1. Préparer l'état de départ $|U\rangle$
2. **pour** $\frac{1}{\sqrt{\epsilon}}$ itérations **faire**
3. Appliquer O_f à l'état courant (inversion de la phase)
4. Appliquer $H^{\otimes n}RH^{\otimes n}$ à l'état courant (inversion par rapport à la moyenne)

On a à la sortie de l'algorithme de Grover une superposition $|Z\rangle$ de bons états. Nous pouvons ensuite soit mesurer le registre contenant $|Z\rangle$ afin d'obtenir la valeur d'un bon état $|z\rangle$, ou bien le laisser tel quel, par exemple pour servir d'entrée à un autre algorithme.

Si nous ne connaissons pas t en avance, on ne saura pas combien de fois il faut boucler dans l'algorithme. Il est possible de pallier à ce problème, sans grande perte de temps d'exécution, en devinant des valeurs de k d'une certaine façon.

Analyse de complexité

On a tout d'abord l'égalité suivante [Ghazal]-p21 :

$$\begin{aligned} & \sin(\arcsin(\sqrt{\epsilon}))|B\rangle + \cos(\arcsin(\sqrt{\epsilon}))|M\rangle \\ &= \sqrt{\epsilon}|B\rangle + \sqrt{1-\epsilon}|M\rangle \\ &= \sqrt{\frac{t}{N}} \frac{1}{\sqrt{t}} \sum_{z,(z)=1} |z\rangle + \frac{N-t}{N} \frac{1}{\sqrt{N-t}} \sum_{z,(z)=1} |z\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} |z\rangle \\ &= |U\rangle \end{aligned}$$

Si l'on note $\theta = \arcsin(\sqrt{\epsilon})$

On peut voir géométriquement qu'un tour de la boucle dans l'algorithme de Grover augmente l'angle de 2θ degrés. Par conséquent, après k tours de boucle, l'état devient : $\sin((2k+1)\theta)|B\rangle + \cos((2k+1)\theta)|M\rangle$.

La probabilité d'observer les bons états, i.e. $|B\rangle$ est alors de $\sin((2k+1)\theta)$. Pour que cette probabilité soit proche de 1 il faut :

$$(2k+1)\theta \approx \frac{\pi}{2} \Rightarrow k \approx \frac{\pi}{4\theta} - \frac{1}{2} \Rightarrow k = O(\frac{1}{\theta}) = O(\frac{1}{\sqrt{\epsilon}}) = O(\sqrt{\frac{N}{t}})$$

L'algorithme de Grover a par conséquent une complexité en requêtes en $O(\frac{1}{\sqrt{\epsilon}})$. De plus, il a été montré que ce type de recherche nécessite au minimum $O(\sqrt{N})$ étapes. L'algorithme de Grover est par conséquent optimal.

En conclusion, puisque l'on fait une seule requête à la fonction f à chaque itération de la boucle, si l'on désigne par T_f le temps d'exécution de la fonction f , l'algorithme de Grover a une complexité temporelle en $O(\frac{1}{\sqrt{\epsilon}} T_f)$.

3.3.2 Marche aléatoire sur un graphe

Marche aléatoire classique

On se donne un graphe $G = (E, V)$ non-dirigé et d -régulier et un ensemble $M \subset V$. On dit que les éléments de M sont marqués. On pose $N := |V|$ et $\epsilon = \frac{|M|}{N}$.

Pour trouver un élément marqué, on peut faire une marche aléatoire : partant d'un sommet y , tester si y est marqué. S'il l'est, c'est fini, sinon choisir un de ses voisins comme sommet courant.

On peut implémenter une marche aléatoire classique à l'aide d'une chaîne de Markov. On utilise alors la procédure suivante pour trouver un sommet marqué :

Algorithme 5 : Random Walk [Ghazal]-p22

Entrées : $G = (E, V)$ et $M \subset V$

Sorties : un e tel que $e \in M$

1. **SETUP** : Préparer un état initial v (coût S)
2. **pour** $\frac{1}{\epsilon}$ itérations **faire**
3. **CHECK** : **si** Le sommet courant est marqué (coût) **alors**
4. **retourner** le sommet courant
5. **sinon**
6. **répéter** $\frac{1}{\delta}$ fois
7. **UPDATE** : Faire une étape de la marche aléatoire (coût U)
8. **jusqu'à** arriver à une distribution uniforme

Par conséquent la complexité totale de l'algorithme, en fonction de la complexité de ses étapes élémentaires, est :

$$S + \frac{1}{\epsilon} \left(C + \frac{1}{\delta} U \right)$$

Ici comme dans la version quantique, le but sera de faire en sorte que le terme dominant entre parenthèses soit $\frac{1}{\delta} U$. Nous aurons alors dans le cas classique un algorithme en $O(\frac{1}{\delta \epsilon})$.

Marche aléatoire quantique (QW)

La marche aléatoire quantique (Quantum Walk-QW) est une généralisation de l'algorithme de Grover, il est en effet possible de redériver l'algorithme de Grover à partir de la marche aléatoire quantique [Ghazal].

Dans la version quantique de la marche aléatoire, il y a toujours une chaîne de Markov ayant une matrice de transition P , mais les états du système ne sont plus les sommets mais les arêtes que l'on représentera par le produit tensoriel $|x\rangle|y\rangle$ où x est le sommet courant et y le sommet précédent.

On dit qu'une arête $|x\rangle|y\rangle$ est *bonne* lorsque $x \in M$. On définit aussi $|p_x\rangle = \sum_y \sqrt{P(y|x)} |y\rangle$. Comme pour l'algorithme de Grover on définit la superposition uniforme des bonnes et mauvaises arêtes :

$$|G\rangle = \frac{1}{\sqrt{M}} \sum_{x \in M} |x\rangle |p_x\rangle$$

$$|B\rangle = \frac{1}{N-\sqrt{M}} \sum_{x \notin M} |x\rangle |p_x\rangle$$

Ainsi que la superposition uniforme sur tous les états :

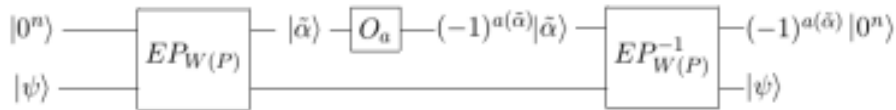
$$|U\rangle := \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |p_i\rangle = \cos(\theta)|G\rangle + \sin(\theta)|B\rangle \text{ avec } \theta = \arcsin(\sqrt{\epsilon}).$$

Les étapes de l'algorithme de marche aléatoire quantique sont les mêmes que pour l'algorithme de Grover, à savoir : préparer $|U\rangle$, faire une boucle où l'on reflète par $|B\rangle$ et ensuite par $|U\rangle$ et mesurer le premier registre à la fin de la boucle. Il y a, pour les mêmes raisons que dans l'algorithme de Grover, $P(\frac{1}{\sqrt{\epsilon}})$ tours de boucles.

La nouveauté est la réflexion par $|U\rangle$ dont nous allons expliquer l'implémentation.

1. Il faut d'abord dériver une matrice unitaire $W(P)$ à partir de la matrice stochastique P . Pour ce faire, on définit $\mathbb{A} := \text{span}\{|x\rangle |p_x\rangle\}$ et $\mathbb{B} := \text{span}\{|p_y\rangle |y\rangle\}$. Soient R_A et R_B les réflexions par \mathbb{A} et \mathbb{B} , i.e. $R_A = 2|x\rangle |p_x\rangle\langle p_x| \langle x| - Id$ et $R_B = 2|p_y\rangle |y\rangle\langle y| \langle p_y| - Id$. On définit $W(P) := R_B R_A$. $W(P)$ est une matrice unitaire dont les valeurs propres dépendent de celles de P de la manière suivante : si on écrit $\lambda_i = \cos(\alpha_i)$ pour les valeurs propres de P , les valeurs propres de $W(P)$ seront de la forme $\mu_i = e^{i2\alpha_i}$. Alors on a en particulier que :
 - $W(P)$ a un vecteur propre de valeur propre 1, car $\lambda_1 = 1 \Rightarrow \alpha_1 = 0 \Rightarrow \mu_1 = e_0 = 1$.
 - Pour tout i , $1 - \delta \geq |\lambda_i| = \cos(\alpha_i) \geq 1 - \frac{\alpha_i^2}{2} \Rightarrow \alpha_i \geq \sqrt{2\delta}$. Par conséquent, il suffit de connaître $\alpha \pm \frac{\sqrt{\delta}}{2}$ pour savoir si $\alpha_i = 1$ ou non. Autrement dit, il faut $n = O(-\log(\sqrt{\delta}))$ bits de précision.
2. On note $EP_{W(P)}$ l'algorithme de l'estimation de phase sur $W(P)$. Rappelons que pour une estimation à n bits de précision, il fera $2^n = O(\frac{1}{\sqrt{\delta}})$ fois appel à $W(P)$. On notera $\tilde{\alpha}$ l'estimation de l'angle α .
3. On note a l'oracle qui, appliqué à un angle α , renvoie 0 si $\alpha = 0$ et 1 sinon.

On peut maintenant définir $R(P) : |U\rangle \mapsto |V\rangle \mapsto -|V\rangle \forall |V\rangle \perp |U\rangle$.



L'algorithme de recherche d'un sommet marqué à l'aide d'une marche aléatoire est alors analogue dans la forme à sa version classique.

Si l'on note :

- S : coût de préparation de $|U\rangle$ (SETUP)

- C : coût de l'opération de vérification (Check) $|x\rangle|y\rangle \mapsto (-1)^{I_M(x)}|x\rangle|y\rangle$
- U : coût de l'opération de mise à jour $W(P)$ (Update) : il s'agit du coût des opérations $|x\rangle|0\rangle \mapsto |x\rangle|p_x\rangle$ et $|p_y\rangle|y\rangle \mapsto |0\rangle|y\rangle$ ainsi que leurs inverses.

La marche aléatoire quantique a la complexité suivante :

$$S + \frac{1}{\sqrt{\epsilon}} \left(C + \frac{1}{\sqrt{\delta}} U \right)$$

De même que dans le cas classique, on cherche à faire en sorte que C soit dominé par $\frac{1}{\sqrt{\delta}} U$, de sorte à avoir une complexité en $O(\frac{1}{\sqrt{\delta\epsilon}})$.

Quantum Walk avec structure de données

Étant donné un ensemble Ω on se donne une fonction $D : \Omega \rightarrow H_D$, qui associe à un élément x de Ω un élément $D(x)$ de l'espace de Hilbert H_D . On appellera structure de données l'ensemble $(D(x))_{x \in \Omega}$. L'élément $D(x)$ dépend souvent d'une fonction $f : \Omega \rightarrow \{0, 1\}$, dans ce cas on écrira $D_f(x)$ pour plus de clarté.

On s'intéresse donc au produit tensoriel $|x\rangle|y\rangle|D(x)\rangle$. Cela change les étapes de l'algorithme de la manière suivante :

- SETUP : Préparer $|U_D\rangle := \frac{1}{\sqrt{N}} \sum_{x,y=0}^{N-1} |x\rangle|p_x\rangle|D(x)\rangle$
- CHECK : $|x\rangle|y\rangle|D(x)\rangle \mapsto (-1)^{I_M(x)}|x\rangle|y\rangle|D(x)\rangle$
- UPDATE : cette étape fera intervenir en plus les fonctions $|x\rangle|0\rangle|D(x)\rangle \mapsto |x\rangle|y\rangle|D(x)\rangle$ et $|x\rangle|y\rangle|D(x)\rangle \mapsto |x\rangle|y\rangle|D(y)\rangle$ et leurs inverses.

Comme nous le verrons, l'intérêt de l'incorporation des structures de données vient principalement du fait qu'elles permettent d'accélérer la vérification et donc de diminuer C . [Ghazal]

3.3.3 Algorithme de Recherche de collisions

Dans cet algorithme il s'agit de répondre au problème suivant :

Problème : (Recherche de collisions généralisée). Étant donné $f_i : S_i \rightarrow E, i = 1, 2, S_1, S_2$ et E des ensembles quelconques, trouver $(s_1, s_2) \in S_1 \times S_2$ tel que $f_1(s_1) = f_2(s_2)$. [Ghazal]

Dans l'algorithme qui résout ce problème à l'aide de l'algorithme de Grover et que l'on nommera *CollisionGrover*, on commence par construire la liste

$L_1 := \{(f_1(s_1), s_1) | s_1 \in S_1\}$ et on la trie.

Ensuite, on définit la fonction $C_{L_1} : S_2 \rightarrow \{0, 1\}$ comme suit :

$$C_{L_1}(s_2) \begin{cases} 1 & \text{s'il existe } s_1 \text{ tel que } (f_2(s_2), s_1) \in L_1 \\ 0 & \text{sinon} \end{cases}$$

Enfin, on se sert de l'algorithme de Grover pour trouver $s_2 \in S_2$ tel que $C_{L_1}(s_2) = 1$ et on cherche le s_1 correspondant et on retourne (s_1, s_2) . On donne le pseudo-code de cet algorithme :

Algorithme : *CollisionGrover* [Ghazal]-p19

Entrées : S_1, S_2, f_1, f_2

Sorties : $(s_1, s_2) \in S_1 \times S_2$ tels que $f_1(s_1) = f_2(s_2)$, NULL s'il n'y a pas de collision

1. $L_1 \leftarrow$ Nouvelle liste
2. Construire et trier la liste L_1
3. Trouver $s_2 \in S_2$ tel que $C_{L_1}(s_2) = 1$ avec l'algorithme de Grover
4. **retourner** (s_1, s_2) ou NULL si aucun tel élément n'a pu être trouvé

Cet algorithme est une complexité totale $\tilde{O}(|S_1| + \sqrt{|S_2|})T_f$ pour l'algorithme de Grover. Si on a un intérêt dans ce cas, il faudrait avoir $|S_1| \ll |S_2|$, le cas optimal étant $|S_1| = O(\sqrt{|S_2|})$. Quant à la complexité en espace, on stocke L_1 qui est de taille $O(|S_1|)$. Par conséquent la complexité en espace est en $O(|S_1|)$.

3.4 Décodage par ensemble d'information (ISD)

Étant donné un code C de paramètres $[n, k]$ et sa matrice de parité H , on dit que $S \subset \{1, \dots, n\}$ est un ensemble d'information si $|S| = k$. Une technique qui est basée sur l'utilisation de tels ensembles pour le décodage du code s'appelle décodage par ensemble d'information ou information set decoding (ISD) en anglais.

Les détails des calculs de complexité et des affirmations autour des paramètres optimaux ainsi que le code pour calculer les courbes de complexité se trouvent dans l'appendice C.[Ghazal]-p29

3.4.1 Algorithme de Prange

Le premier algorithme ISD et le seul algorithme ISD à proprement parler est l'algorithme de Prange : un algorithme probabiliste de type Las Vegas qui échantillonne les ensembles d'information en espérant que le vecteur d'erreur solution e de poids t ait toutes ses positions d'erreurs dans le complémentaire de l'ensemble d'information \bar{S} ($|\bar{S}| = n - k$). Cela réduit

ainsi le problème du décodage à celui de la résolution d'un système linéaire, une tâche de complexité polynomiale.

Version classique

On notera cet algorithme P . [Ghazal]

L'algorithme est constitué d'une boucle sur les ensembles d'information possibles et pour chaque ensemble d'information, on procède comme suit :

1. On note A la sous-matrice de H dont les colonnes sont indicées par \bar{S} .
2. Si A est inversible, on fait $A^{-1}s = e$ et si $w_H(e) = t$, on retourne e .
3. Si une des conditions sur A ou sur e n'est pas vérifiée, on recommence avec un autre ensemble d'information.

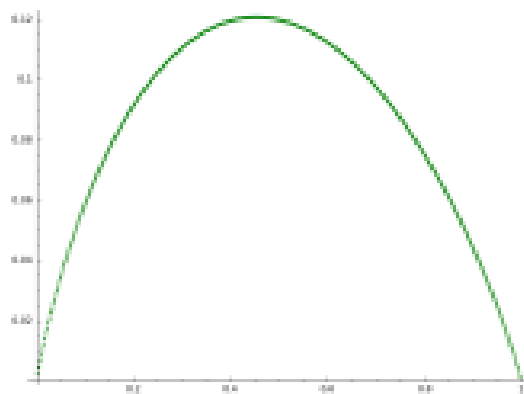
Ainsi le coût des opérations au sein de chaque tour de boucle est polynomial.

On dit qu'un ensemble d'information est bon si le tour de boucle retourne le bon vecteur e , ce qui arrive si la sous-matrice A est inversible (cet événement arrive dans 29% des cas [3]) et que $w_H(e) = t$. Si l'on note P_p la probabilité de tomber sur un bon ensemble d'information, la complexité temporelle de cet algorithme est en $\tilde{O}(\frac{1}{P_p})$.

Théorème 20 Si H est une matrice aléatoire, en notant $R := \frac{k}{n}$ et $\tau = \frac{t}{n}$, on a l'expression suivante pour l'exposant α_p de l'algorithme de Prange :

$$\alpha_p(R) = H(\tau) - (1 - R)H\left(\frac{\tau}{1-R}\right)$$

Et $\max_R(\alpha_p) = 0.1207019$ pour $R = 0.4537$



Courbe de α_p en fonction de R

Démonstration : Rappelons que le vecteur d'erreur devrait avoir la forme suivante : les t positions d'erreur sont parmi les $n - k$ indices de \bar{S} .

Par conséquent $P_p = \frac{\binom{n-k}{t}}{\binom{n}{t}}$, et l'algorithme a une complexité en $\tilde{O}\left(\frac{\binom{n-k}{t}}{\binom{n}{t}}\right)$. Afin d'avoir l'exposant de l'algorithme en fonction de R (et d'autres paramètres). Nous utiliserons le théorème $\binom{n}{t} \approx 2^{nH(\frac{t}{n})}$ et on écrira $R := \frac{k}{n}$ et $\tau = \frac{t}{n}$.

$$\begin{aligned} \frac{\binom{n}{t}}{\binom{n-k}{t}} &\approx \frac{2^{nH(\frac{t}{n})}}{2^{(n-k)H(\frac{t}{n-k})}} \\ &\approx \frac{2^{nH(\tau)}}{2^{n(1-R)H(\frac{\tau}{1-R})}} \\ &\approx 2^{n(H(\tau) - (1-R)H(\frac{\tau}{1-R}))} \end{aligned}$$

On obtient ainsi l'expression suivante pour l'exposant α_p :

$$\alpha_p(R) = H(\tau) - (1-R)H\left(\frac{\tau}{1-R}\right)$$

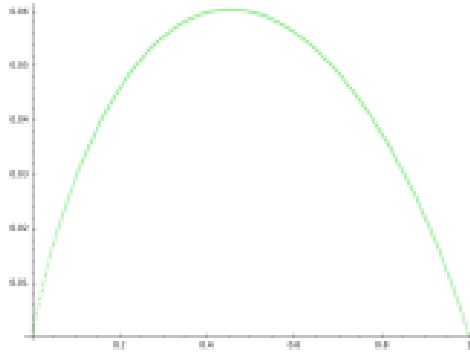
Version quantique avec Grover

On donne ici une version quantique de l'algorithme de Prange, donné pour la première fois dans [3], que l'on notera PG . Dans cet algorithme, on part de la même hypothèse sur la répartition des erreurs (donc, si l'on note P_{PG} la probabilité de tomber sur le bon ensemble d'information, on a $P_{PG} = P_p$). On utilise l'algorithme de Grover pour trouver le bon ensemble d'information (l'oracle pour l'algorithme de Grover s'obtient facilement en modifiant la suite d'étapes détaillée dans la partie précédente pour renvoyer 1 si l'ensemble d'information est bon et 0 sinon). Nous obtenons ainsi un algorithme de complexité temporelle $\tilde{O}\left(\sqrt{\frac{1}{P_{PG}}}\right)$. [Ghazal]

Théorème 21 Si H est une matrice aléatoire, en notant $R := \frac{k}{n}$ et $\tau := \frac{t}{n}$, on a l'expression suivante pour l'exposant α_{PG} l'algorithme de Prange+Grover :

$$\alpha_{PG}(R) = \frac{H(\tau) - (1-R)H(\frac{\tau}{1-R})}{2}$$

Et $\max_R(\alpha_{PG}) = 0.0603509$ pour $R = 0.4537$



Courbe de α_{PG} en fonction de R

Démonstration : L'algorithme a une complexité temporelle en $\tilde{O}\left(\sqrt{\frac{1}{p_{PG}}}\right) = \tilde{O}\left(\sqrt{\frac{\binom{n}{t}}{\binom{n-k}{t}}}\right)$.
On déduit l'expression suivante pour l'exposant α_{PG} :

$$\alpha_{PG}(R) = \frac{H(\tau) - (1-R)H\left(\frac{\tau}{1-R}\right)}{2}$$

3.5 Squelette d'un algorithme ISD

Il y a d'autres algorithmes inspirés de l'algorithme de Prange et qui ont une meilleure complexité. On introduit un paramètre p en plus. Ces algorithmes ISD "généralisés" échantillonnent des ensembles d'indices un peu plus généraux que les ensembles d'informations, en espérant que la majeure partie, $t - p$, des positions d'erreur soit dans le complémentaire de cet ensemble. Ils cherchent ensuite les p positions d'erreur restantes en se servant de stratégies efficaces mais de complexité exponentielle.

Un algorithme ISD comporte donc une phase de recherche d'un ensemble d'indices convenables suivi d'une phase de recherche d'un vecteur d'erreur vérifiant certaines propriétés. Nous allons formaliser cette structure en donnant un squelette pour ces algorithmes et en introduisant un cadre pour l'analyse de leur complexité.

La différence principale entre les algorithmes ISD réside dans la façon dont ils trouvent le vecteur d'erreur. Chaque algorithme A fait appel à une fonction $Recherche_A$ propre à elle, dont la spécification est la suivante :

$Recherche_A : S, H, s, t, p \rightarrow \{e \mid s = He^T\} \cup \{NULL\}$ e est poids p sur S $t - p$ sur \bar{S} où S est un ensemble d'indices, H la matrice de parité du code et s le syndrome de l'erreur cherché.

Nous pouvons maintenant donner le squelette d'un algorithme ISD :

Analyse de complexité Comme nous l'avons dit, un tour de boucle de l'algorithme débouche sur le bon vecteur d'erreur si ses positions d'erreurs sont réparties d'une certaine manière (sa forme exacte dépend de l'algorithme A). Si P_A est la probabilité qu'il ait cette

Algorithm 4 Squelette ISD

[Ghazal]-p32

Require: $H, y, s = Hy^T, t, p$

Ensure: e de poids t tel que $s = He^T$

```
1:
2: repeat
3:   Tirer un ensemble d'indices  $S \subset \{1, \dots, n\}$ 
4:    $e \leftarrow Recherche_A(S, H, s, t, p)$ 
5:
6: until  $w_H(e) = t$ 
7: retourner  $e$ 
```

forme, il faut donc en moyenne P_A^{-1} répétitions de la boucle avant de le trouver. En réalité, il faut également que la sous-matrice indicée par S soit inversible, mais cela ne fait que multiplier le nombre de tours de boucles par un facteur constant. Le temps d'exécution de la fonction $Recherche_A$ dépend de l'algorithme A . On le notera T_A . Cette fonction pourrait également avoir une complexité spatiale non-négligeable que l'on notera S_A .

L'algorithme aura par conséquent une complexité temporelle en $\tilde{O}(P_A^{-1}T_A)$ et une complexité spatiale en S_A . [Ghazal]

Pour simplifier les notations Dans ce qui suit, nous allons considérer une version modifiée de l'algorithme présenté ci-dessus afin de faciliter la présentation des algorithmes. En effet, étant donné un ensemble d'indices S de taille $n - k$ tel que les colonnes indicées par r cet ensemble forment une sous-matrice inversible, on considère $P \in M_{n,m}$ la matrice qui permute les colonnes de H indicées par S avec les dernières colonnes de H . Alors, la matrice HP possède une sous-matrice carrée à droite. On peut alors faire un pivot de Gauss T sur cette matrice afin d'obtenir la matrice $THP := \tilde{H} = [A | Id_{n-k}]$, $A \in M_{n-k,k}$. Alors, si l'on note $\tilde{s} := Ts$ et $\tilde{e} := eP$, on a :

1. Comme P est une permutation, $w_H(\tilde{e}) = w_H(\tilde{e}P^{-1}) = w_H(e) = t$.
2. $\tilde{s} = \tilde{H}\tilde{e}^T \Leftrightarrow Ts = THP(eP)^T = THPP^{-1}e^T \Leftrightarrow Ts = THe^T \Leftrightarrow s = He^T$

Par conséquent, nous allons modifier l'algorithme de la manière suivante : on transforme d'abord H et s en \tilde{H} et \tilde{s} , puis on applique $Recherche_A$ à ces arguments et à l'ensemble \tilde{S} des $|S|$ derniers indices, ce qui fait qu'à la fin de l'algorithme, le résultat est \tilde{e} et on retourne ensuite $\tilde{e}P^{-1} = e$.

Cela étant un point de détail qui permet de faciliter la présentation, afin d'alléger la notation, dorénavant nous écrirons directement $H = [A | Id_{n-k}]$.

3.5.1 Décodage par ensemble d'information quantique

On rappelle la spécification et la complexité en requêtes de l'algorithme de Grover.

Grover : $\epsilon, \{f : E \rightarrow \{0, 1\}\} \rightarrow |X\rangle$, où $|X\rangle$ est une superposition d'états $x \in E$ tel que $f(x) = 1$. Cela s'effectue en $O(\frac{1}{\sqrt{\epsilon}})$ requêtes à la fonction f , où ϵ est la proportion des $x \in E$ tel que $f(x) = 1$.

Étant donné une matrice H , un syndrome s , des entiers t et p et un algorithme de décodage générique A , on définit la fonction $F_A : S \rightarrow \{0, 1\}$ comme la fonction qui exécute $RechercheA(S, H, s, t, p)$ et renvoie 1 si le vecteur renvoyé n'est pas NULL et est de poids t , et 0 sinon.

Alors, pour trouver une erreur de poids t tel que $s = He^T$, il suffit d'utiliser l'algorithme de Grover sur F_A pour obtenir un ensemble d'indices S . Il suffit ensuite d'appliquer $RechercheA$ avec cet ensemble d'indices.

Ainsi, un algorithme ISD quantique a la forme générale suivante :

Algorithme 5 : Squelette ISD quantique [Ghazal], p32

Entrées : $H, y, s = Hy^T, t, p$
 Sorties : e de poids t tel que $s = He^T$

1. $|S\rangle \leftarrow Grover(P_A, F_A)$
2. $|S_0\rangle \leftarrow Mesure(|S\rangle)$
3. $e \leftarrow Recherche_A(S_0, H, s, t, p)$
4. **retourner** e

La fonction F_A a la même complexité temporelle et spatiale que $Recherche_A$, c'est-à-dire respectivement $\tilde{O}(T_A)$ et S_A . L'algorithme de Grover trouve S au bout de $O(\sqrt{P_A^{-1}})$

requêtes à F_A , d'où une complexité temporelle totale de $\tilde{O}(\sqrt{P_A^{-1}} T_A)$. La complexité de l'étape 3 est $\tilde{O}(T_A)$ et donc la complexité temporelle totale d'un algorithme ISD quantique est $\tilde{O}(\sqrt{P_A^{-1}} T_A)$ (et sa complexité spatiale est $\tilde{O}(S_A)$).

Nous finissons par une remarque générale sur les algorithmes ISD : dans les algorithmes classiques A autres que Prange, $P_A^{-1} \leq P_p^{-1}$ mais $T_A \gg P_p$.

En effet, T_p est polynomial, mais dans tous les autres algorithmes T_A est exponentiel. Les algorithmes qui ont battu l'algorithme de Prange classique ont réussi à avoir $P_A^{-1} T_A < P_p^{-1}$. Pour battre l'algorithme de Prange quantique, on cherche des algorithmes A tels que $q \sqrt{P_A^{-1} T_A} = \sqrt{P_A^{-1} T_A^2} < \sqrt{P_{PG}^{-1}}$. On va donc se focaliser sur l'algorithme RechercheA et les diverses façons de l'améliorer en se servant d'algorithmes quantiques. De plus, en regardant la formule de complexité $\sqrt{P_A^{-1} T_A}$, on se rend compte qu'il faut des améliorations bien plus importantes qu'en classique pour battre P_G .

3.6 Les algorithmes d'attaques ISD et ses dérivés

Depuis l'apparition du cryptosystème de McEliece en 1978, la sécurité a fait l'objet d'un certain nombre d'études d'un point de vue classique et quantique. Des efforts considérables ont été déployés dans le but de résoudre le problème du décodage par syndrome :

Par exemple, Esser et Bellini [Belestr5] ont proposé un estimateur, qui est un calcul plus réaliste des algorithmes ISD.

Narisada et al. introduisent, dans [Narisada15], des études sur le décodage des SDP à haute dimension en parallélisant des parties de l'algorithme ISD.

Cependant, il n'y a pas beaucoup d'études du point de vue quantique.

-En 2010, Bernstein [Bergrover3] a proposé l'algorithme ISD quantique.

- Perriello et al [Perriello18] ont proposé une attaque en utilisant l'algorithme de Bernstein pour BIKE et Classic McEliece. Ils ont ensuite amélioré cette attaque en utilisant la version quantique de l'algorithme de Lee-Brickell [Leebr10], qui est l'un des algorithmes de la DSI.

- Esser et al [Essramos6] ont également proposé une autre méthode d'attaque en utilisant l'algorithme de Bernstein, qui est l'un des algorithmes ISD et l'ont étendue à tous les candidats CBC lors du 4e tour du NIST PQC, y compris le HQC.

- En 2017 Kachigar et Tillich [Kati9] ont utilisés les algorithmes MMT [Mmt11], Grover, Quantum Walk, Shamir-Schroeppel quantiques pour réduire davantage l'exposant de Prange .

Malgré toute cette énergie consacrée, les meilleurs algorithmes ne donnent qu'une résolution exponentielle pour retrouver le vecteur d'erreurs e de poids de Hamming w pour un code binaire linéaire de longueur n et de dimension k , avec un coût approximatif de $\tilde{O}\left(2^{\alpha\left(\frac{k}{n}, \frac{w}{n}\right)n}\right)$ où $\alpha(R, w)$ est positif quand R et w sont positifs.

Nonobstant les recherches ne sont pas vaines car celles-ci ont permis de réduire légèrement l'exposant $\alpha(R, w)$.

Le tableau suivant donne un aperçu de la complexité temporelle moyenne des algorithmes classiques existants lorsque w est la distance de Gilbert-Varshamov $d_{GV}(n, k)$ du code telle que $d_{GV}(n, k) \stackrel{\Delta}{=} nH_2^{-1}\left(1 - \frac{k}{n}\right)$ où

$H_2 \stackrel{\Delta}{=} -x \log_2(x) - (1-x) \log_2(1-x)$ est la fonction d'entropie binaire et son inverse H_2^{-1} définie de $[0, 1]$ dans $\left[0, \frac{1}{2}\right]$. Il correspond à la plus grande distance pour laquelle nous pouvons encore espérer une solution unique au problème du décodage. Si nous voulons que la solution soit unique, elle peut donc être considérée comme l'instance de décodage la plus difficile.

Dans le tableau suivant, w_{GV} est défini par le rapport $w_{GV} \stackrel{\Delta}{=} d_{GV}(n, k)/n$.

La question de l'utilisation d'algorithmes quantiques pour accélérer les algorithmes de décodage de la ISD a été posée pour la première fois dans [Over22]. Cependant, la façon dont

Author(s)	Year	$\max_{0 \leq R \leq 1} \alpha(R, w_{GV})$ to 4 dec. places
Prange [23]	1962	0.1207
Dumer [Dumer11]	1991	0.1164
MMT [Mmt11]	2011	0.1114
BJMM [Bjmm2]	2012	0.1019
MO [Mo19]	2015	0.0966

TABLE 3.3 – Les algorithmes ISD classiques

Author(s)	Year	Ingredients	$\max_{0 \leq R \leq 1} \alpha(R, w_{GV})$ to 4 dec. places
Bernstein[Berst5]	2010	Prange + Gr ¹	0.06035
Kachigar- Tillich	2017	SS ² +Gr+QW ³	0.05970
Kachigar- Tillich	2017	MMT+“1+1=0”+Gr+QW	0.05869

TABLE 3.4 – Les algorithmes ISD quantiques

l’algorithme de Grover a été utilisé dans ,[Over22] sous-section 3.5] pour accélérer le décodage n’a pas permis d’obtenir des améliorations significatives par rapport aux algorithmes ISD classiques. Plus tard, Bernstein a montré dans [Berst5] qu’il était possible d’obtenir des accélérations bien plus importantes avec l’algorithme de Grover : en l’utilisant pour trouver un ensemble d’informations sans erreur, l’exposant de l’algorithme de Prange peut en effet être réduit de moitié.

Dans un article publié le 22 avril 2017 [Kati9], Kachigar et Tillich s’appuient sur l’algorithme de recherche de Grover, ainsi que sur les algorithmes quantiques développés par Bernstein, Jeffery, Lange et Meurer dans [Bernstein6] pour résoudre plus efficacement le problème de la somme des sous-ensembles. Le tableau suivant résume les ingrédients et la complexité temporelle moyenne de l’algorithme de [Berst5] et des nouveaux algorithmes quantiques présentés dans cet article.

En faisant une analogie on voit que l’exposant de complexité de l’algorithme MMTQW apporte une amélioration d’une manière faible par rapport au meilleur algorithme ISD classique MO.

3.6.1 Les Algorithmes de Prange et de Bernstein

Rappelons que l’objectif est de trouver e de poids w étant donné $s^T = He^T$ où H est une matrice $(n - k) \times n$. En d’autres termes, le problème que nous cherchons à résoudre consiste à trouver une solution à un système linéaire à $n - k$ équations avec n variables et la solution est unique en raison de la condition de poids.

L’algorithme de Prange est basé sur l’observation suivante : si l’on sait que k composantes données du vecteur d’erreur sont nulles, les positions d’erreur se trouvent parmi les $n - k$

composantes restantes. En d'autres termes, si l'on est sûr que les k variables correspondantes ne sont pas impliquées dans le système linéaire, le vecteur d'erreur peut être trouvé en résolvant le système linéaire résultant de $n-k$ équations à $n-k$ variables en $n-k$ variables en temps polynomial.

La difficulté consiste à trouver un ensemble de taille k correct (d'indices des composants). L'algorithme de Prange échantillonne de tels ensembles et résout l'équation linéaire résultante jusqu'à ce qu'un vecteur d'erreur de poids w soit trouvé. La probabilité de trouver un tel ensemble est de l'ordre de $\Omega\left(\frac{\binom{n-k}{w}}{\binom{n}{w}}\right)$ et l'algorithme de Prange a donc une complexité

$$O\left(\frac{\binom{n}{w}}{\binom{n-k}{w}}\right) = \tilde{O}\left(2^{\alpha_{prange}(R,w)n}\right)$$

tel que

$$\alpha_{prange}(R, w) = H_2(w) - (1 - R)H_2\left(\frac{w}{1-R}\right)$$

en utilisant la formule binomiale connue

$$\binom{n}{w} = \tilde{O}\left(2^{H_2\left(\frac{w}{n}\right)n}\right)$$

L'algorithme de Bernstein consiste à utiliser l'algorithme de Grover pour trouver un ensemble de taille- k correct. En effet, un oracle permettant de vérifier qu'un ensemble de taille- k est correct peut être obtenu en suivant les mêmes étapes que dans l'algorithme de Prange, c'est-à-dire en dérivant et en résolvant un système linéaire de $n-k$ équations à $n-k$ variables et en retournant 1 si le vecteur d'erreur résultant a un poids w . Ainsi, la complexité de l'algorithme de Bernstein est la moitié de celle de l'algorithme de Prange,

$$\text{i.e. } \alpha_{Bernstein} = \frac{\alpha_{Prange}}{2}$$

Parmi les algorithmes ISD ci-dessus certains ont une version classique et quantique (avec imbrication dans leur squelette ISD l'algorithme de Grover quantique et la marche aléatoire sous forme de fonction d'où nous devons prendre compte implicitement leur complexité dans l'implémentation).

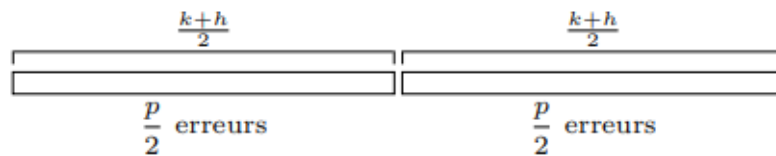
De plus ils peuvent être classés suivant les ISD avancés avec technique des représentations et les ISD avancés avec techniques de représentations étendues cette nouvelle donne est obtenue avec la somme de sous-ensemble (subset-sum).

3.6.2 Les algorithmes ISD avancés avec technique de représentation

Beaucoup d'algorithmes n'ont pas permis de réduire la complexité temporelle asymptotique de l'algorithme de Dumer à l'instar de l'algorithme de Shamir-Schroeppel . C'est l'utilisation de la technique des représentations, une technique qui a été d'abord utilisée pour accélérer la résolution du subset sum, qui va permettre d'accélérer la recherche de collisions. L'algorithme de May, Meurer et Thomae ([12], [1]), que nous allons décrire dans cette section et que nous noterons MMT, a été le premier algorithme de décodage à utiliser cette technique.[Ghazal]-p45

Considérations préliminaires

1. On rappelle que l'on avait supposé, pour simplifier les notations, que $H = [A|I_{n-k}]$. Notons qu'on peut également écrire la matrice H sous la forme $H = \begin{bmatrix} H'|O_{h,n-k-h} \\ B|I_{n-k-h} \end{bmatrix}$ avec $H' \in \mathcal{M}_{h,k+h}$, $B \in \mathcal{M}_{n-k-h,k+h}$. On considère plus précisément le code C' obtenu en poinçonnant les $n-k-h$ dernières colonnes de $G := [I_k|A^T]$, qui est une matrice génératrice de C , la matrice génératrice G' de C' est alors donnée par la sous-matrice de G restreinte aux $k+h$ premières colonnes. Alors, d'après le théorème 11, la matrice H' est une matrice de parité du code C' . Qui plus est, puisque G' admet une sous-matrice inversible (la matrice identité), alors le théorème 10 entraîne que si l'on note $c := mG$, il est possible d'obtenir $c' := m_{|k+h}G'$, à partir de c via l'opération de poinçonnage et inversement, il est possible de trouver c à partir de c' de manière unique. On aura alors $H'c'^T = 0$. On est donc ramené à un problème de décodage par syndrome sur une matrice de taille plus petite, plus précisément une matrice $(h+k) \times h$. Nous appellerons ce problème le problème de décodage par collision, d'après la technique principale qui sera utilisée pour le résoudre. L'intérêt de se ramener à ce problème est que, bien qu'il y ait moins d'équations de parité (car h est petit), le nombre de motifs d'erreurs possibles a diminué car on se restreint aux mots de codes de poids p du code C' .
2. De plus, pour optimiser la recherche de collisions, on fera l'hypothèse que les p erreurs sont réparties de la manière suivante dans la fenêtre de taille $k+h$:



3. Il est possible de décrire autrement le problème du décodage par collision, à savoir avec des ensembles d'indices. En effet, si l'on note M_l la $l_{\text{ème}}$ colonne d'une matrice

M , alors les deux problèmes suivants sont équivalents :

- (a) : Trouver e de longueur n tel que $s = He^T$ et $w_H(e) = t$.
- (b) Trouver $I \subset \{1, \dots, n\}, |I| = t$, tel que $s = \sum_{l \in I} H_l$.

La deuxième formulation met en lumière des similitudes fortes avec le problème de la somme des sous-ensembles (subset sum) : le problème du décodage par collision peut être vu comme une version vectorielle du problème subset sum. Nous allons en effet commencer à nous servir de cette autre formulation.

Dans la suite on fixe les notations suivantes :

$h = h_1 + h_2$ et on va *diviser* H' en deux sous-matrices ayant h_1 et h_2 lignes chacune. Pour simplifier les notations, nous allons définir $H'^{(1)}$ comme étant la matrice H' restreinte aux h_1 premières lignes et $H'^{(2)}$ comme étant la matrice H' restreinte aux lignes $h_1 + 1$ jusqu'à $h_1 + h_2 = h$. Ainsi :

$$H' = \begin{bmatrix} H'^{(1)} \\ H'^{(2)} \end{bmatrix}$$

De même, pour un vecteur colonne v de longueur h , on notera $v^{(1)}$ et $v^{(2)}$.

y' est le vecteur poinçonné de y

Algorithme MMT classique

Technique des représentations [Ghazal]

1. Dans l'algorithme de Dumer comme dans l'algorithme de Shamir-Schroeppel, nous avons considéré l'ensemble I des indices des composantes non-nulles du vecteur d'erreur e et nous l'avons écrit sous la forme $I = I_1 \sqcup I_2$ avec $I_1 \subset \left\{1, \dots, \frac{k+h}{2}\right\}$, $I_2 \subset \left\{\frac{k+h}{2} + 1, \dots, k+h\right\}$. On appelle (I_1, I_2) une représentation de l'ensemble I . L'idée de l'algorithme de May, Meurer et Thomae est de relâcher les contraintes sur (I_1, I_2) afin d'avoir un nombre bien plus grand de représentations. En particulier, nous n'exigerons plus que $\frac{t}{2}$ erreurs soient réparties dans chaque moitié du vecteur d'erreur. En faisant cela, on augmente de façon importante la probabilité de tomber sur le bon vecteur d'erreur. Mais alors, un nouveau problème surgit : en augmentant le nombre de représentations, nous augmentons aussi la taille des listes. Nous verrons qu'il est possible de choisir les paramètres de sorte à avoir une seule représentation qui subsiste, ce qui diminue la taille des listes, mais garde l'avantage procuré par la multiplicité des représentations. Cela nous permettra de réduire aussi bien la complexité en l'espace qu'en temps.

2. L'algorithme de May, Meurer et Thomae repose donc sur la méthode des représentations pour améliorer les complexités en temps et en espace. Mais il y a aussi un deuxième niveau d'amélioration dans la mesure où nous nous servons d'une recherche de collisions pour trouver les bonnes représentations. Cela réintroduit des contraintes sur la forme du vecteur d'erreur, mais comme nous le verrons plus tard, ces contraintes font perdre au plus un facteur polynomial.

Description de l'algorithme [Ghazal]

On rappelle que l'ensemble $S = \{I \subset \{1, \dots, h+k\} \mid |I| = p\}$ est isomorphe à l'ensemble des vecteurs d'erreur de poids p et de longueur $k+h$. On cherche à représenter l'ensemble I' des coordonnées non-nulles du vecteur d'erreur comme $I_1 \sqcup I_2$ avec $I_i \subset \{1, \dots, k+h\}, |I_i| = \frac{p}{2}$ pour $i = 1, 2$. Il y a $\binom{p}{p/2}$ telles représentations.

On prend :

$$\begin{cases} J_{1,1} = J_{2,1} & \left\{1, \dots, \frac{k+h}{2}\right\} \\ J_{1,2} = J_{2,2} & \left\{\frac{k+k}{2} + 1, \dots, k+h\right\} \end{cases}$$

Alors $S_{i,j} = \left\{I_{i,j} \subset J_{i,j} \mid |I_{i,j}| = \frac{p}{4}\right\}$ et $|S_{i,j}| = \binom{\frac{k+h}{2}}{\frac{p}{4}} \approx \left(\frac{k+h}{2}\right)^{1/2}$.

On construit les listes $L_{i,j}$ à partir des $S_{i,j}$, puis on construit les listes L_i à partir des listes $L_{i,j}$, en considérant l'union exclusive \sqcup des ensembles d'indices. Par contre pour construire L à partir des L_i , nous prendrons des ensembles d'indices $I' = I_1 \triangle I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$, puisqu'il n'y a aucune garantie que les $I_i \in L_i$ soient disjoints.

Cette façon de construire les choses entraîne que I' ne s'écrit tout à fait comme $I_1 \cap I_2$ avec $I_i \subset \{1, \dots, k+h\}, |I_i| = \frac{p}{2}$, car avec cette construction, les I_i s'écrivent comme $I_{i,1} \sqcup I_{i,2}$.

L'intérêt de la méthode des représentations : en augmentant le nombre de façons de représenter un ensemble d'indices comme union de sous-ensembles, on arrive à réduire la taille des vecteurs cherchés par un facteur 2^{h_1} , ce qui réduit la taille des listes de recherche et améliore donc la complexité, tout garantissant qu'il suffit de répéter un nombre constant de fois le procédé avant de tomber sur la bonne réponse.

La construction des listes se fait suivant l'algorithme *DecodeCollision*_{MMT} ci-dessous en utilisant le sous programme merge-join

Algorithme 6 : DecodeCollision_{MMT} [Ghazal]

Entrées : $H', y', s' = H' y'^T, p, h_1, h_2, r$ vecteur de longueur h tel que $r^{(2)} = 0, (S_{i,j})_{i,j=1,2}$

Sorties : une liste L d'ensembles $I', |I'| = p$ tels que $\sum_{l \in I'} H'_l = s'$

1. $r_1 \leftarrow r$
2. $r_2 \leftarrow s' - r$
3. Initialiser les listes $L_{1,1}, L_{1,2}, L_{2,1}, L_{2,2}$
4. **pour** $i = 1, 2$ **faire**
5. **pour** $I_{i,1} \in S_{i,1}$ **faire**
6. $L_{i,1} \leftarrow L_{i,1} \cup (\sum_{l \in I_{i,1}} H_l^{(1)}, I_{i,1})$
7. **pour** $I_{i,2} \in S_{i,2}$ **faire**
8. $L_{i,2} \leftarrow L_{i,2} \cup (r_i^{(1)} + \sum_{l \in I_{i,2}} H_l^{(1)}, I_{i,2})$
9. $L_i(r) \leftarrow L_{i,1} \bowtie_{h_1}^{r_i} L_{i,2} (*)$
10. $L \leftarrow L_1(r) \bowtie_{h_2}^{s'} L_2(r) (**)$
11. **retourner** L

$(*)(L_{i,1} \bowtie_{h_1}^{r_i} L_{i,2} := ((\sum_{l \in I_i} H_l^{(2)}, I_i) | I_i := I_{i,1} \sqcup I_{i,2} \text{ tel que } \sum_{l \in I_{i,1}} H_l^{(1)} = r_i^{(1)} + \sum_{l \in I_{i,2}} H_l^{(1)}))$

$(**) L_1(r) \bowtie_{h_2}^{s'} L_2(r) = (I' | I' := I_1 \Delta I_2 \text{ } | I' | = p \text{ tel que } \sum_{l \in I_1} H_l^{(2)} = s'^{(2)} + \sum_{l \in I_2} H_l^{(2)})$

Preuve de correction de DecodeCollision_{MMT}

- La liste L_1 contient les $I_1 = I_{1,1} \sqcup I_{1,2}$ tels que $r^{(1)} = r_1^{(1)} = \sum_{l \in I_{1,1}} H_l^{(1)} + \sum_{l \in I_{1,2}} H_l^{(1)} = \sum_{l \in I_1} H_l^{(1)}$.
- La liste contient les $I_2 = I_{1,2} \sqcup I_{2,2}$ tels que $s'^{(1)} - r^{(1)} = r_2^{(1)} = \sum_{l \in I_{2,1}} H_l^{(1)} + \sum_{l \in I_{2,2}} H_l^{(1)} = \sum_{l \in I_2} H_l^{(1)}$.
- On a $|I_i| = \frac{p}{2}$ pour $i = 1, 2$ par construction car $I_{i,1} \cap I_{i,2} = \emptyset$. De plus, $I' = I_1 \Delta I_2$ vérifie $|I'| \leq p$
- La liste L contient les $I' = I_1 \Delta I_2$ tels que :

$$\begin{cases} I_1 \in L_1, \text{ i.e. } \sum_{l \in I_1} H_l^{(1)} = r^{(1)} \\ I_2 \in L_2, \text{ i.e. } \sum_{l \in I_2} H_l^{(1)} = s'^{(1)} - r^{(1)} \\ \sum_{l \in I_1} H_l^{(2)} + \sum_{l \in I_2} H_l^{(2)} = s'^{(2)} \\ |I'| = p, \text{ i.e. } I_1 \cap I_2 = \emptyset \end{cases}$$

$$\Rightarrow \begin{cases} \sum_{l \in I_1} H_l^{(1)} + \sum_{l \in I_2} H_l^{(1)} = s'^{(1)} \\ \sum_{l \in I_1} H_l^{(2)} + \sum_{l \in I_2} H_l^{(2)} = s'^{(2)} \\ |I'| = p \end{cases}$$

$$\Rightarrow \sum_{l \in I'} H'_l = s' \text{ et } |I'| = p$$

Par conséquent la liste de sortie \mathbf{L} est bien comme on souhaite.

Analyse de complexité [Ghazal]

1. On parcourt $|S_{i,j}|$ éléments pour construire la liste $L_{i,j}$ pour $i, j = 1, 2$.
 - Complexité en temps : $\tilde{O}(|S_{i,j}|)$
 - Complexité en espace : $|L_{i,j}| = |S_{i,j}| \Rightarrow \tilde{O}(|S_{i,j}|)$
2. Pour construire L_1 et L_2 , on fait un Merge-Join sur des listes de taille $|L_{i,j}|$ contenant des vecteurs binaires de longueur h_1 .
 - Complexité en temps : $\tilde{O}(\max(|L_{i,j}|, |L_{i,j}|^2 2^{-h_1}))$
 - Complexité en espace : $|L_i| = \tilde{O}(|L_{i,j}|^2 2^{-h_1}) \Rightarrow \tilde{O}(\max(|L_{i,j}|, |L_i|))$
3. Pour construire L , on fait un Merge-Join sur des listes de taille $|L_i|$ contenant des vecteurs binaires de longueur h_2 .
 - Complexité en temps : $\tilde{O}(\max(|L_i|, |L_i|^2 2^{-h_2})) = \tilde{O}(\max(|L_{i,j}|^4 2^{-2h_1-h_2}, |L_{i,j}|^2 2^{-h_1}))$
 - - Complexité en espace : on peut réutiliser de l'espace mémoire $\Rightarrow O(1)$.

Complexité

Théorème 22 [Ghazal] Si H est une matrice aléatoire, en notant $R := \frac{k}{n}, \tau := \frac{t}{n}, \pi := \frac{p}{n}$ et $\eta := \frac{h}{n}$, on a l'expression suivante pour l'exposant α_{MMT} de l'algorithme de MMT :

$$\alpha_{MMT}(R) = \min_{\pi, \eta} (\beta(R, \pi, \eta) + \max_{i=1,2,3} \gamma_i(R, \eta, \pi)) \text{ avec}$$

$$\beta(R, \pi, \eta) = H(\tau) - (1 - R - \eta)H\left(\frac{\tau - \pi}{1 - R - \eta}\right) - (R + \eta)H\left(\frac{\pi}{R + \eta}\right)$$

$$\gamma_1(R, \eta, \pi) = \frac{R + \eta}{2} H\left(\frac{\pi}{2(R + \eta)}\right)$$

$$\gamma_2(R, \eta, \pi) = (R + \eta)H\left(\frac{\pi}{2(R + \eta)}\right) - \pi$$

$$\gamma_3(R, \eta, \pi) = 2(R + \eta)H\left(\frac{\pi}{2(R + \eta)}\right) - \pi - \eta$$

Soumise aux contraintes :

$$0 \leq \pi \leq \min(\tau, \eta) \leq R + \eta$$

$$0 \leq \eta \leq 1 - R - \tau + \pi \leq 1 - R$$

Et $\max_R(\alpha_{MMT}) = 0.1114837$ pour $R = 0.4397$, valeur obtenue pour $\pi \approx 0.021447363$ et $\eta \approx 0.0752051973$

Complexité en temps de $\text{DecodeCollision}_{MMT}$

$$\tilde{O}(\max(|S_{i,j}|, |S_{i,j}|^2 2^{-h_1}, |S_{i,j}|^4 2^{-2h_1-h_2})) = \tilde{O}\left(\max\left(\binom{k+h}{\frac{p}{2}}^{\frac{p}{2}}, \binom{k+h}{\frac{p}{2}} 2^{-h_1}, \binom{k+h}{\frac{p}{2}}^2 2^{-2h_1-h_2}\right)\right)$$

Complexité en espace de $DecodeCollision_{MMT}$

$$\tilde{O}(\max(|S_{i,j}|, |S_{i,j}|^2 2^{-h_1})) = \tilde{O}\left(\max\left(\left(\frac{k+h}{2}\right)^{\frac{1}{2}}, \left(\frac{k+h}{2}\right) 2^{-h_1}\right)\right)$$

Algorithme de recherche MMT Cet algorithme est décrit comme ci-dessous en se basant sur la recherche d'un bon vecteur

Algorithme 7 : $Recherche_{MMT}$ [Ghazal]

Entrées : $H, y, s = Hy^T, t, p, h_1, h_2, (S_{i,j})_{i,j=1,2}$

Sortie : e tel que $y = mG + e$ et $w_H(e) = t$, NULL si aucun tel vecteur n'a pu être trouvé

1. Préparer G', H', y' et calculer $s' = H'y'^T$
2. **pour** tous les vecteurs r de longueur h tel que $r^{(2)} = 0$ **faire**
3. $L \leftarrow DecodeCollision_{MMT}(H', y', s', p, h_1, h_2, r, (S_{i,j})_{i,j=1,2})$
4. **pour** $I' \in L$ **faire**
5. $e \leftarrow Depoinçonner(I')$
6. **si** $w_H(e) = t$ **alors**
7. **retourner e**
8. **retourner NULL**

$Recherche_{MMT}$ fait appel à l'algorithme $DecodeCollision_{MMT}$ pour obtenir la liste L à partir de laquelle l'ensemble d'indices I' est depoinçonné pour un vecteur e s'il est de poids t sinon l'algorithme retourner $NULL$ ceci est bouclé sur 2^{h_1} éléments au maximum avec $h_1 = \log_2\left(\frac{p}{2}\right)$

la complexité en temps de $Recherche_{MMT}$ est la même que celle de $DecodeCollision_{MMT}$ à savoir :

$$T_{MMT} = \tilde{O}\left(\max\left(\left(\frac{k+h}{2}\right)^{\frac{1}{2}}, \left(\frac{k+h}{2}\right) 2^{-p}, \left(\frac{k+h}{2}\right)^2 2^{-2p-(h-p)}\right)\right)$$

On en déduit la complexité de l'algorithme :

$$\tilde{O}\left(\frac{\binom{n}{t}}{\binom{k+h}{p} \binom{n-k-h}{t-p}} \max\left(\left(\frac{k+h}{2}\right)^{\frac{1}{2}}, \left(\frac{k+h}{2}\right) 2^{-p}, \left(\frac{k+h}{2}\right)^2 2^{-2p-(h-p)}\right)\right)$$



L'intérêt de la méthode des représentations : en augmentant le nombre de façons de représenter un ensemble d'indices comme union de sous-ensembles, on arrive à réduire la taille des vecteurs cherchés par un facteur 2^{h_1} , ce qui réduit la taille des listes de recherche et améliore donc la complexité, tout garantissant qu'il suffit de répéter un nombre constant de fois le procédé avant de tomber sur la bonne réponse.

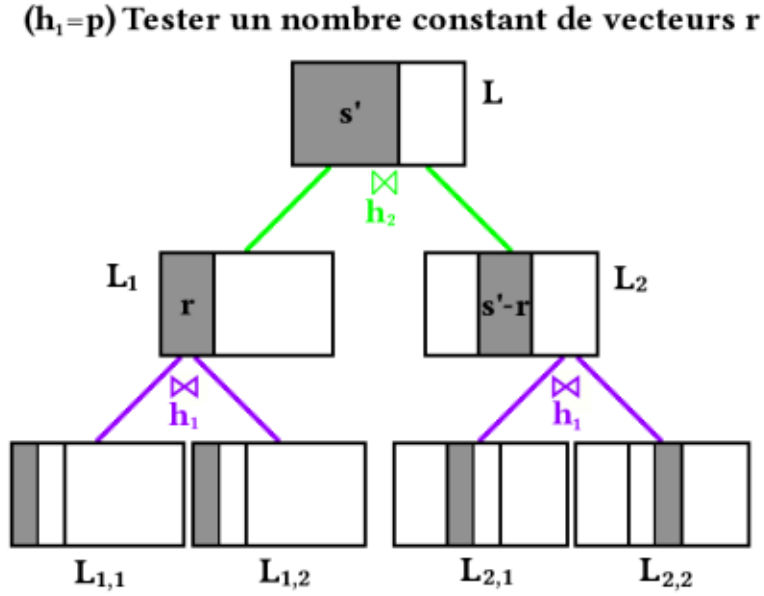


FIGURE 3.1 – *Recherche_{MMT}*- illustration sous forme d'arbre

Violet : recherche de collisions classiques

vert : technique des représentations

⋈ : Merge-Join

[Ghazal]

Algorithme de May, Meurer et Thomae quantique avec Quantum Walk

Dans cette partie il s'agit de résoudre le problème du décodage par collision à l'aide d'une marche aléatoire quantique. En effet, nous pouvons reformuler le problème que cherchent à résoudre les fonctions RechercheSS et RechercheMMT comme le problème de recherche de 4-uplet suivant :

Problème 7 [Ghazal] (Décodage comme recherche de 4-uplet). Soit

$$f : S_{1,1} \times S_{1,2} \times S_{2,1} \times S_{2,2} \longrightarrow \mathbb{F}_2^h$$

$$f : (I_{1,1}, I_{1,2}, I_{2,1}, I_{2,2}) \mapsto \sum_{i,j=1,2} \sum_{l \in I_{i,j}} H'_l + s$$

Trouver $4-uplet(I_{1,1}, I_{1,2}, I_{2,1}, I_{2,2})$ tel que $f(I_{1,1}, I_{1,2}, I_{2,1}, I_{2,2}) = 0$ et $|\cup_{i,j=1,2} I_{i,j}| = p$.

Or, nous avons vu qu'il peut être intéressant d'utiliser une marche aléatoire pour résoudre des problèmes qui admettent ce type de formulations. C'est effectivement le cas : comme nous le verrons, si certaines conditions sont réunies, la marche aléatoire quantique permet de contourner le problème du goulot d'étranglement qui se posait avec l'algorithme de Grover. De plus, la recherche du bon vecteur d'erreur parmi les éléments de la liste finale peut se faire au sein de la marche aléatoire, ce qui évite le recours à des recherches imbriquées qui augmentent la complexité comme dans DG

Squelette d'un algorithme ISD avancé avec Quantum Walk

La marche aléatoire et son graphe [Ghazal] La solution cherchée est un $4-uplet(I_{1,1}, I_{1,2}, I_{2,1}, I_{2,2}) \in S_{1,1} \times S_{1,2} \times S_{2,1} \times S_{2,2}$. Pour éviter de parcourir tout l'espace de recherche, on va considérer quatre sous-ensembles $T_{i,j}$ des ensembles $S_{i,j}$, tous de même taille $T \leq |S_{i,j}|$.

A l'aide de ces identifications, on considère donc une marche aléatoire sur le graphe $J(4 | S_{i,j} |, 4T)$ pour trouver un $4-uplet$ convenable pour une fonction f_A que l'on définira ci-dessous. Il s'agit de la marche $QW(J(4 | S_{i,j} |, 4T), f_A)$ en reprenant une notation introduite dans la section sur l'algorithme quantique.

On a besoin de définir ou expliciter les éléments suivants :

1. Le trou spectral $\delta = \frac{4|S_{i,j}|}{4T(4|S_{i,j}|-4T)} = \Omega(T^{-1})$
2. La fonction f_A à calculer et la structure de données associée.
3. L'ensemble des éléments marqués M_{f_A} , ce qui donnera la proportion ϵ des éléments marqués.
4. Le coût S de Setup.
5. Le coût C de Check.
6. Le coût U de Update.

La fonction f_A et la structure de données associée La fonction f_A a à peu de choses près la même structure que les fonctions $DecodeCollision_A$ des algorithmes ISD avancés.

En effet, elle calcule des structures de données ordonnées contenant des couples " (\sum_J, J) ", où J désigne un ensemble d'indices, sur trois niveaux, où celles du niveau le plus bas sont générées directement et celles du niveau suivant sont obtenues à partir de celles du niveau précédent en faisant un Merge-Join et en exigeant une égalité à un vecteur d'abord sur h_1 coordonnées et ensuite sur les h_2 coordonnées restantes.

Ainsi, la fonction f_A crée quatre structures de données $D_{f,i,j}, i = 1, 2$ au dernier niveau, puis

deux au niveau suivant, D_{f_1} et D_{f_2} . Ces structures de données sont ordonnées en fonction du premier élément du couple. Enfin, au niveau final, une dernière structure de données D est créée à partir de D_{f_1} et D_{f_2} . Dans cette structure de données, toutes les sommes sont égales et donc on peut se contenter de ne stocker que l'ensemble d'indices J .

Ce qui change des fonctions $DecodeCollision_A$, c'est que tout d'abord, f_A stocke en parallèle les ensembles d'indices séparément à chaque niveau dans des structures de données ordonnées. Elle crée ainsi quatre structures de données $D_{I_{i,j}}, i, j = 1, 2$ au niveau le plus bas, puis deux, D_{I_1} et D_{I_2} au niveau suivant.

Par conséquent, la structure de données associée à la fonction f_A comporte ainsi treize sous-structures de données $(D_{I_{i,j}})_{i,j=1,2}, (D_{f_{i,j}})_{i,j=1,2}, (D_{I_i})_{i=1,2}, (D_{f_i})_{i=1,2}, D$.

Enfin, f_A diffère des fonctions $DecodeCollision_A$ dans la mesure où, après avoir créé la structure de données finale D , elle cherche s'il existe $I' \in D$ tel que, en y appliquant l'opération *Dépoïnçonner*, on tombe sur un vecteur d'erreur e de poids t . Elle renvoie 0 si c'est le cas, 1 sinon.

La fonction f_A est définie comme suit :

Algorithme 8 : f_A [Ghazal]

Entrées : $T_{i,j}$ pour $i, j = 1, 2, r$ vecteur de longueur h tel que $r^{(2)} = 0$

$H', y', s' = H' y'^T, p, h_1, h_2$

Sorties : une liste L d'ensembles $I', |I'| = p$ tels que $\sum_{l \in I'} H'_l = s'$

1. $r_1 \leftarrow r$
2. $r_2 \leftarrow s' - r$
3. Initialiser et remplir les $D_{I_{i,j}}$ et $D_{f_{i,j}}, i, j = 1, 2$
4. **pour** $i = 1, 2$ **faire**
5. $D_{f_i}(r) \leftarrow D_{f_{i,1}} \bowtie_{r_i}^{h_1} D_{f_{i,2}}$
6. $D_{I_i}(r) \leftarrow D_{I_{i,1}} \bowtie_{r_i}^{h_1} D_{I_{i,2}}$
7. $D \leftarrow D_{f_1} \bowtie_{h_2, |I'|=p}^{s'} D_{f_2}(r)$
8. **si** on peut trouver e de poids t à partir de D **alors**
9. **retourner** 0
10. **sinon**
11. **retourner** 1

La proportion ϵ des éléments marqués

On définit $M_{f_A} := \{I' | w_H(e) = t \text{ où } e = \text{Dépoïnçonner}(I')\}$.

Sous l'hypothèse d'avoir choisi le bon vecteur r , la structure de données D contient des ensembles I' qui s'écrivent $I_1 \diamond I_2 = (I_{1,1} \sqcup I_{1,2}) \diamond (I_{2,1} \sqcup I_{2,2})$ avec $\diamond \in \{\sqcup, \Delta\}$ en fonction de l'algorithme. La probabilité qu'un seul des $T_{i,j}$ contienne l'ensemble $I_{i,j}$ en question est $\frac{T}{|S_{i,j}|}$.

Comme les $T_{i,j}$ sont indépendants, la probabilité de succès ϵ est donc $\left(\frac{T}{|S_{i,j}|}\right)^4$

Le squelette de la fonction de $Recherche_A$:

Algorithme 9 : MMTQW [Ghazal]

Entrées : $H, y, s = Hy^T, p, t$

Sorties : e tel que $s = mG + e$ et $w_H(e) = t$, $NULL$ si aucun tel vecteur n'a pu être trouvé

1. Préparer G', H', y' et calculer $s' = H'y'^T$
2. $|p\rangle \leftarrow Grover(R_A^{-1}, QW_A)$
3. $r \leftarrow Measure(|p\rangle)$
4. $I' = (I_{1,1}, I_{1,2}, I_{2,1}, I_{2,2}) \leftarrow QW(J(4 | S_{i,j} |, 4T), f_A(\dots, r, \dots))$
5. $e \leftarrow Dépoinçonner(I')$
6. **retourner** e

On rappelle $Grover(\epsilon, f)$ pour la recherche d'un x tel que $f(x) = 1$ avec l'algorithme de Grover, et on rappelle que l'on a défini la marche $QW(J(4 | S_{i,j} |, 4T), f_A)$ ci-dessus.

Puis, on définit la fonction $QW : r \mapsto \{0, 1\}$ comme la fonction qui renvoie 1 ssi $QW(J(4 | S_{i,j} |, 4T), f_A)$ trouve un bon 4-uplet lorsque f_A prend en argument r .

Complexité Sous l'hypothèse d'avoir le bon vecteur r , la marche aléatoire trouve le bon vecteur d'erreur e en temps [Ghazal]

$$\begin{aligned} S + \frac{1}{\sqrt{\epsilon}}(C + \frac{1}{\sqrt{\delta}}U) &= O(T) + |S_{i,j}|^2 T^{-2} (\tilde{O}(1) + O(\sqrt{T})O(1)) \\ &= \max(T, |S_{i,j}|^2 T^{-3/2}) \end{aligned}$$

Il faut ajouter à ce coût le coût de la recherche du bon vecteur r parmi R_A possibilités, ce qui est faisable en temps $O(\sqrt{R_A})$ avec l'algorithme de Grover. Ainsi le coût total sera $\tilde{O}(\sqrt{R_A} \max(T, |S_{i,j}|^2 T^{-3/2}))$

Algorithme MMTQW

Impossibilité d'une approche directe [Ghazal] une adaptation directe de MMT ne marche pas car le choix de h_1 et h_2 est imposé et pour avoir exactement une représentation de

chaque ensemble de positions d'erreur, il faut prendre $h_1 = p$. D'autre part, pour garder l'équilibre qui fait fonctionner l'algorithme, il faut prendre $h_1 = \log_2(T)$. Par conséquent, on devrait avoir $p = \log_2(T)$, or on devrait avoir $p < t$, mais dans les cas qui nous intéressent on a $\log_2(T) > t$, par conséquent la totalité des conditions souhaitées sur les paramètres ne peuvent pas être vérifiées.

Pour pallier à ce problème, nous allons opter pour **un algorithme hybride entre MMT et SS(Shamir-Schroeppe)** Cet algorithme, que l'on note MMT_{QW} , utilise la technique des représentations. Il s'agit en effet de prendre le cas de figure $h_1 > p$ sur ce, les conditions d'équilibre nécessaires sont bien remplies, et il y a une garantie d'égalité pour p composantes du vecteur r par la technique des représentations, et il faut chercher exhaustivement les $h_1 - p$ composantes restantes.

Ainsi, on a $|S_{i,j}| \approx \left(\frac{k+h}{2}\right)^{1/2}$. Nous allons donc considérer ici des sous-ensembles $T_{i,j}$ de $S_{i,j}$ de taille $T \leq \left(\frac{k+h}{2}\right)^{1/2}$ et donc une marche sur $J\left(4\left(\frac{k+h}{2}\right)^{1/2}, 4T\right)$.

Comme on cherche exhaustivement $h_1 - p$ composantes du vecteur r et que $2^{h_1} = T$, on a $R_{MMT_{QW}} = \frac{T}{2^p}$ et donc la marche aléatoire a la complexité suivante :

$$\tilde{O}\left(\sqrt{R_{MMT_{QW}}} \max\left(T, |S_{i,j}|^2 T^{-3/2}\right)\right) = \tilde{O}\left(\frac{1}{2^{p/2}} \max\left(T^{3/2}, \left(\frac{k+h}{2}\right) T^{-1}\right)\right).$$

Les deux termes de la complexité sont égaux si l'on prend $2^{h/2} = T = \left(\frac{k+h}{2}\right)^{2/5}$ auquel cas

on obtient une complexité temporelle $T_{MMT_{QW}}$ en $\tilde{O}\left(\frac{\left(\frac{k+h}{2}\right)^{3/5}}{2^{p/2}}\right)$

3.6.3 ISD avancé 2 avec technique de représentation étendue

3.6.4 Algorithme MMT* classique

Technique des représentations étendue ou $1 + 1 = 0$ [Ghazal]-p66

Dans l'algorithme MMT, on représente l'ensemble I' des positions d'erreurs à l'aide de deux sous ensembles disjoints de taille $\frac{p}{2}$ de $\{1, \dots, k+h\}$.

Autrement dit, les coordonnées non-nulles du vecteur d'erreur sont écrites comme $1 + 0$ ou $0 + 1$ et les coordonnées nulles comme $0 + 0$. Or, en binaire, 0 s'écrit également comme $1 + 1$. Il est possible d'augmenter le nombre de représentations en exploitant cette double façon d'écrire le 0 binaire.

Par conséquent, on considérera dans cette section des représentations de l'ensemble I' comme $I_1 \Delta I_2$ avec $|I_i| = \frac{p}{2} + \epsilon, i = 1, 2, \epsilon > 0, |I_1 \cap I_2| = \epsilon$. Cela permet d'avoir $\binom{p}{p/2} \binom{k+h-p}{\epsilon}$ représen-

tations. Le deuxième facteur est grand lorsque p est petit et par conséquent ϵ sont petits, ce qui arrive souvent car les vecteurs d'erreur sont souvent creux.

De plus, contrairement à MMT où à la fin il fallait filtrer les sous-ensembles I' de taille p parmi les I' , $|I'| \leq p$, ici, en choisissant à bon escient la valeur de ϵ , par construction on n'aura à la fin que très peu d'ensembles I' de taille différente de p et ce sans avoir besoin de filtrage supplémentaire.

Description de l'algorithme [Ghazal]

En effectuant les modifications décrites ci-dessous, on se retrouve avec un algorithme semblable à SS et à MMT avec quelques différences dans les définitions des ensembles et listes

$$\begin{cases} J_{1,1}=J_{2,1}=\left\{1,\dots,\frac{k+h}{2}\right\} \\ J_{1,2}=J_{2,2}=\left\{\frac{k+h}{2}+1,\dots,k+h\right\} \end{cases}$$

Et $S_{i,j} = \{I_{i,j} \subset J_{i,j} \mid |I_{i,j}| = p/4 + \epsilon/2\}$, $|S_{i,j}| = \binom{\frac{k+h}{2}}{p/4+\epsilon/2} \approx \binom{k+h}{p/2+\epsilon}^{1/2}$

Comme dans MMT, on remarque que I' ne s'écrit pas tout à fait comme $I_1 \Delta I_2$ avec $I_i \subset \{1, \dots, k+h\}$, $|I_i| = p/2 + \epsilon$, mais un calcul semblable à celui fait dans MMT montre que cette restriction ne fait perdre qu'un facteur polynomial.

Preuve de correction de $\text{DecodeCollision}_{\text{MMT}^*}$

- La liste L_1 contient les $I_1 = I_{1,1} \sqcup I_{1,2}$ tels que $r^{(1)} = r_1^{(1)} = \sum_{i \in I_{1,1}} H_i'^{(1)} + \sum_{i \in I_{1,2}} H_i'^{(1)} = \sum_{i \in I_1} H_i'^{(1)}$
- La liste L_2 contient les $I_2 = I_{2,1} \sqcup I_{2,2}$ tels que $s'^{(1)} - r^{(1)} = r_2^{(1)} = \sum_{i \in I_{2,1}} H_i'^{(1)} + \sum_{i \in I_{2,2}} H_i'^{(1)} = \sum_{i \in I_2} H_i'^{(1)}$
- On a $|I_i| = p/2 + \epsilon$ pour $i = 1, 2$ par construction car $I_{i,2} \cap I_{i,2} = \emptyset$
- La liste L contient donc les $I' = I_1 \Delta I_2$ tels que :

$$\begin{cases} I_1 \in L_1, i.e. \sum_{i \in I_1} H_i'^{(1)} = r^{(1)} \\ I_2 \in L_2, i.e. \sum_{i \in I_2} H_i'^{(1)} = s'^{(1)} - r^{(1)} \\ \sum_{i \in I_1} H_i'^{(2)} + \sum_{i \in I_2} H_i'^{(2)} = s'^{(2)} \end{cases}$$

$$\Rightarrow \begin{cases} \sum_{i \in I_1} H_i'^{(1)} + \sum_{i \in I_2} H_i'^{(2)} = s'^{(1)} \\ \sum_{i \in I_1} H_i'^{(2)} + \sum_{i \in I_2} H_i'^{(2)} = s'^{(2)} \end{cases}$$

$$\Rightarrow \sum_{i \in I'} H_i' = s'$$

En choisissant bien la valeur de ϵ , $I' = I_1 \Delta I_2$ vérifiera en moyenne $|I'| = p$

Par conséquent la liste finale L est bien comme on souhaite.

Représentations Un argument analogue à celui utilisé pour MMT montre que, pour qu'il reste à la fin en moyenne un nombre constant de représentations de chaque ensemble de positions d'erreurs, il faut prendre

$$h_1 = \log_2 \left(\left(\binom{p/2}{p/4} \binom{k+h-p}{\epsilon/2} \right)^2 \right) \approx p + (k+h-p)H\left(\frac{\epsilon}{k+h-p}\right)$$

avec $\epsilon \approx \frac{\pi^2}{4R}$

le but de MMT* était d'appliquer la technique des représentations étendues ($1+1=0$) à un seul niveau en choisissant ϵ de sorte à avoir le poids du vecteur final égal à p en moyenne et on avait vu que cela n'améliorait pas vraiment le temps d'exécution asymptotique.

De ce fait, on va enlever la contrainte sur ϵ en essayant d'avoir $\epsilon \approx 0$ avec l'algorithme $MMT^\#$.

3.6.5 Algorithme de $MMT^\#$

Il s'agit ici comme pour MMT d'utiliser la version "hybride" avec $h_1 > p$.

Version classique

Théorème 23 [Ghazal] Si H est une matrice aléatoire, en notant $R := \frac{k}{n}$, $\tau := \frac{t}{n}$, $\pi := \frac{p}{n}$, $\eta := \frac{h}{n}$ et $\epsilon = \frac{\epsilon}{n}$, on a l'expression suivant l'exposant $\alpha_{MMT^\#}$:

$\alpha_{MMT^\#}(R) = \min_{\pi, \eta, \epsilon} (\beta(R, \pi, \eta) + \max_{i=1,2,3} \gamma_i(R, \eta, \pi, \epsilon))$ avec :

$$\beta(R, \pi, \eta) = H(\tau) - (1 - R - \eta)H\left(\frac{\tau - \pi}{1 - R - \eta}\right) - (R + \eta)H\left(\frac{\pi}{R + \eta}\right)$$

$$\gamma_1(R, \eta, \pi, \epsilon) = \frac{R + \eta}{2} H\left(\frac{\pi/2 + \epsilon}{R + \eta}\right)$$

$$\gamma_2(R, \eta, \pi, \epsilon) = (R + \eta)H\left(\frac{\pi/2 + \epsilon}{R + \eta}\right) - \pi - (1 - R - \eta)H\left(\frac{\epsilon}{1 - R - \eta}\right)$$

$$\gamma_3(R, \eta, \pi, \epsilon) = 2(R + \eta)H\left(\frac{\pi/2 + \epsilon}{R + \eta}\right) - (1 - R - \eta)H\left(\frac{\epsilon}{1 - R - \eta}\right) - \pi - \eta$$

Soumise aux contraintes :

$$0 \leq \epsilon \leq \pi$$

$$0 \leq \pi \leq \min(\tau, \eta) \leq R + \eta$$

$$0 \leq \eta \leq 1 - R - \tau + \pi \leq 1 - R$$

Et $\max_R (\alpha_{MMT^\#}) = 0.1052837$ pour $R = 0.4271$, valeur obtenue pour $\eta \approx 0.131441089$, $\pi \approx 0.036218826$ et $\epsilon \approx 0.003388306$.

Démonstration : Il s'agit des mêmes calculs que pour $MMT^\#$, à cette différence près qu'il n'y a plus de contrainte de contrainte sur $\epsilon \approx 0.003388306$.

Version quantique avec Quantum Walk

Théorème 24 [Ghazal] Si H est une matrice aléatoire, en notant $R := \frac{k}{n}$, $\tau := \frac{t}{n}$, $\pi := \frac{p}{n}$, $\eta := \frac{h}{n}$ et $\epsilon = \frac{\epsilon}{n}$, on a l'expression suivant l'exposant $\alpha_{MMTQW\#}$:

$$\alpha_{MMTQW\#}(R) = \min_{\pi, \eta, \epsilon} \left(\frac{H(\tau) - (1-R-\eta)H(\frac{\tau-\pi}{1-R-\eta}) - (R+\eta)H(\frac{\pi}{R+\eta}) + \mu(R, \eta, \pi, \epsilon)}{2} \right)$$

Avec :

$$\mu(R, \eta, \pi, \epsilon) = \frac{3(R+\eta)}{5} H\left(\frac{\pi/2 + \epsilon}{R+\eta}\right) - \pi - (1-R-\eta)H\left(\frac{\epsilon}{1-R-\eta}\right)$$

Soumise aux contraintes :

$$\begin{aligned} 0 &\leq \epsilon \leq \pi \\ \pi &= 2 \left((R+\eta)H^{-1}\left(\frac{5\eta}{4(R+\eta)}\right) - \epsilon \right) \\ 0 &\leq \pi \leq \min(\tau, R+\eta) \\ 0 &\leq \eta \leq 1-R-\tau+\pi \leq 1-R \end{aligned}$$

Et $\max_R (\alpha_{MMTQW\#}) = 0.0586953$ pour $R = 0.4514$, valeur obtenue pour $\eta \approx 0.0375776$, $\pi \approx 0.0107$ et $\epsilon \approx 0.0006935$.

Démonstration : Il s'agit des mêmes calculs que pour $MMT^\#$, à cette différence près qu'il n'y a plus de contrainte sur ϵ .

L'algorithme de MMT choisit systématiquement $\epsilon = 0$, ce qui n'est pas optimal. Avec un ϵ plus grand, on peut diminuer la probabilité de créer des éléments mal formés. Par ailleurs, choisir ϵ plus grand permet également d'augmenter la taille de l'espace de recherche. Ces deux considérations vont dans le même sens : il est beaucoup plus facile de trouver de nombreuses solutions avec $\epsilon > 0$. Cela dit il y'aura alors à nouveau des éléments mal formés. Donc la difficulté consiste à trouver le bon équilibre. Sur ce, nous attaquons la technique de recherches de voisins proches.

3.6.6 Les recherches de voisins proches : Algorithme de May-Ozerov

L'algorithme de May et Ozerov adopte une autre approche : étant donnés deux listes L_1 et L_2 , que l'on a généré directement ou par collisions/représentations, trouver directement la solution du problème principal. En termes plus formels, il s'agit de résoudre le problème du voisin le plus proche mais avant d'énoncer le problème on va définir les 6 lemmes [Ghazal] et les définitions sur lesquels la recherche du voisin le plus se repose :

Définition 28 Soit $(u^*, v^*) \in L_1 \times L_2$ la solution cible. On dit qu'un chemin de la racine jusqu'à une feuille de l'arbre de calcul de *NearestNeighbour* est bon si toutes les paires de sous-listes à chaque noeud contiennent (u^*, v^*)

Lemme 2 Soit (L_1, L_2, γ) une instance d'un problème *NearestNeighbour* de paramètres (m, γ, λ) dont une solution inconnue est $(u^*, v^*) \in L_1 \times L_2$. On écrit $u^* = (u_1^*, \dots, u_t^*), v^* = (v_1^*, \dots, v_t^*)$ avec chaque $v_j^*, 1 \leq j \leq t$ sur une bande de longueur $\alpha_j m, \sum_{j=1}^t \alpha_j = 1$. Alors, il est vrai avec une probabilité inversement polynomiale que dans au moins une des listes après la permutation uniforme et l'ajout d'un vecteur r uniformément aléatoire sur chaque bande :

1. $w_H(u_j^* + v_j^*) = \gamma \alpha_j m$
2. $w_H(u_j^*) = w_H(v_j^*) = \frac{\gamma \alpha_j m}{2}, \forall 1 \leq j \leq t$.

Le lemme suivant porte sur la probabilité importante que les listes construites et gardées à chaque étape par l'algorithme *NearestNeighbourRec* contiennent la solution cible. On y prouve notamment la nécessité de tester $\tilde{O}(2^{y \alpha_j m})$ à chaque étape de la récursion.

Lemme 3 Soit $t \in \mathbb{N}$ la hauteur de l'arbre de calcul de *NearestNeighbour*. Alors cet arbre de calcul comporte un bon chemin avec une probabilité pas trop faible (inversement polynomiale).

Le lemme suivant prouve que le choix d'imposer une limite de taille sur les listes gardées est pertinente.

Lemme 4 Soit $t \in \mathbb{N}$ la hauteur de l'arbre de calcul de *NearestNeighbour* et $\epsilon > 0$. Alors, parmi les $2t$ listes aux noeuds d'un bon chemin, chacune au niveau j est de taille $\tilde{O}\left(2^{\lambda m(1 - \sum_{i=1}^j \alpha_i) + \frac{\epsilon}{2}}\right)$ avec une probabilité inversement polynomiale.

On aura besoin du lemme suivant dans la preuve du théorème final.

Lemme 5 Soient $0 < \gamma < \frac{1}{2}$ et $0 < \lambda < 1 - H(\frac{\gamma}{2})$. Soit $y = (1 - \gamma)\left(1 - H(\frac{h - \frac{\gamma}{2}}{1 - \gamma})\right)$ alors $y > \gamma$

Lemme 6 Soit $\alpha_1, \dots, \alpha_t$ une suite de nombres telle que $\alpha_{j+1} = r \alpha_j$ et $\sum_{i=1}^t \alpha_i = s$. Alors $\alpha_1 = \frac{s(1-r)}{1-r^t}$ et $\alpha_t = \frac{s(1-\frac{1}{r})}{1-(\frac{1}{r})^t}$

Lemme 7

1. Si $\lambda + y \alpha_1 + \frac{\epsilon}{2} = y + \epsilon$, alors $\alpha_1 = \frac{y - \lambda + \frac{\epsilon}{2}}{y}$
2. Si $\lambda \alpha_t + y + \frac{\epsilon}{2} = y + \epsilon$, alors $\alpha_t = \frac{\epsilon}{2\lambda}$

Problème 1 *Nearest Neighbour de paramètres (m, γ, λ) . Soit $m \in \mathbb{N}, 0 < \gamma < \frac{1}{2}$ et $0 < \lambda < 1$. Etant donné deux listes L_1, L_2 de taille égale $|L_1| = |L_2| = 2^{\lambda m}$ de vecteurs uniformément choisis et indépendants deux-à-deux de \mathbb{F}_2^m , s'il existe $(u^*, v^*) \in L_1 \times L_2$ tel que $w_H(u^* + v^*) = \gamma m$, donner une liste C contenant (u^*, v^*) .*

Notons qu'un algorithme naïf pour trouver (u^, v^*) consisterait à tester, pour tous les couples $(u, v) \in L_1 \times L_2$ s'ils résolvent du problème, i.e. si $w_H(u + v) = \gamma m$. Autrement dit, on est à la recherche d'un élément parmi $2^{2\lambda m}$. Ainsi, l'algorithme naïf de résolution du problème coûte $O(2^{2\lambda m})$ dans le cas classique et $O(2^{\lambda m})$ dans le cas quantique avec l'algorithme de Grover.*

3.6.7 Version classique

Algorithme de résolution du problème Nearest Neighbour

On va donner deux algorithmes : le premier, *NearestNeighbour*, définit quelques paramètres et initialise un environnement et fait un premier appel au second algorithme, *NearestNeighbourRec*, qui, comme son nom l'indique, est un algorithme récursif et fait le travail principal de construction de la liste C , en se basant sur le constat suivant : si on divise longueur m du vecteur en t bandes de longueur $\alpha_1 m, \dots, \alpha_t m$, alors si on a $w_H(u^* + v^*) = \gamma m$, on a aussi, éventuellement après avoir appliqué une même permutation aléatoire aux coordonnées des vecteurs, que $w_H(u_i^* + v_i^*) = \gamma \alpha_i m$ si w_i désigne la restriction du vecteur w à la bande de longueur $\gamma \alpha_i m$. Autrement dit, la propriété globale sur le poids vaut aussi localement sur chaque bande, à quelques nuances près.

Algorithme 9 : NearestNeighbour [Ghazal]-p85**Entrées :** $L_1, L_2, m, \gamma, h, \lambda, \epsilon$ **Sorties :** Liste C contenant la solution $(u^* + v^*) \in L_1 \times L_2$

1. $y \leftarrow (1 - \gamma) \left(1 - H \left(\frac{H^{-1}(1-\lambda) - \frac{\gamma}{2}}{1-\gamma} \right) \right)$
2. $t \leftarrow \left\lceil \frac{\log_2(y - \lambda + \frac{\epsilon}{2}) - \log_2(\frac{\epsilon}{2})}{\log_2(y) - \log_2(\lambda)} \right\rceil$
3. $\alpha_1 \leftarrow \frac{1 - \frac{\lambda}{y}}{1 - (\frac{\lambda}{y})^t}$
4. **pour** $j \leftarrow 2$ **à** t **faire**
5. $\alpha_j \leftarrow \frac{y}{\lambda} \alpha_{j-1}$
6. $poly(m)$ permutations uniformément aléatoires π de $\{1, \dots, m\}$ **faire**
7. **pour** $poly(m)$ vecteurs uniformément aléatoires r ayant poids $\frac{\alpha_j m}{2}$ sur chaque bande de longueur $\alpha_j m$ **faire**
8. $\bar{L}_i \leftarrow \pi(L_i) + r$ pour $i = 1, 2$
9. Enlever des \bar{L}_i les vecteurs de poids différent de $\frac{\alpha_j m}{2}$ sur chaque bande
10. **retourner** $NearestNeighbourRec(\bar{L}_1, \bar{L}_2, m, t, \gamma, \lambda, (\alpha_j)_{j=1, \dots, t}, y, h, \epsilon, 1)$

Algorithme 10 : NearestNeighbourRec [Ghazal]-p85**Entrées :** $L_1^{j-1}, L_2^{j-1}, m, t, \gamma, \lambda, (\alpha_j)_{j=1, \dots, t}, y, h, \epsilon, 1$ **Sortie :** Liste C contenant la solution $(u^*, v^*) \in L_1 \times L_2$

1. $C \leftarrow \emptyset$
2. **si** $j = t + 1$ **alors** /* cas de base
3. $C \leftarrow \left\{ (u, v) \in L_1^{(j)} \times L_2^{(j)} \mid w_H(u + v) = \gamma m \right\}$
4. **sinon**
5. **pour** $O(2^{\tilde{y} \alpha_j m})$ tours de boucle **faire** /* cas général
6. $A_j = A_{j,1} \sqcup A_{j,2} \leftarrow$ partition de la bande de longueur $\alpha_j m$ en deux parties égales
7. **pour** $i = 1, 2$ **faire**
8. $L_i^{(j)} \leftarrow \left\{ v_{L_i^{(j-1)}} \in L_i^{(j-1)} \mid \text{de poids } \frac{\alpha_j h m}{2} \text{ sur } A_{j,1} \right\}$
9. **si** $|L_i^{(j)}| \lesssim \tilde{O} \left(2^{\lambda m (1 - \sum_{i=1}^j \alpha_i + \frac{\epsilon}{2})} \right)$ **alors**
10. $C \leftarrow C \cup NearestNeighbourRect(L_1^{(j)}, L_2^{(j)}, m, t, \gamma, \lambda, (\alpha_j)_{j=1, \dots, t}, y, h, \epsilon, j + 1)$

Le principe de cette méthode repose sur le théorème suivant :

Théorème 25 [Ghazal]-p95 Pour tout $\epsilon > 0$ et $\lambda < 1 - H(\frac{\gamma}{2})$, l'algorithme *NearestNeighbour*

résout le problème Nearest Neighbour de paramètres (m, γ, λ) avec une probabilité inversement polynomiale en temps $\tilde{O}(2^{(y+\epsilon)m})$ où $y = (1 - \gamma) \left(1 - H\left(\frac{h-\gamma}{1-\gamma}\right)\right)$

Démonstration :

D'après le lemme 2, l'arbre de calcul de l'algorithme comporte un bon chemin avec une probabilité inversement polynomiale. On considère ce chemin. Alors, d'après le lemme 3, la taille de chaque liste d'entrée au niveau $j, 1 \leq j \leq t$ est $\tilde{O}\left(2^{m(\lambda(1-\sum_{i=1}^{j-1} \alpha_i) + \frac{\epsilon}{2})}\right)$. On construit $\tilde{O}(2^{y\alpha_j m})$ nouvelles sous-listes à ce niveau, ce qui, par récurrence, donne un total de $\tilde{O}\left(2^{y\sum_{i=1}^j \alpha_i m}\right)$ sous-listes à ce niveau, construites à l'aide d'une procédure de recherche dans les listes d'entrées. La complexité de calcul au niveau j est donc :

$$\begin{aligned} &\tilde{O}\left(2^{m(\lambda(1-\sum_{i=1}^{j-1} \alpha_i) + \frac{\epsilon}{2})} 2^{y\sum_{i=1}^j \alpha_i m}\right) \\ &\tilde{O}\left(2^{m(\lambda(1-\sum_{i=1}^{j-1} \alpha_i) + \frac{\epsilon}{2} + y\sum_{i=1}^j \alpha_i)}\right) \end{aligned}$$

Au dernier niveau ($j = t + 1$), il y a $\tilde{O}(2^{ym})$ listes d'entrée de taille $\tilde{O}\left(2^{m(\lambda(1-\sum_{i=1}^t \alpha_i) + \frac{\epsilon}{2})}\right) = \tilde{O}(2^{\frac{\epsilon}{2}})$ on construit la liste des solutions de manière naïve (i.e. complexité quadratique en la taille des listes) à partir de ces listes, donc la complexité totale à ce niveau est $\tilde{O}\left(2^{m(y+\epsilon)}\right)$. La complexité totale sera donc :

$$\max_{j=1, \dots, t} \left(2^{m(\lambda(1-\sum_{i=1}^{j-1} \alpha_i) + y\sum_{i=1}^j \alpha_i + \frac{\epsilon}{2})}, 2^{m(y+\epsilon)}\right)$$

D'après le lemme 5, la complexité de tous les étapes dépendant de j seront égales pour le choix de

$$(*) \alpha_1 = \frac{1-\frac{\lambda}{y}}{1-(\frac{\lambda}{y})^t} \text{ avec une valeur de } (**) \tilde{O}\left(2^{m(\lambda+y\alpha_1) + \frac{\epsilon}{2}}\right)$$

Ainsi le lemme 6 permet l'égalité de complexité des j étapes et le dernier étape de l'algorithme en résolvant l'équation suivante :

$$(*) = (**) \Rightarrow t = \frac{\log_2(y-\lambda+\frac{\epsilon}{2}) - \log_2(\frac{\epsilon}{2})}{\log_2(y) - \log_2(\lambda)}$$

Cette technique possède aussi une version quantique avec l'utilisation de l'algorithme de Grover pour effectuer les opérations de recherche mais celui demeure inefficace à cause de la taille des listes construites. Cependant en utilisant le quantum Walk sur un graphe de Johnson.

On a dit que l'on pouvait considérer l'algorithme de May et Ozerov sous forme d'un arbre de calcul à $t + 1$ niveaux numérotés de 0 à t , où à chaque noeud il y a deux listes $L_j^{(1)}$

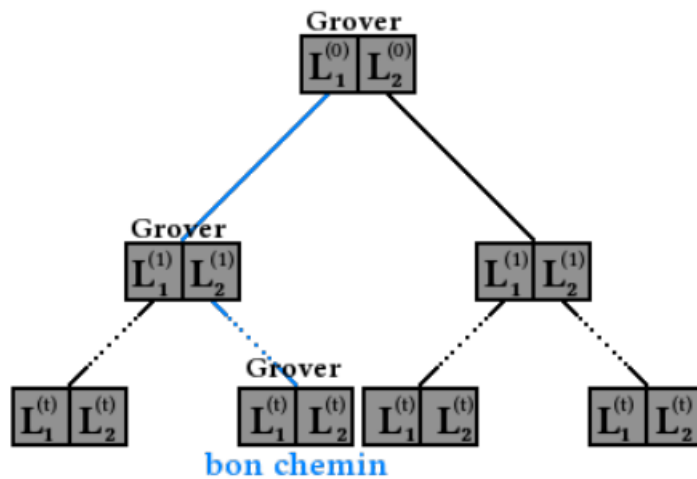


FIGURE 3.2 – illustration sous forme d'arbre

et $L_j^{(2)}$ et chaque noeud de niveau j a en moyenne $2^{y\alpha_j m}$ fils. Le but était de trouver un bon chemin, vérifiant certaines propriétés, de la racine à une feuille. Ce qu'on a fait avec l'algorithme de Grover consistait à améliorer le temps de recherche du bons fils à chaque niveau de l'arbre. Illustration

Chapitre 4

Etudes comparatives

Contents

4.1	Classic McEliece	128
4.1.1	Avantages	128
4.1.2	Les inconvénients	128
4.1.3	Cryptanalyse du système	129
4.2	BIKE	129
4.2.1	Les avantages	129
4.2.2	Les inconvénients	130
4.2.3	Cryptanalyse du système	131
4.3	Hamming quasi-cyclic	132
4.3.1	Les avantages	133
4.3.2	Les inconvénients	133

Les algorithmes maintenus pour 4 ème tour de NIST ont apporté des améliorations par rapport à leur ancienne version si elle existe soit au niveau de la sécurité , soit dans leur implémentation. Dans ce chapitre on verra une analogie des avantages et des inconvénients des candidats.

4.1 Classic McEliece

Pou rappel le système de McEliece est muni des paramètres (n, k, t) pour un (n, k) -code linéaire \mathbf{C} capable de corriger t partagés par les différentes parties, d'une paire de clef publique (G', t) et privée (S, G, P) tel que $G' = SGP$ avec S une matrice binaire aléatoire de taille $k \times n$ non singulier, G , une matrice génératrice de \mathbf{C} de taille $k \times n$ et P , une matrice de permutation aléatoire de taille $n \times n$.

Les clés publiques et privée sont de grandes matrices ce qui constitue un des plus grands désavantages. De plus la grandeur du texte chiffré est deux fois celle du texte d'origine.[McEliece2]

4.1.1 Avantages

Le principal avantage de cette méthode est sa facilité d'implémentation : les seules opérations sont des opérations bit à bit. Par contre, l'implémentation nécessite beaucoup de place mémoire.

Ce cryptosystème, reposant sur un problème difficile de la théorie des codes, n'a pas rencontré de véritable soutien dans la communauté cryptographique. L'une des principales raisons de cet état de fait est la taille de la clé. Pourtant, le cryptosystème de McEliece possède des propriétés intéressantes, citons notamment

- la sécurité croît beaucoup plus avec la taille des clés que pour le système RSA ;
- la rapidité du chiffrement. Un autre avantage est de reposer sur un problème très différent des algorithmes asymétriques usuels.

En conséquence de quoi une percée théorique dans le domaine de la factorisation, qui ruinerait RSA, n'affecterait en rien ce cryptosystème. Le cryptosystème de McEliece résiste à ce jour à toute tentative de cryptanalyse, mais est rarement utilisé en pratique du fait de la grande taille des clés .

4.1.2 Les inconvénients

1. La taille de la clef publique posera certainement des problèmes d'exécution.
2. Le message chiffré est plus long que le message clair. Cette augmentation de la largeur du message chiffré rend le système plus sensible aux erreurs de transmission.

3. Le crypto système n'est employé pour la signature ou l'authentification parce que l'algorithme de chiffage n'est pas linéaire et tout l'algorithme est vraiment asymétrique

4.1.3 Cryptanalyse du système

La sécurité repose sur l'impossibilité de déduire un algorithme de décodage efficace de la seule clé publique.

Si tel est le cas, le décryptage d'une instance du système se ramènera à la résolution d'une instance d'un problème de décodage dans un code linéaire. Il en découle que toute attaque du système devra appartenir à l'une des catégories suivantes :

- les attaques par décodage : il s'agit de décoder une instance du cryptosystème à l'aide d'un algorithme général (i.e. le code public est considéré comme un code linéaire aléatoire : Pour être plus précis, nous disons que le Goppa distinguishing problem, qui demande de discriminer si un ensemble de vecteurs est la base d'un code de Goppa ou d'un code aléatoire, est intraitable du point de vue informatique).
- les attaques structurelles : il s'agit, à l'aide de la clé publique, de retrouver tout ou partie de la structure du code secret.

4.2 BIKE

BIKE est une suite de mécanismes d'encapsulation de clé sécurisée (KEM) IND-CPA composée de BIKE-1, BIKE-2 et BIKE-3. Chaque variante a ses propres avantages et inconvénients.

[Rap]

4.2.1 Les avantages

1. toutes les variantes de BIKE sont basées sur une densité modérée quasi-cyclique contrôle de parité (codes QC-MDPC), qui peuvent être décodés efficacement grâce à des techniques de décodage par retournement de bits. Ce type de décodeur est extrêmement simple : il estime quelles sont les positions les plus susceptibles d'être erronées, retournez-les et observez si le résultat est meilleur (poids du syndrome plus petit) qu'avant ou non. Ce processus converge très rapidement ; en particulier, on a présenté un décodeur à retournement de bits à 1 itération.
2. Une autre caractéristique commune à toutes les variantes de BIKE est le fait qu'elles s'appuient sur des clés éphémères. Cela conduit à deux choses : dans un premier temps, cela va à l'encontre du GJS. attaque de réaction mentionnée dans la section 5,

qui est une attaque qui doit être observée un grand nombre de décodages pour une même clé privée (chose impossible quand des clés éphémères sont utilisées). L'autre aspect de ce choix est que la génération clé doit être efficace puisqu'il est exécuté à chaque encapsulation de clé.

3. Concernant la bande passante de communication, dans BIKE-1 et BIKE-3 toutes les clés publiques, les clés privées et les cryptogrammes ont une longueur de n bits, correspondant à la bande passante des messages échangés par les parties. BIKE-2 propose des clés publiques plus petites et textes chiffrés, r bits uniquement, correspondant à la bande passante des messages échangés également par les parties. Deux messages sont échangés par encapsulation de clé de celui-ci taille (soit n , soit r bits). En pratique, ces chiffres vont de 1,24 Ko par message en BIKE-2 niveau de sécurité 1, jusqu'à 8,82 Ko par message en BIKE-3 niveau de sécurité 5. Ces chiffres semblent assez raisonnables par rapport à la taille moyenne d'un page sur Internet (actuellement près de 2 Mo [2]), à titre d'exemple

4.2.2 Les inconvénients

1. Un point d'attention dans BIKE est le fait que, de nos jours, le bit flipping les techniques de décodage n'atteignent pas un taux d'échec de décodage négligeable. Cela fait défi pour atteindre des notions de sécurité plus élevées telles que IND-CCA. Cela peut également limiter l'utilisation de BIKE dans certaines applications comme par exemple Hybrid Encryption, où KEM et DEM doivent tous deux satisfaire à la sécurité IND-CCA pour garantir la sécurité du texte chiffré choisi pour le schéma de chiffrement hybride. Nous soulignons cependant qu'il semble possible (mais pas simple) de prouver que certaines techniques de décodage peuvent en fait, ils atteignent des taux d'échec de décodage négligeables pour les codes QC-MDPC.
2. Concernant la propriété intellectuelle, au meilleur de nos connaissances, BIKE-1 et BIKE-2 n'est couvert par aucun brevet. BIKE-3 est couvert par un brevet dont les propriétaires sont disposés à accorder une licence non exclusive aux fins de mise en œuvre la norme sans compensation et selon des termes et conditions raisonnables qui sont manifestement exempts de toute discrimination injuste, comme indiqué dans les documents ci-joints. déclarations signées. Nous soulignons que BIKE-1 et BIKE-2 ne sont pas couverts par le brevet susmentionné, et que l'équipe BIKE est prête à abandonner BIKE-3 si cela devient toujours un inconvénient lorsque l'on compare notre suite avec d'autres propositions.

4.2.3 Cryptanalyse du système

1. Concernant la sécurité, toutes les variantes de BIKE s'appuient sur des problèmes de théorie du codage : problèmes de décodage du syndrome quasi-cyclique et de recherche de mots de code quasi-cycliques. Les meilleures stratégies pour résoudre ces problèmes reposent sur les techniques de décodage des ensembles d'informations (ISD), un domaine de recherche qui existe depuis très longtemps. historique (l'ouvrage fondateur de Prange remonte à 1962) et qui semble très peu amélioré au fil des années. De plus, nous montrons que dans le cadre quantique, L'algorithme de Grover utilisé en plus de l'algorithme séminal de Prange ISD est toujours le choix le plus préférable dans notre cas.
2. "BIKE cible la sécurité de l'IND-CPA et ne tente pas d'y parvenir difficile pour un attaquant de lancer une attaque par texte chiffré choisi si les clés sont réutilisées. Cette décision de conception a été prise par les auteurs, sur la base des difficultés de concevoir un décodeur bit-flipping avec un signal suffisamment faible taux d'échec de décodage pour permettre une sécurité IND-CCA2 efficace construction."
3. La meilleure attaque ISD est Prange qui le fait en 50 ans

Dans la sécurité pratique les meilleures attaques connues sont [Rap] :

-key distinguishing attack : qui trouve un mot chiffré de C^\perp de poids w dans ce cas l'adversaire applique un isd sur le syndrome et la matrice de contrôle de parité pour aboutir à la résolution d'un problème avec r solutions telles r soit le nombre de lignes creuses de la matrice de contrôle de parité. Pour $N_s = r$ et $N_i = 1$, le coût de l'attaque diminue d'un facteur :

$$WF_{dist}(n, r, w) = WF_{isd}(n, n - r, w)/r$$

Dans le cas quasi-cyclique, il n'y a pas d'accélération évidente et on distingue l'attaque a le même coût que ci-dessus.

-key recovery attack : qui trouve r mots chiffrés de C^\perp de poids w

Dans ce cas on veut estimer le coût $WF_{reco}(n, r, w)$ de l'attaque en produisant r mots cachés de poids w dans C^\perp . On applique aléatoirement r attaques ISD pour trouver un mot . Sur chaque attaque a en moyenne $\frac{WF_{isd}(n, n-r, w)}{r}$ puisqu'il y'a r mots codés de poids w . Donc le coût de l'ensemble est :

$$WF_{reco}(n, n - r, w) = r \cdot \frac{WF_{isd}(n, n-r, w)}{r} = \frac{WF_{isd}(n, n-r, w)}{r}$$

Dans l'utilisation des codes quasi-cycliques tous les mots de poids faibles peuvent être retrouvés et l'attaque de key recovery attack n'est plus coûteuse que le key distinguishing attaque.

$$WF_{reco}^{QC}(n, r, w) = WF_{dist}^{QC}(n, r, w) = \frac{WF_{isd}(n, n-r, w)}{r}.$$

	MDPC	QC-MDPC
Key distinguishing	$\frac{1}{r} WF_{isd}(n, n-r, w)$	$\frac{1}{r} WF_{isd}(n, n-r, w)$
Key recovery	$WF_{isd}(n, n-r, w)$	$\frac{1}{r} WF_{isd}(n, n-r, w)$
Decoding	$WF_{isd}(n, r, t)$	$\frac{1}{\sqrt{r}} WF_{isd}(n, r, t)$

TABLE 4.1 – Les meilleures attaques contre MDPC (or QC-MDPC) codes

-Decoding attack : qui décode t erreurs d'un $(n, n-r) - \text{code linéaire}$.

Soit une estimation du coût de l'attaque $WF_{dect}(n, r, t)$ pour décoder t erreurs. Dans le cas des codes MDPC on a :

$$WF_{dec}(n, r, t) = WF_{isd}(n, r, t) .$$

Dans le cas quasi-cyclique, tout déplacement cyclique du syndrome cible $s \in \mathbb{F}_2^r$ fournit une nouvelle instance dont la solution est égale à l'originale, à une précision près décalage cyclique par bloc. Ainsi le nombre d'instances et de solutions est $N_i = N_s = r$. Donc un facteur \sqrt{r} est gagné :

$$WF_{reco}^{QC}(n, r, t) \geq \frac{WF_{isd}(n, r, t)}{\sqrt{r}}$$

La dernière est le cas de Decoding One Out Of Many (DOOM), l'attaquant analyse le gain (le gain réel pourrait être légèrement inférieur, puisque ces algorithmes dépendent de paramètres optimaux qui peuvent ne pas être les mêmes pour plusieurs instances) lorsque le problème de décodage est résolu plusieurs solutions et l'adversaire se contente d'en trouver une seule. En bref, lorsque le problème a N_s solutions, la probabilité de succès P augmente d'un facteur N_s (tant que $N_s P \ll 1$) et quand N_i les instances sont traitées simultanément, la taille de la liste L augmente au maximum d'un facteur $\sqrt{N_i}$ donc :

La technique DOOM apporte un gain de $N_s/\sqrt{N_i}$

Pour toutes les attaques, nous devons résoudre le problème du décodage par syndrome. La meilleure technique est le décodage d'ensembles d'informations (ISD). Les facteurs de travail ISD sont couramment utilisés pour l'évaluation de la sécurité des systèmes basés sur le code.

4.3 Hamming quasi-cyclic

Hamming Quasi-Cyclic (HQC) est un algorithme cryptographique post-quantique qui présente ses propres avantages et inconvénients. Explorons-les en fonction des informations disponibles.

4.3.1 Les avantages

1. Sécurité contre les attaques quantiques : HQC est conçu pour résister aux attaques des ordinateurs quantiques. À mesure que les ordinateurs quantiques deviennent plus puissants, les algorithmes cryptographiques traditionnels peuvent devenir vulnérables, faisant du HQC une solution importante pour sécuriser les communications à l'ère post-quantique.
2. Implémentation efficace : HQC offre une implémentation relativement efficace par rapport aux autres algorithmes post-quantiques. Il offre un bon équilibre entre sécurité et performances, ce qui le rend adapté à divers cas d'utilisation.
3. Aucune structure cachée : HQC ne s'appuie pas sur des structures cachées dans son processus de cryptage, ce qui ajoute une couche de sécurité supplémentaire. Cette résilience face à la cryptanalyse fait du HQC une option prometteuse pour des communications sécurisées.

4.3.2 Les inconvénients

1. Taille de clé de chiffrement plus petite : comparé à certains autres algorithmes cryptographiques post-quantiques, HQC a une taille de clé de chiffrement plus petite. Cela peut limiter son caractère pratique dans certains scénarios où des clés plus grandes peuvent être souhaitées pour une sécurité renforcée.
2. Adoption et recherche limitées : HQC est un algorithme relativement nouveau dans le domaine de la cryptographie post-quantique, ce qui signifie qu'il n'a pas encore été largement adopté et n'a pas fait l'objet de recherches approfondies. Cela peut entraîner une certaine incertitude quant à la sécurité et à la maturité à long terme de l'algorithme.

Il convient de noter que le domaine de la cryptographie post-quantique est en constante évolution et que de nouvelles recherches et développements pourraient affecter les avantages et les inconvénients du HQC à l'avenir.

Nous avons vu que les algorithmes ISD classiques et quantiques sont jusque là les meilleures attaques sur nos candidats malgré que la complexité soit toujours exponentielle. Cependant l'informatique quantique apporte aussi une perspective en ce sens.

	Classic Mc Eliece	Bike			HQC
Niveau		I	II	III	
clé public	1044992	8188	4094	9033	4616
clé privée	13892	548	548	532	764
Chiffré	240	8188	4094	9033	4616

TABLE 4.2 – Tableau comparatif des candidats sur le niveau de sécurité 5

Chapitre 5

Quelques implémentations

Chapitre 6

Conclusion et Perspectives

Au terme de notre recherche nous avons vu que le NIST (National Institut of Standardization and Technology) a lancé son concours international dans l'optique de sélectionner les meilleurs algorithmes chiffrements post-quantiques basés sur les codes correcteurs d'erreur capables de résister à la puissance de la machine quantique.

De cette compétition qui était à son 3 tours en 2021, il a été retenu 3 candidats qui sont le classic McEliece, Bike (Bit flipping key encryption), Hamming quasi-cyclic suivant les critères (la sécurité, la performance, la flexibilité, la simplicité de mise en oeuvre et la qualité des implémentations proposées). Chacun a ses avantages et ses inconvénients.

Premièrement parlant de classic McEliece, il est fait à partir des codes de Goppa binaire. Il possède le couple (\tilde{G}, t) comme clé publique tels que $\tilde{G} = [I_{n-k}|A]$ une matrice systématique avec t , le poids de Hamming du vecteur à chiffrer et (s, Γ) , sa clé privée telles s , une chaîne aléatoire de n -bits et $\Gamma = (g, \alpha_1, \dots, \alpha_n)$ avec g , un polynôme minimal irréductible et $\alpha_1, \dots, \alpha_n$

Du point de vue de la sécurité les meilleures attaques ont une complexité exponentielle à l'instar des attaques de décodage par ensemble d'information (ISD) classiques et quantiques. Celles-ci se basant sur diverses méthodologies à savoir avec technique de représentation ou représentation étendue ou bien par la recherche du voisin le plus proche (Nearest Neighbor) tendent à réduire cette complexité exponentielle dont l'exposant est appelé exposant de Prange. Ainsi les meilleurs ISD sont quantiques imbriquant l'algorithme de Grover qui a une complexité $O(\sqrt{N})$ avec $N = 2^n$. Ces ISD ont donné une réduction quadratique de l'exposant de Prange. Cependant face au développement fulgurant du monde de la technologie et des réseaux il est opportun que l'on pense à ébaucher de nouveaux circuits classiques implémentant des candidats.

Bibliographie

- [1] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zemor, "BIKE : Bit Flipping Key Encapsulation Round 3 Submission." Available online : <https://bikesuite.org/>, 2020. Accessed on 17 February 2020
- [2] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zemor, "BIKE : Bit Flipping Key Encapsulation Round 3 Submission." Available online : <https://bikesuite.org/files/v4.o/BIKEspec.2020.05.03.1.pdf>, 2020. Accessed on 23 May 2020.
- [3] D. J. Bernstein, T. Chou, T. Lange, R. Misoczki, R. Niederhagen, E. Persichetti, P. Schwabe, J. Szefer, and W. Wang, "Classic McEliece : Conservative Code-based Cryptography 30 March 2019." <https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/>
- [4] D. J. Bernstein, "Grover vs. McEliece," in International Workshop on PostQuantum Cryptography (PQCRYPTO), vol. 6061 of Lecture Notes in Computer Science, pp. 73–80, Springer, 2010.
- [5] Daniel J. Bernstein, Tung Chou, and Peter Schwabe. McBits : Fast constant-time code based cryptography. In Guido Bertoni and Jean-Sébastien Coron, editors, Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings, volume 8086 of Lecture Notes in Computer Science, pages 250–272. Springer, 2013.
- [6] Bernstein, D. J., Jeffery, S., Lange, T., and Meurer, A. Quantum algorithms for the subset sum problem. In Post-Quantum Cryptography 2011 (Limoges, France, June 2013), vol. 7932 of Lecture Notes in Computer Science, pages 1–14. Springer, 2013.
- [7] John-Marc Desmarais "Error Correcting Codes in Post-Quantum Cryptography" A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements for the degree of Master of Science Ottawa Carleton Institute Aug. 2020

Department of Mathematics Carleton University Ottawa, Ontario, Canada August 2020

- [8] N. Drucker, S. Gueron, and D. Kostic, "Fast Polynomial Inversion for Post Quantum QC-MDPC Cryptography," IACR Cryptol. ePrint Arch., vol. 2020, p. 298, 2020.
- [9] R. G. Gallager. Low-Density-Parity-Check Codes M.I.T Press, 1963
- [10] Bruno Grenet, *Introduction aux algorithmes probabilistes*, Septembre 2018.

- [11] T. Itoh and S. Tsujii, "A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases," *Information and Computation*, vol. 78, no. 3, pp. 171–177, 1988.
- [12] E. Prange, "The Use of Information Sets in Decoding Cyclic Codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5-9, 1962.
- [13] N. Drucker, S. Gueron, and D. Kostic, "QC-MDPC Decoders with Several Shades of Gray," in *International Conference on Post-Quantum Cryptography*, vol. 12100 of *Lecture Notes in Computer Science*, pp. 35–50, Springer, 2020.
- [14] Dumer, I. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory (Moscow, 1991)*, pp. 50–52.
- [15] N. Drucker, S. Gueron, and D. Kostic, "On Constant-time QC-MDPC Decoding with Negligible Failure Rate," *Technical Report, Cryptology ePrint Archive*, Report 2019/1289, 2019.
- [16] N. Aragon, P. S. L. M. Barreto, S. Bettaleb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zemor, "BIKE : Bit Flipping Key Encapsulation Round 3 Submission." Available online : <https://bikesuite.org/files/v4.0/BIKEspec.2020.05.03.1.pdf>, 2020. Accessed on 23 May 2020.
- [17] J. Chaulet, Etude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques. PhD thesis, Paris 6, 2017.
- [18] N. Aragon, P. S. L. M. Barreto, S. Bettaleb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, and G. Zemor, "BIKE : Bit Flipping Key Encapsulation Round 1 Submission." Available online : <https://bikesuite.org/>, 2017. Accessed on 5 July 2020.
- [19] M. Baldi, QC-LDPC Code-based Cryptography. Springer Science and Business, 2014.
- [20] A. Becker, A. Joux, A. May, A. Meurer : "Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding", In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 520–536, 2012.
- [21] D. J. Bernstein : "Grover vs. McEliece", In *Post-Quantum Cryptography 2010 (2010)*, N. Sendrier, Ed., vol. 6061 of *Lecture Notes in Comput. Sci.*, Springer, pp. 73–80, 2010.
- [22] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.
- [23] Gilbert N. Dione, *Schéma d'identification et Mécanisme d'encapsulation de clé*, Thèse de doctorat Unique, Université Cheikh A. DIOP, 30 novembre 2020
- [24] N. Drucker, S. Gueron, and D. Kostic, " On Constant-time QC-MDPC Decoding with Negligible Failure Rate", *Technical Report, Cryptology ePrint Archive*, Report 2019/1289, 2019.
- [25] A. Esser, E. Bellini : "Syndrome decoding estimator", In : *IACR International Conference on Public-Key Cryptography*. Springer, Cham, pp. 112–141, 2022
- [26] A. Esser, S. Ramos-Calderer, E. Bellini, J. I. Latorre, M. Manzano : "Hybrid Decoding—Classical-Quantum Trade-Offs for Information Set Decoding", *Cryptology ePrint Archive*, 2022.

- [27] G. Kachigar and J.P. Tillich : “Quantum information set decoding algorithms”, In : *International Workshop on Post-Quantum Cryptography*. Springer, Cham, pp.69-89, 2017
- [28] P. Lee and E. Brickell : “An observation on the security of McEliece’s public-key cryptosystem”, In *Advances in Cryptology—EUROCRYPT’88*, C. Günter, Ed. New York : Springer-Verlag, p.275, 1988.
- [29] C. Londahl, T. Johansson, M. K. Shooshtari, M. Ahmadian-Attari, and M. R. Aref, “Squaring Attacks on McEliece Public-key Cryptosystems Using Quasi cyclic Codes of Even Dimension,” *Designs, Codes and Cryptography*, vol.80,no.2,pp.359–377,2016.
- [30] Mr Ibrahima MBAYE, *Memoire de DEA Mathématiques, LA Théorie des codes correcteurs d’erreurs*, 18 juillet 2009
- [31] R. Misoczki, J. -P. Tillich, N. Sendrier, and P. S. L. M. Barreta, “MDPCMEliece : New McEliece variants from moderate density parity-check codes,” in *Proc, IEEE Int. Symposium Inf : Theory - ISIT* , 2013 , pp. 2069-2073
- [32] A. May, A. Meurer, E. Thomae : “Decoding random linear codes in $O(2^{0.054n})$ ”, In *International Conference on the Theory and Application of Cryptology and Information Security*, pp.107–124, 2011
- [33] May, A., and Ozerov, I. On computing nearest neighbors with applications to decoding of binary linear codes. In *Advances in Cryptology - EUROCRYPT 2015* (2015), E. Oswald and M. Fischlin, Eds., vol. 9056 of *Lecture Notes in Comput. Sci.*, Springer, pp. 203–228.
- [34] R.J. McEliece, A public-key cryptosystem based on algebraic coding theory, *JPL DSN Progress Report 4244* (1978), 114-116.
- [35] C. A. Melchor, N. Aragon, S. Bettaiieb, L. Bidoux, O. Blazy, J. Bos, J.-C. Deneuville, P. Gaborit, E. Persichetti, J.-M. Robert, P. V´eron, and G. Z´emor, “Hamming Quasi-Cyclic (HQC),” *Technical Report, National Institute of Standards and Technology* 2017, 2020.
- [36] S. Narisada, K. Fukushima, S. Kiyomoto : “Multi-Parallel MMT Algorithm for Solving High-Dimensional SDP”, *The 2022 Symposium on Cryptography and Information Security SCIS2022*, 4A2-1, 2022.
- [37] R. Overbeck, *Public Key Cryptography Based on Coding Theory*. PhD thesis, Technische Universität, 2007. Retrieved March 3, 2020.
- [38] S. Perriello, A. Barengi, G. Pelosi : “A complete quantum circuit to solve the information set decoding problem”, In : *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, pp.366-377, 2021.
- [39] C. E. Shannon, “A mathematical theory of communication,” *Bell Systems Technical Journal*, vol.27,pp.379–423,623–656, July 1948.
- [40] Charles, S., Ferreol, M., Chaumot, A., et Pery, A.R.R. (2004) Food availability effect on population dynamics of the midge *Chironomus riparius* : a Leslie modeling approach. *Ecological Modelling*, **175**, 217-229.
- [41] Prange, E. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory* 8, 5 (1962), 5–9
- [42] www.ssi.gouv.fr/actualite/selection-par-le-nist-de-futurs-standards-en-cryptographie-post-quantique/

