N° d'inscription : 830/2021/UA/ED STII/TASI Année universitaire : 2021





UNIVERSITÉ D'ANTANANARIVO

ECOLE DOCTORALE EN SCIENCES ET TECHNIQUES DE L'INGENIERIE ET DE L'INNOVATION

Équipe d'Accueil Doctorale en Télécommunication, Automatique, Signal et Images

Laboratoire de recherche en Télécommunication, Automatique, Signal et Images

THÈSE de DOCTORAT

pour l'obtention du **DIPLOME NATIONAL** ayant pour :

Grade :	**DOCTEUR**
Institution support:	**UNIVERSITE D'ANTANANARIVO**
Domaine :	**Sciences de l'Ingénieur**
Thématique :	**Sciences et Techniques de l'Ingénierie et de l'Innovation**
Spécialité :	**Télécommunication**

Titre : **CONTRIBUTION À LA MODÉLISATION ET À L'ÉVALUATION DE LA PERFORMANCE DES CRYPTOSYSTÈMES POST QUANTIQUES À BASE DE MLWE - CAS DE CRYSTALS KYBER ET DILITHIUM**

Par : **RAZAFIMAHEFA Fenofahasoavana Ombana**

Soutenu :	**Le 27 avril 2022, devant la Commission d'Examen composée de :**
Président de Jury :	**Mr RANDIMBINDRAINIBE Falimanana, Professeur Titulaire**
Examinateurs :	**Mr RAKOTOMIRAHO Soloniaina, Professeur** **Mr RANDRIAMITANTSOA Andry Auguste, Professeur**
Rapporteur externe :	**Mr RATIARISON Adolphe Andriamanga, Professeur Emérite**
Rapporteur interne :	**Mr RANDRIAMAROSON Rivo Mahandrisoa, Maître de Conférences - HDR**
Directeur scientifique :	**Mr RANDRIAMITANTSOA Paul Auguste, Professeur Titulaire**
Codirecteur scientifique :	**Mr RANDRIANANDRASANA Marie Emile, Maître de Conférences**

REMERCIEMENTS

Cette thèse décrit les travaux de recherche qui ont été effectués au Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (LR-TASI) au sein de l'Equipe d'Accueil Doctorale en TASI, de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation (ED-STII) sous l'Institution de l'École Supérieure Polytechnique d'Antananarivo (ESPA).

J'adresse mes sincères remerciements à Monsieur RAVELOMANANA Raoul Mamy, Professeur Titulaire, Président de l'Université d'Antananarivo.

J'adresse mes profonds remerciements à Messieurs les membres du jury qui ont accepté de rapporter et d'examiner cette thèse malgré leurs innombrables occupations :

- Monsieur RATIARISON Adolphe Andriamanga, Professeur Emérite, Responsable du Laboratoire de Recherche en Dynamique de l'Atmosphère, du Climat et des Océans (DyACO), membre de l'Ecole Doctorale en Physique et Applications, en sa qualité de rapporteur externe ;

- Monsieur RANDRIAMAROSON Rivo Mahandrisoa, Maître de Conférences - HDR, Membre du Laboratoire de Recherche en Systèmes Embarqués, Instrumentation et Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE) de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, en sa qualité de rapporteur interne ;

- Monsieur RANDIMBINDRAINIBE Falimanana, Professeur Titulaire, Responsable de l'Equipe d'Accueil Doctorale en Sciences Cognitives et Applications (SCA) de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, président du jury ;

- Monsieur RAKOTOMIRAHO Soloniaina, Professeur, Responsable du Laboratoire de Recherche en Systèmes Embarqués, Instrumentation et Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE) de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, examinateur ;

- Monsieur RANDRIAMITANTSOA Andry Auguste, Professeur, Responsable du Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (TASI) de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, examinateur ;

Je tiens à remercier chaleureusement :

- Monsieur RANDRIAMITANTSOA Paul Auguste, Professeur Titulaire, Directeur de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, Responsable de l'Equipe d'Accueil Doctorale en Télécommunication, Automatique, Signal et Images (TASI), en sa qualité de Directeur scientifique ;

- Monsieur RANDRIANANDRASANA Marie Emile, Maître de Conférences, Chef de la Mention Télécommunication au sein de l'Ecole Supérieure Polytechnique d'Antananarivo – Annexe Antsirabe, et Membre du Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (TASI) de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation, en sa qualité de Codirecteur scientifique ;

merci d'avoir accepté de diriger ensemble cette thèse, de m'avoir conseillé durant la réalisation de ce travail. Vous avez su vous montrer disponibles tout au long de mes travaux de recherche.

J'adresse tout naturellement mes remerciements à tous les Enseignants et le personnel administratif et technique de l'Ecole Supérieure Polytechnique d'Antananarivo et de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation.

J'exprime ma profonde gratitude à ma famille, plus particulièrement à mon épouse Marinah, mon garçon Noah, et mes deux filles, Yael et Yona ; pour leurs précieux soutien et encouragements tout au long de ces années d'études.

J'adresse mes remerciements à tous mes proches et amis pour leur support.

Enfin, je ne saurai oublier toutes les personnes qui, par leurs aides, ont permis l'élaboration de cette thèse.

Merci à tous.

TABLES DES MATIERES

REMERCIEMENTS ..... **i**

TABLES DES MATIERES ..... **iii**

NOTATIONS ..... **xv**

1. Minuscules latines ..... **xv**

2. Majuscules latines ..... **xviii**

3. Minuscules grecques ..... **xxi**

4. Majuscules grecques ..... **xxii**

5. Notations spéciales ..... **xxii**

6. Abréviations ..... **xxvi**

LISTE DES TABLEAUX ..... **xxviii**

LISTE DES FIGURES ..... **xxx**

INTRODUCTION GENERALE ET PRESENTATION DU PROBLEME **1**

1. Problématiques **1**

2. Objectifs **1**

CHAPITRE 1 LA CRYPTOLOGIE, OU CRYPTOGRAPHIE ET CRYPTANALYSE, DE L'ERE PRE-INFORMATIQUE **3**

1.1 Introduction **3**

1.2 Chiffrement par substitution **3**

1.2.1 Chiffrement mono-alphabétique ***3***

1.2.1.1 Chiffrement par addition 3

a. Chiffrement *4*

b. Déchiffrement *4*

c. Faiblesse *4*

1.2.1.2 Chiffrement par multiplication 5

a. Chiffrement *5*

b. Déchiffrement *5*

c. Faiblesse *6*
1.2.1.3 Chiffrement affine 7
a. Chiffrement *8*
b. Déchiffrement *8*
c. Atbash Cipher *8*
d. Faiblesse *9*
1.2.2 Chiffrement homophonique ***9***
1.2.2.1 Chiffrement 9
1.2.2.2 Phi test ou calcul de l'indice de coïncidence 10
a. Indice de coïncidence pour l'alphabet de A à Z et uniformément distribué *10*
b. Indice de coïncidence pour l'alphabet de A à Z dans la langue anglaise *10*
1.2.3 Chiffrement affine de Hill, un chiffrement type polygraphique ***11***
1.2.3.1 Chiffrement 11
1.2.3.2 Déchiffrement 12
1.2.3.3 Faiblesse 13
1.2.4 Chiffrement poly-alphabétique ***15***
1.2.4.1 Chiffre d'Alberti 15
a. Présentation *16*
b. Chiffrement *17*
c. Faiblesse et/ou avantage *18*
1.2.4.2 Du Tabula Recta au chiffre de Vigenère 18
a. Tabula Recta *18*
b. Chiffre de Vigenère *19*
c. Faiblesse *20*
d. Indice de coïncidence pour un chiffre à clé périodique *21*
e. Superposition et réduction *23*
f. Amélioration de la robustesse du chiffre de Vigenère *25*
1.2.5 Machines cryptographiques ***26***
1.2.5.1 Pinwheel 26

a. Description *26*

b. Fonctionnement *28*

c. Modèle mathématique *29*

d. Cryptanalyse des machine de type pinwheel *31*

1.2.5.2 Rotor 33

a. Description *34*

b. Modèle mathématique *36*

c. Enigma, un type de machine cryptographique à rotor *37*

d. La bombe ou machine d'aide à la cryptanalyse d'Enigma *39*

1.3 Chiffrement par transposition 41

1.3.1 Scytale 41

1.3.2 Méthodes de chiffrement dérivées du remplissage d'une forme géométrique 42

1.3.2.1 Idée de base 42

1.3.2.2 Chiffre Rail fence 43

1.3.3 Chiffre par permutation 43

1.3.3.1 Définition et fonctionnement 43

1.3.3.2 Modélisation mathématique 44

1.3.3.3 Renforcement d'un chiffre par permutation 45

a. Extension *45*

b. Produit de chiffre par permutation *45*

c. Combinaison d'un chiffre colonnaire ou columnar cipher avec un chiffre par permutation *46*

1.3.4 Cryptanalyse d'un chiffre de type colonnaire, à permutation ou la combinaison des deux 47

1.3.4.1 Passer de la clé k désignant le nombre de lignes pour le chiffre scytale, à une clé k représentant le nombre de colonnes pour un chiffre colonnaire 47

1.3.4.2 Proportion voyelles/consonnes pour mieux aider à la détection de la clé k 47

1.3.4.3 Détection de l'ordre de permutation correcte ou la clé de permutation m 49

1.4 Conclusion 51

CHAPITRE 2 THÉORIE DE L'INFORMATION, THÉORIE DES NOMBRES ET

MATHÉMATIQUE DISCRÈTE **52**
2.1 Introduction **52**
2.2 Théorie de l'information **52**
2.2.1 Claude Shannon ***52***
2.2.1.1 Quelques définitions 52
2.2.1.2 Théorèmes 53
a. Premier théorème ou théorème du codage source *53*
b. Deuxième théorème ou théorème du codage canal *60*
2.2.2 Concepts de base autour de la théorie de l'information ***62***
2.2.2.1 Entropie d'une information 62
a. Définition *62*
b. Continuité *63*
c. Symétrie *64*
d. Maximum *64*
e. Additivité *65*
f. Positive *66*
g. Evènement à probabilité 0 n'influence pas la valeur de l'entropie *67*
2.2.2.2 Confusion et diffusion 67
a. Confusion *67*
b. Diffusion *67*
2.2.2.3 Transformation complète et effet d'avalanche 67
a. Transformation complète *67*
b. Effet d'avalanche *68*
c. Critère d'avalanche strict *68*
2.2.2.4 Distance et poids de Hamming 68
a. Distance de Hamming *68*
b. Poids de Hamming *69*
2.2.2.5 Principe de Kerckhoffs et/ou maxime de Shannon 69
2.2.3 Mathématique binaire ***69***

2.2.3.1 Correspondance nombre binaire nombre décimal 69
2.2.3.2 Opérations binaires 70
a. OR *70*
b. AND *71*
c. XOR *71*
d. NOT *71*
2.3 Théorie des nombres et mathématique discrète 72
2.3.1 Classification des nombres 72
2.3.1.1 Nombres entiers naturels 72
2.3.1.2 Nombres entiers relatifs 72
2.3.1.3 Nombres rationnels 72
2.3.1.4 Nombres réels 73
2.3.1.5 Nombres complexes 73
2.3.2 Quelques opérations et/ou propriétés importantes 73
2.3.2.1 Divisibilité et algorithme de division 73
a. Divisibilité *73*
b. Algorithme de division *74*
2.3.2.2 Plus Grand Commun Diviseur ou PGCD et Algorithme d'Euclide 74
a. Plus Grand Commun Diviseur ou PGCD *74*
b. Algorithme d'Euclide *75*
2.3.2.3 Arithmétique modulaire 77
a. Modulo *77*
b. Quelques propriétés de congruence *77*
c. Opération arithmétique modulaire *78*
d. Quelques propriétés de l'arithmétique modulaire *79*
2.3.2.4 Algorithme d'Euclide revisité et Algorithme d'Euclide Etendu 80
a. Algorithme d'Euclide revisité *80*
b. Algorithme d'Euclide étendu *80*
2.3.2.5 Théorème des restes chinois 82

2.3.3 Nombres premiers ***85***

2.3.3.1 Factorisation 85

2.3.3.2 Quelques propriétés 85

a. Factorisation de produit de deux nombres entiers *86*

b. Divisibilité de deux nombres *86*

c. Plus grand commun diviseur ou PGCD de deux nombres *87*

2.3.3.3 Théorème de Fermat, indicatrice (ou *Totient function* en anglais) et théorème d'Euler 87

a. Théorème de Fermat *88*

b. Indicatrice d'Euler *88*

c. Théorème d'Euler *89*

2.3.3.4 Test de primalité, algorithme de Miller-Rabbin 90

a. Quelques propriétés utiles *91*

b. Implémentation de l'algorithme *93*

c. Usage répété de l'algorithme de Miller-Rabin *94*

2.4 Conclusion **95**

CHAPITRE 3 LATTICE, UN DES FONDEMENTS MATHEMATIQUES DES CRYPTOSYSTEMES POST-QUANTIQUES **96**

3.1 Introduction **96**

3.2 Ordinateur quantique, cryptographie quantique et post-quantique **97**

3.2.1 Ordinateur quantique ***97***

3.2.1.1 Bits quantiques ou qubits 98

a. Amplitude d'un simple qubit *98*

b. Amplitude d'un ensemble de qubits *99*

3.2.1.2 Quantum gates ou portes quantiques 99

a. Portes quantiques comme multiplication dans les matrices *100*

b. Porte quantique d'Hadamard *101*

c. Portes quantiques, des matrices unitaires *101*

3.2.1.3 Accélération quantique ou quantum speedup en anglais 102

a. Accélération quadratique dans le cas de la recherche d'un élément parmi n éléments donnés ... *102*

b. Accélération quadratique et problème de Simon (Simon's problem en anglais) ... *102*

c. Menace qu'apporte l'algorithme de Shor ... *104*

d. Algorithme de Shor et problème de factorisation ... *104*

e. Algorithme de Shor et logarithme discret ... *105*

f. Algorithme de Grover ... *105*

3.2.1.4 Construire un ordinateur quantique, une tâche difficile ... 106

3.2.2 Algorithme cryptographique post-quantique ... ***107***

3.2.2.1 Groupes de cryptographies post-quantiques ... 107

3.2.2.2 Cryptographie à base de treillis ou lattice-based cryptography en anglais ... 108

3.2.2.3 Cryptographie à base de codes ou code-based cryptography en anglais ... 109

a. Généralités sur les cryptosystèmes à base de codes ... *109*

b. Cryptosystème de McEliece comme exemple ... *109*

3.2.2.4 Cryptographie multivariée ou multivariate cryptography en anglais ... 111

a. Problème NP-difficile ou NP-hard en anglais ... *111*

b. Généralités sur les cryptographies multivariées ... *112*

c. Algorithme de signature UOV (pour Unbalanced Oil and Vinegar) comme exemple ... *113*

3.3 Généralités sur les lattices ... **116**

3.3.1 Lattices ... ***116***

3.3.1.1 Déterminant ... 120

3.3.1.2 Minimums successifs ou Successive minima en anglais ... 123

3.3.1.3 Théorème de Minkowski ... 127

3.3.2 Problèmes mathématiques autour des lattices ... ***131***

3.3.2.1 Shortest Vector Problem ou SVP ... 131

3.3.2.2 Closest Vector Problem ou CVP ... 131

3.3.2.3 SVP approximatif ou Approximative Shortest Vector Problem pour son nom en anglais ... 132

3.3.2.4 CVP approximatif ou Approximative Closest Vector Problem pour son nom en

anglais 133
3.4 Algorithme approximatif 133
3.4.1 Résolution approximative du SVP dans la dimension n 133
3.4.1.1 Bases réduites 134
3.4.1.2 Algorithme de réduction de base LLL 136
3.4.1.3 Analyse du temps d'exécution 138
a. Majoration du nombre d'itérations 139
b. Majoration du temps d'exécution de chaque itération 141
3.4.1 Résolution approximative du CVP dans la dimension n 143
3.5 Conclusion 147
CHAPITRE 4 MLWE OU LEARNING-WITH-ERRORS IN MODULE LATTICES – OUTIL MATHEMATIQUE DU CRYPTOSYSTEME CRYSTALS KYBER/DILITHIUM 148
4.1 Introduction 148
4.2 LWE ou Learning With Error 149
4.2.1 Préliminaires 149
4.2.1.1 Notations en Big O 149
4.2.1.2 Dual lattice 150
4.2.1.3 Transformation de Fourier d'une probabilité de distribution D sur une lattice L 150
4.2.2 Problème learning from parity with error 152
4.2.3 Problème Learning With Error 153
4.2.4 Aperçu sur la démonstration du Théorème 4.01 155
4.2.4.1 Discrete Gaussian distribution DL,r et smoothing parameter $\eta\varepsilon(L)$ 155
4.2.4.2 Problème discret Gaussian sampling (DGS) 156
4.2.4.3 Algorithme d'échantillonnage à partir de DL,r 157
4.3 Le ring-LWE ou ring-Learning With Error 160
4.3.1 Définition 161
4.3.2 Efficacité 162
4.3.3 Sécurité 162

4.4 Le MLWE ou Learning With Error over Module lattices 163

4.4.1 Preliminaires 164

4.4.1.1 Quelques théories algébriques des nombres 164

a. Corps des nombres et corps cyclotomique 164

b. Complexe embeddings 164

c. Espace H 165

d. Ideals 166

e. Isomorphisme d'anneaux quotients 166

f. Modules 166

4.4.1.2 Quelques notions additionnelles sur les lattices 168

a. GIVP, une généralisation du SIVP 168

b. Ideal et module lattice 168

4.4.2 Mesures Gaussiennes 169

4.4.3 Réduction du Mod-GIVP au M-LWE (Learning With Errors over Modules) 171

4.4.3.1 Variantes du LWE 171

a. LWE 171

b. R-LWE 173

c. LWE over module 173

4.4.3.2 Réduction du Mod-GIVP au M-SLWE 175

4.5 Cryptosystème CRYSTALS 181

4.5.1 Préliminaires 182

4.5.1.1 Classe d'indiscernabilité pour les cryptosystèmes type KEM, le préfixe IND- (de INDistinguishability en anglais) 182

a. IND-CPA: INDistinguishability under Chosen Plaintext Attack 182

b. IND-CCA1: INDistinguishability under (non adaptive) Chosen Ciphertext Attack 183

c. IND-CCA2: INDistinguishability under adaptive Chosen Ciphertext Attack 184

4.5.1.2 xUF-CMA, catégories de sécurité de cryptosystèmes type signature 184

a. EUF-CMA 184

b. SUF-CMA *185*

4.5.2 CRYSTALS Kyber ***186***

a. Généralités *186*

b. Justifications de conception *186*

4.5.3 CRYSTALS Dilithium ***189***

a. Généralités *189*

b. Aperçu de l'approche de base *190*

4.6 Apport personnel et innovation des travaux de recherches **192**

4.7 Conclusion **193**

CHAPITRE 5 APPLICATION DU MLWE DANS LE CRYPTOSYSTEME CRYSTALS - KYBER COMME KEM ET DILITHIUM COMME SIGNATURE **194**

5.1 Introduction **194**

5.2 Cryptosystème type KEM, CRYSTALS Kyber **194**

5.2.1 Spécification écrite ***194***

5.2.1.1 Préliminaires et notations 194

5.2.1.2 Spécification de Kyber. CPAPKE 201

5.2.1.3 Spécification de Kyber. CCAKEM 204

5.2.1.4 Jeu de paramètres de Kyber 205

5.2.1.5 Justifications additionnelles de conception 207

5.2.2 Analyses de performance ***209***

5.2.2.1 Considérations de mise en œuvre et compromis 210

5.2.2.2 Performances de Kyber sur les processeurs Intel Haswell 212

5.2.2.3 Performance of Kyber on ARM Cortex-M4 CPUs 215

5.2.3 Niveau de sécurité attendu ***218***

5.2.3.1 Justification des estimations de sécurité 218

5.2.3.2 Hypothèse de sécurité 219

a. Réduction étroite du MLWE dans le ROM *219*

b. Réduction non étroite de MLWE dans le QROM *220*

5.2.3.3 Estimation de niveau de sécurité 221

5.2.3.4 Propriétés de sécurité additionnelles 224
a. Sécurité iterative *224*
b. Attaques par canal auxiliaire *224*
c. Attaques multi-cibles *226*
d. Résilience face aux erreurs de manipulation *227*
5.3 Cryptosystème type signature, CRYSTAL Dilithium 227
5.3.1 Introduction 227
5.3.1.1 Génération de clé, signature et vérification 227
5.3.1.2 Information additionnelle 228
a. Points importants de la phase d'implémentation *229*
b. Sécurité *230*
5.3.2 Operations basiques 235
5.3.2.1 Opérations dans l'anneau 235
5.3.2.2 Représentation dans le domaine NTT 235
5.3.2.3 Hachage 236
5.3.2.4 Bits de poids fort/faible, indices *h* (de l'anglais hints) 238
5.3.3 Signature 240
5.3.3.1 Notes de mise en œuvre et compromis en matière d'efficacité 243
5.3.3.2 Signatures déterministes et aléatoires 243
5.3.3.3 Preuve du fait que la présente signature est correcte 244
5.3.3.4 Nombre d'itérations 244
5.4 Apport personnel et innovation des travaux de recherches 246
5.5 Conclusion 246
CONCLUSION GENERALE 248
ANNEXES 250
Annexe 1 Publication de notoriété nationale et internationale 250
A1.1 La cryptologie, ou cryptographie et cryptanalyse, de l'ère pré-informatique 250
A1.2 Théorie de l'information, théorie des nombres et mathématique discrète 266
A1.3 Application du MLWE dans le cryptosystème CRYSTALS - Kyber comme KEM et

Dilithium comme signature *279*

A1.4 Application of the MLWE in the cryptosystem CRYSTALS – KYBER as KEM and Dilithium as signature *298*

A1.5 Certificat de la publication de notoriété internationale *311*

Annexe 2 Communications scientifiques au sein de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation **312**

A2.1 Communication scientifique année D1, semestre 11 ..... *312*

A2.2 Communication scientifique année D1, semestre 12 ..... *316*

A2.3 Communication scientifique année D2, semestre 13 ..... *321*

A2.4 Communication scientifique année D2, semestre 14 ..... *326*

A2.5 Communication scientifique année D3, semestre 15 ..... *332*

REFERENCES **336**

FICHE DE RENSEIGNEMENT **340**

RESUMES ET MOTS CLES **341**

ABSTRACT AND KEYWORDS **341**

NOTATIONS

1. Minuscules latines

a-z	Lettres en clair
a	Un nombre réel donné (*) Niveaux de quantification d'un signal (*)
$\boldsymbol{a_k}$	k-ième lettre de l'alphabet
b	Un nombre réel donné
$\boldsymbol{b_n}$	n-ième vecteur de base d'une lattice. $n \geq 1$
$\boldsymbol{b'_n}$	Un autre n-ième vecteur de base d'une lattice
$\boldsymbol{b^*_n}$	n-ième vecteur de base orthogonalisé d'une lattice
	Texte chiffré (*)
$\boldsymbol{c}$	Une constante donnée (*) Nombre de consonnes (*)
$\boldsymbol{c'}$	Une valeur intermédiaire de texte chiffré
$\boldsymbol{c_i}$	i-ième lettre de l'alphabet (*) i-ième lettre d'un bloc de n-graphe (*)
$\boldsymbol{d}$	Un message donné (*) Variable représentant le module d'un vecteur lattice (*)
$\boldsymbol{d_k}$	Entier associé à la base B d'une lattice après k itérations
$\boldsymbol{e_i}$	i-ième vecteur unitaire standard
$\boldsymbol{f}$	Une fonction donnée
$\boldsymbol{f^{(n)}}$	n-ième polynôme quadratique
$\boldsymbol{f(k, P)}$	Fonction représentant un chiffre de clé k, appliqué à la lettre en clair P
$\boldsymbol{f(x)}$	Fonction de variable d'entrée x

$\boldsymbol{g}$	Un nombre entier donné
$\boldsymbol{h(x,y)}$	Distance de Hamming entre les deux textes x et y
$\boldsymbol{i}$	Rang de la lettre dans l'alphabet. Valeur allant de 1 au nombre z de lettres de l'alphabet (*)
	Rang de la lettre dans le message (*)
	Nombre imaginaire i, qui par définition est égal à $\sqrt{-1}$ (*)
	Un nombre entier donné, pour numérotation (*)
$\boldsymbol{j}$	Un nombre entier donné, pour numérotation
$\boldsymbol{k}$	Un nombre entier donné
$\boldsymbol{k_{i,j}}$	i-ième ligne et j-ième colonne de la matrice de clé K
$\boldsymbol{k, l, m}$	Clé du chiffre (*)
$\boldsymbol{l}$	Longueur de la clé du chiffre (*)
	Rang d'un sous ensemble d'un alphabet variant de $1, \dots, K_l$ (*)
$\boldsymbol{l_i}$	Longueur de la i-ième clé
$\boldsymbol{\ell_p}$	Norme ℓ caractérisé par l'entier p, $p \geq 1$
	Une valeur d'entrée d'une fonction (*)
$\boldsymbol{m}$	Message à chiffrer (*)
	Rang d'une lattice. Rank en anglais (*)
$\boldsymbol{m_i}$	i-ième élément du vecteur clé M
$\boldsymbol{n}$	Taille du polygraphe utilisé. 2 pour le cas du digraphe (*)
	Longueur du message (*)
	Un nombre entier donné (*)
	Dimension d'une lattice (*)
$\boldsymbol{o}$	Un nombre entier donné

$\boldsymbol{p}$	Un nombre entier donné
$\boldsymbol{p_i}$	i-ième lettre en clair
$\boldsymbol{p_k}$	Probabilité d'apparition de la lette a_k de l'alphabet $\mathcal{A}$ k-ième nombre premier résultant de la factorisation d'un nombre entier (*)
$\boldsymbol{p_{max}}$	Probabilité maximum
$\boldsymbol{p_{typ}}$	Probabilité d'un message typique
$\boldsymbol{q}$	Un nombre entier donné (*) Quotient de la division entre deux nombres (*)
$\boldsymbol{r}$	Un nombre réel donné (*) Un nombre rationnel donné (*) Taux de message en messages/s (*) Reste de la division entre deux nombres (*)
$\boldsymbol{t}$	Un nombre entier donné (*) Un vecteur cible donné (*)
$\boldsymbol{t^*}$	La composante de t orthogonal au $\mathrm{span}(b_1, \ldots, b_n)$
$\boldsymbol{v}$	Un nombre entier donné (*) Nombre de voyelles (*)
$\boldsymbol{v_n}$	n-ième vecteur associé au n-ième minimum successif λ_n
$\boldsymbol{w}$ et $\boldsymbol{w'}$	Une valeur intermédiaire de message obtenu après une transformation donnée
$\boldsymbol{x}$	Une valeur d'entrée d'une fonction (*) Message aléatoire (*)
$\boldsymbol{x_n}$	Une n-ième variable donnée (*) un n-ième vecteur lattice donné (*)
$\boldsymbol{y}$	Une valeur d'entrée d'une fonction

z	Nombre de lettres de l'alphabet (*)
z_n	Un vecteur numéro n d'une lattice

2. Majuscules latines

AND	Opération booléenne AND
A	Personne *A*, envoyeur du message
$\mathcal{A}$	Alphabet
A_k^n	Arrangement de k éléments parmi un ensemble de n éléments
A-*Z*	Lettres chiffrées
B	Matrice composée des vecteurs de base d'une lattice
B	Personne *B*, receveur du message (*) Bande passante du canal en hertz (Hz) (*)
B'	Une autre matrice composée des vecteurs de base d'une lattice
	Le texte chiffré (*)
	Un code linéaire binaire donné (*)
C	Une matrice donnée (*)
	Variable représentant la lettre chiffrée ou le message chiffré (*)
	Capacité du canal en bits par seconde (bits/s) (*)
C_k^n	Combinaison de k éléments parmi un ensemble de n éléments
$\mathbb{C}$	Ensemble des nombres complexes
E	Chiffrement (Encryption) de la machine Enigma dans sa globalité (*)
$\mathcal{F}$	Une transformation donnée
G	Une matrice génératrice
$\mathbf{H}$	Porte quantique d'Hadamard
$\mathcal{H}$	Fonction de Hachage

$\boldsymbol{H_K, H_L, H_{K_l}}$	Entropie de l'alphabet respectivement de cardinal K, L et H_{K_l}
$\boldsymbol{H(X)}$	Entropie d'un message X
$\boldsymbol{I}$	Matrice identité (*) Nombre moyen de bits par lettre (*)
$\mathbb{I}$	Ensemble des nombres irrationnels
$\boldsymbol{IC}$	Indice de coïncidence
$\boldsymbol{IC_{anglais}}$	Indice de Coïncidence pour la langue anglaise
$\boldsymbol{IC_{CCP}}$	Indice de Coïncidence pour un Chiffre à Clé Périodique
$\boldsymbol{IC_{CUD}}$	Indice de Coïncidence pour Chiffre Uniformément Distribué
$\boldsymbol{I_N}$	Nombre de bits pour représenter un message de longueur N
	Clé d'un chiffre (*)
$\boldsymbol{K}$	Matrices clé de chiffre (*)
	Nombre d'éléments d'un alphabet (*)
$\boldsymbol{K_l}$	Nombre d'élément du sous ensemble de rang l
$\boldsymbol{\mathcal{L}}$	Une lattice
$\boldsymbol{L}$	Nombre de sous-ensembles d'un alphabet
	Vecteur clé de chiffre (*)
$\boldsymbol{M}$	Nombre de messages possibles ou encore nombre de niveaux de quantifications (*)
$\boldsymbol{M}$ et $\boldsymbol{M^{-1}}$	Chiffre du rotor de gauche (L pour Left) et son inverse
$\boldsymbol{N}$	Un grand nombre entier donné (*) Nombre de lettre d'un message aléatoire (*)
$\boldsymbol{N_0}$	Nombre entier de référence
$\boldsymbol{N_k}$	Fréquence d'apparition de chaque lettre a_k pour un message de longueur N
$\mathbb{N}$	Ensemble des nombres entiers naturels

$\boldsymbol{O}$	Oil, ensemble de nombre entier
	Une matrice aléatoire de permutation (*)
$\boldsymbol{P}$	Variable représentant la lettre en clair ou le message en clair (*)
	Ensemble regroupant tous les nombres premiers p_k (*)
$\boldsymbol{\mathcal{P}}$	Une transformation donnée
$\boldsymbol{P_{err}}$	Probabilité d'erreur
$\boldsymbol{P_T}$	Probabilité de toutes les séquences typiques de messages
$\boldsymbol{PVC_{anglais}}$	Proportion Voyelles/Consonnes dans la langue anglaise
$\mathbb{Q}$	Ensemble des nombres entiers rationnels
$\boldsymbol{Q_f}$	Une porte quantique construite à base de la fonction f
$\boldsymbol{R}$	Taux de transfert du message par le canal en bit/s
$\boldsymbol{R}$ et $\boldsymbol{R^{-1}}$	Chiffre du rotor de droite (R pour Right) et son inverse
$\mathbb{R}$	Ensemble des nombres réels
	Une matrice aléatoire non singulière (inversible) (*)
$\boldsymbol{S}$	Ensemble mesurable d'un ensemble engendré par un espace vectoriel (*)
	Puissance moyenne d'un signal (*)
$\boldsymbol{S'}$	Un autre ensemble mesurable d'un ensemble engendré par un espace vectoriel
$\boldsymbol{S/N}$	Rapport signal sur bruit
$\boldsymbol{T}$	Une matrice triangulaire (*)
	Message typique (*)
$\boldsymbol{T' \subset T}$	Sous ensemble du message typique T
$\boldsymbol{T}$ et $\boldsymbol{T^{-1}}$	Chiffre du tableau de connexion (Table of connection ou plugboard) et son inverse
$\boldsymbol{\mathcal{T}}$	Une transformation donnée

$\boldsymbol{U}$	Une matrice uni-modulaire, $\det(U) = \pm 1$ (*)
	Chiffre symétrique du réflecteur (*)
$\boldsymbol{V}$	Vinegar, ensemble de nombre entier (*)
	Une matrice carrée donnée (*)
$\boldsymbol{W}$	Une matrice carrée donnée
$\boldsymbol{X}$	Ensemble de message aléatoire
	Ensemble des entiers inférieurs à n qui ne divisent par n (*)
$\boldsymbol{X_n}$	Lettre aléatoire de rang n
XOR	Opération booléenne XOR ou Exclusive OR
$\mathbb{Z}$	Ensemble des nombres entiers relatifs
$\mathbb{Z}_{\boldsymbol{N}}$	Ensemble des restes de la division par l'entier N

3. Minuscules grecques

$\boldsymbol{\alpha}$	Amplitudes d'un qubit
$\boldsymbol{\alpha_i}$	Amplitudes groupe de qubits. $i \geq 0$
$\boldsymbol{\beta}$	Amplitudes d'un qubit
$\boldsymbol{\gamma}$	Un nombre réel donné
$\boldsymbol{\delta}$	Un nombre réel donné
$\boldsymbol{\delta, \varepsilon, \varepsilon'}$	Nombre réel positif, à valeur aussi petit que l'on veut
$\boldsymbol{\lambda}$	Un nombre réel donné (*)
	Facteur multiplicatif
$\boldsymbol{\lambda_n}$	n-ième minimum successif d'une lattice
$\boldsymbol{\mu_{i,j}}$	Coefficient du processus d'orthogonalisation de Gram-Schmidt
$\boldsymbol{\pi}$	Le nombre pi soit ~3.14
$\boldsymbol{\sigma}$	Moyenne quadratique d'un signal

$\boldsymbol{\tau}$	Temps d'attaque d'un filtre passe bas idéal
$\boldsymbol{\omega}$	Période d'une fonction donnée
$\boldsymbol{\omega'}$	Période d'une fonction donnée

4. Majuscules grecques

$\Delta^2 f(X)$	Variance de la fonction $f(X)$
Λ	Une lattice
Λ_k	Une sous-lattice formée uniquement par les k premiers vecteurs de base
Λ'_k	Une autre sous-lattice formée uniquement par les k premiers vecteurs de base
Φ	Qubit
Ψ	Qubit

5. Notations spéciales

$-$	Soustraction
$(.)!$	Factorielle.
$\boldsymbol{^t}(.)$	Transposée d'une matrice
$\overline{(.)}$	Inverse multiplicative modulaire
$\prod(.)$	Produit
$(.)^{\boldsymbol{n}}$	Elevé à la puissance n
$(.)^{\boldsymbol{-1}}$	Inverse d'une fonction, matrice
$\sum(.)$	Somme
$\|\cdot\rangle$	Vecteurs dans un état quantique
$\boldsymbol{\lim_{x\to 0}}(\cdot)$	Limite quand x tend vers 0
$\boldsymbol{\tilde{f}^{(n)}}$	Une valeur transformée de la fonction $f^{(n)}$ suivant un algorithme de calcul

	donné
$\lfloor x \rceil$	Arrondissement de la valeur $x + \frac{1}{2}$ à l'entier directement inférieur ou égal à elle-même (*round half up* en anglais)
$\|\|x\|\|_p$	Norme caractérisé par l'entier p, $p \geq 1$
$\|.\|$	Longueur d'un message
$\lfloor . \rfloor$	Arrondissement a l'entier directement inférieur
$\langle . \rangle$	Moyenne ou espérance
$\|\|.\|\|: \mathbb{R}^n \longrightarrow \mathbb{R}$	$\|\|.\|\|$ une fonction de $\mathbb{R}^n$ vers $\mathbb{R}$
$\sqrt{\cdot}$	Racine carrée
$\mathbf{ln}(\cdot)$	La fonction logarithme népérien
$\mathbf{log}(\cdot)$	La fonction logarithme, la base utilisée est données par le contexte
$\mathbf{log_2}(\cdot)$	La fonction logarithme à base 2
$\mathbf{max}(\cdot)$	Maximum
$\mathbf{min}(.)$	Minimum
$\mathbb{R}^n$	L'ensemble des nombres réels de dimension n. $n \geq 1$
$\mathbb{Z}^n$	L'ensemble des nombres entiers relatifs de dimension n. $n \geq 1$
$\mathbf{det}(\boldsymbol{\Lambda})$	Déterminant de la lattice Λ
$\mathbf{det}(.)$	Déterminant d'une matrice
$\mathbf{dim}(\cdot)$	Dimension
$\widehat{\boldsymbol{G}}$	Une valeur transformée de la grandeur G suivant un algorithme de calcul donné
$\boldsymbol{P^{-1}}$	Matrice inverse de P
$\boldsymbol{A^T}$	Transposée de la matrice A
$\boldsymbol{N_k^{-1}}\ \mathbf{mod}\ \boldsymbol{n_k}$	Inverse modulaire mod n_k de N_k

$\boldsymbol{e^x}$	Exponentielle de x
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	Produit scalaire entre les deux vecteurs x et y
$\|\boldsymbol{x}\|$	Norme du vecteur x
$\overline{\boldsymbol{\mathcal{B}}}(\mathbf{0}, \boldsymbol{r})$	La sphère fermée de rayon r, centrée en 0
$\boldsymbol{\mathcal{B}_m}(\mathbf{0}, \boldsymbol{r})$	La sphère ouverte de dimension m et de rayon r, centrée en 0
$\mathbb{F}^{\boldsymbol{n}}$	Ensemble de polynômes quadratiques de dimension n
$\boldsymbol{\pi_i}(\boldsymbol{x})$	La composante de x orthogonal aux vecteurs $b_1, \ldots, b_{i-1}$
/	Division
+	Addition
$\oplus$	Opération binaire XOR ou eXclusive OR
*	Multiplication (*) Multiplication matricielle (*)
<	Inférieure
=	Egale
≠	Différent
>	Supérieure
±	Plus ou moins
×	Multiplication
∀	Quel que soit
\	Moins, minus en anglais, en termes d'ensemble
∩	Intersection, en termes d'ensemble
∪	Union
≔	Egale, par définition
≡	Identique

$\leq$	Inférieure ou égale
$\geq$	Supérieure ou égale
$\gtrsim$	Supérieur ou sensiblement égal
$\subset$	Contenu, en termes d'ensemble
$\subseteq$	Contenu ou égal, en terme d'ensemble
$\Leftrightarrow$	Si et seulement si
♣	Début de la démonstration
♦	Fin de la démonstration
$\mathbf{inf}\{\cdot\}$	Infimum
$\mathbf{Mod}$	Opération modulo
$\mathbb{P}(.)$	Probabilité
$\mathbf{span}(\boldsymbol{B})$	Espace vectoriel engendré par la matrice de base B
$\mathbf{vol}(\boldsymbol{S})$	Volume de l'ensemble mesurable S
$\mathbf{card}(.)$	Cardinal
$\mathbf{dist}(\boldsymbol{x},\boldsymbol{y})$	Distance entre deux vecteur x et y
$\mathbf{max}(.)$	Maximum
$\mathbf{min}(.)$	Minimum
$\boldsymbol{O}()$	Complexité en temps, time complexity en anglais
$\boldsymbol{A} \longleftarrow (\boldsymbol{a_1}, \dots, \boldsymbol{a_K})$	A l'ensemble des K-uplet $(a_1, \dots, a_K)$ correspond un unique entier A
$\boldsymbol{A} \longleftrightarrow (\boldsymbol{a_1}, \dots, \boldsymbol{a_K})$	Relation bijective entre l'entier A et l'ensemble des restes $(a_1, \dots, a_K)$
$\boldsymbol{E}(\boldsymbol{K},\boldsymbol{P})$	E un chiffre (cryptosystème) donné, K la clé, P texte en clair
$\boldsymbol{E}\left[\mathbf{log}\dfrac{\mathbf{1}}{\boldsymbol{p}(\boldsymbol{X})}\right]$	Espérance ou moyenne de $\log\frac{1}{p(X)}$
$\boldsymbol{TEST}(\boldsymbol{n})$	Algorithme de Miller-Rabin appliquer à l'entier n
$\boldsymbol{X}$ **minus** $\boldsymbol{Y}$	Ensemble X auquel on enlève les élément dans Y

$\boldsymbol{a \bmod b}$	a modulo b
$\boldsymbol{a \equiv r \pmod b}$	a congrue vers $r \pmod b$
$\boldsymbol{b \mid a}$	b divise a
$\boldsymbol{com(.)}$	Comatrice d'une matrice
$\boldsymbol{f: \mathcal{A} \to \mathbb{R}}$	Fonction prenant en entrée une lettre x de l'alphabet $\mathcal{A}$ et retournant une valeur réelle dans $\mathbb{R}$
$\boldsymbol{p(.)}$	Probabilité d'apparition
$\boldsymbol{x \longrightarrow \infty}$	La variable x tend vers infini
$\boldsymbol{\mathcal{P}(B)}$	Le parallélépipède formée par la base B d'une lattice
$\boldsymbol{\mathcal{B}(0, r)}$	La sphère ouverte de rayon r, centrée en 0
$\boldsymbol{\mathcal{F} \circ \mathcal{T}}$	$\circ$ indique la combinaison de deux transformations $\mathcal{F}$ et $\mathcal{T}$
$\boldsymbol{\gamma(n)}$	Facteur d'approximation dépendant de la dimension n de la lattice
$\boldsymbol{\delta}$LLL	Réduction LLL caractérisée par le facteur δ
$\boldsymbol{\phi(.)}$	Fonction Indicatrice d'Euler

6. Abréviations

AES	Advanced Encryption Standard
CVP	Closest Vector Problem
DES	Data Encryption Standard.
DLP	Discrete Logarithm Problem
ECDLP	Elliptic Curve DLP
EPR	Paradoxe EPR (du nom de leurs inventeurs Einstein-Podolsky-Rosen)
GF	Galois Field
GF	Corps de Galois ou Galois Field
IBM	International Business Machines corporation

KEM	Key Encapsulation Mechanism
LLL	Algorithme de réduction, de Lenstra–Lenstra–Lovász
LPF	Low Pass Filter.
MQ	Multivariate Quadratic equation
NIST	National Institute of Standards and Technology
NP	Non-deterministic Polynomial-time
P	Polynomial-time
PGCD	Plus Grand Commun Diviseur
PKE	Public Key Encryption
PPCM	Plus Petit Commun Multiple
RSA	Cryptosystème RSA, de leurs inventeurs Rivest–Shamir–Adleman
SNR	Signal to Noise Ratio.
SVP	Shortest Vector Problem
UOV	Unbalanced Oil and Vinegar

(*) L’usage est précisé dans le contexte pour éviter l’ambigüité.

LISTE DES TABLEAUX

Tableau 1.01 : *Exemple de message chiffré ayant une période égale à 5.* 22

Tableau 1.02 : *Quelques caractéristiques du message chiffré du Tableau 1.01.* 22

Tableau 1.03 : *Le message chiffré du Tableau 1.01 après superposition.* 24

Tableau 1.04 : *Clé résultant de OK et CURIOUSER.* 26

Tableau 1.05 : *Exemple de configuration du C-36.* 29

Tableau 1.06 : *Exemple de valeur des paramètres de notre formule de chiffrement, partant de la configuration du C-36 du Tableau 1.05.* 30

Tableau 1.07 : *Résultat de chiffrement obtenu pour le mot en clair bork répété 3 fois.* 31

Tableau 1.08 : *Substitution éffectuée selon le numéro de rotor pour la machine Enigma I.* 40

Tableau 1.09 : *Quelques manières de remplir une forme rectangulaire.* 42

Tableau 1.10 : *Exemple de chiffrement par Rail fence, pour le message en clair the rail splitter for president.* 43

Tableau 1.11 : *Exemple de chiffre par permutation en utilisant le mot clé TALE.* 44

Tableau 1.12 : *Message en clair et message chiffré utilisant la formule d'encodage* ***(****1.31****)****.* 45

Tableau 1.13 *:* *Exemple de chiffre colonnaire à clé utilisant un chiffre colonnaire de clé* $k = 4$ *(nombre de colonnes du chiffre colonnaire) combiné à un chiffre par permutation de clé* $m \equiv TALE \equiv 4132$. *Rapellons le groupement par* 5 *lettres du message chiffré pour embrouiller Eve.* 46

Tableau 1.14 : *Exemple de message chiffré par un chiffre colonnaire à clé.* 48

Tableau 1.15 : *Calcul de la variance du nombre de voyelles pour* $k = 6$. 48

Tableau 1.16 : *Superposition pour* $k = 8$. 50

Tableau 1.17 : *Méthode de contact appliqué aux couples de colonnes (I, VII) et (I, VIII). (I, VIII) a une plus forte probabilité de se suivre.* 50

Tableau 2.01 : *Correspondance entre nombre décimal et nombre binaire.* 70

Tableau 2.02 : *Conversion nombre binaire vers nombre décimal.* 70

Tableau 2.03 : *Opération binaire OR.* 70

Tableau 2.04 : *Opération binaire OR.* 71

Tableau 2.05 : *Opération binaire OR.* 71

Tableau 2.06 : *Opération binaire NOT.* 71

Tableau 2.07 : *Quelques propriétés de l'arithmétique modulaire.* 79

Tableau 2.08 : *Algorithme d'Euclide étendu.* 81

Tableau 4.01 : *Quelques notations en Big* O. 149

Tableau 5.01 : *Jeux de paramètres de Kyber.* 205

Tableau 5.02 : *Performances de Kyber sur les processeurs Intel Haswell.* 213

Tableau 5.03 : *Performances de Kyber sur les processeurs Intel Haswell.* 216

Tableau 5.04 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber.* 222

Tableau 5.05 : *Tailles de sortie, sécurité et performances de Dilithium.* 231

Tableau 5.06 : *Jeux de paramètres de Dilithium.* 245

LISTE DES FIGURES

Figure 1.01 : *Disque d'Alberti avec dans un premier temps* $m1 = 0$ *à gauche, puis* $m2 = -4\ modulo\ 26 = 22$ *s à droite.* 16

Figure 1.02 : *Tabula Recta par Johannes Trithemius et amélioré par Vigenère.* 19

Figure 1.03 : *Image réelle du C-36.* 27

Figure 1.04 : *Schéma modélisé de ses pinwheel ou roues codeuses. A gauche, curseur courant et bras correspondant inactifs. A droite, curseur et bras actifs.* 27

Figure 1.05 : *Schéma modélisé du cylindre à* **25** *barres horizontales et de ses ergots. A gauche, bras inactif. A droite, un ergot d'une barre horizontale rencontrant un bras actif.* 28

Figure 1.06 : *Une machine Enigma.* 34

Figure 1.07 : *Image réelle d'un rotor décomposé. Remarquons les fils électriques qui relient l'un à l'autre côté du disque.* 35

Figure 1.08 : *Représentation des câblages électriques d'un rotor. En position de départ à gauche, puis après un tour de pas à droite.* 35

Figure 1.09 : *Deux rotors côte à côte. A gauche, les deux rotors en position initiale pour chiffrer la première lettre en clair. Au milieu, le premier rotor ayant effectué un pas de tour, prêt à chiffrer la seconde lettre en clair. A droite, le premier rotor revenu à sa position initiale et le second rotor ayant effectué un pas de tour, prêt à chiffrer la* 26ème *lettre en clair.* 36

Figure 1.10 : *Le couple de (lettre en clair, lettre chiffré) ou (lettre chiffrée, lettre en clair) (A, D) traité par la machine Enigma.* 38

Figure 1.11 : *Une version reconstruite de la bombe qui marche réellement, exposée au musée du parc de Bletchley. La bombe possède* 3 *groupes de* 36 *rotors. Chaque rotor de la bombe simulant l'action d'un rotor de la machine Enigma.* 39

Figure 1.12 : *Une scytale* 41

Figure 1.13 : *Une vue modélisée de la scytale.* 42

Figure 2.01 : *Algorithme d'Euclide.* 76

Figure 3.01 : *Principe d'un algorithme quantique.* 100

Figure 3.02 : *Circuit d'un algorithme quantique pouvant résoudre efficacement le problème de Simon* 103

Figure 3.03 : *Une Lattice dans* $\mathbb{R}\mathbf{2}$ *où* $\boldsymbol{b}\mathbf{1}$ *et* $\boldsymbol{b}\mathbf{2}$ *en sont les vecteurs de bases.* 108

Figure 3.04 : *Diagramme d'Euler pour les classes de problèmes P, NP, NP-complet et NP-*difficile. 111

Figure 3.05 : *Machine de Turing déterministe et non déterministe.* 112

Figure 3.06 : *La même lattice que la Figure 3.03, mais cette fois ci, générée par de différents vecteurs de base* $\boldsymbol{b}\mathbf{1}'$ *et* $\boldsymbol{b}\mathbf{2}'$. 117

Figure 3.07 : *Les bases* $\boldsymbol{b}\mathbf{1}' = \boldsymbol{b}\mathbf{1} - \boldsymbol{b}\mathbf{2}$ *et* $\boldsymbol{b}\mathbf{2}' = \boldsymbol{b}\mathbf{1} + \boldsymbol{b}\mathbf{2}$ *engendrant une lattice différente que celle générée par les bases originales* $\boldsymbol{b}\mathbf{1}$ *et* $\boldsymbol{b}\mathbf{2}$. 118

Figure 3.08 : *Algorithme du plan le plus proche.* 145

Figure 4.01 : $f1r$ *pour une lattice de dimension égale à* 2. 151

Figure 4.02 : $\Psi\alpha$ *pour* $p = 127$ *et* $\alpha = 0{,}05$ *(à gauche) et* $\alpha = 0{,}1$ *(à droite). Les éléments de* $\mathbb{Z}p$ *sont arrangés sur un cercle.* 154

Figure 4.03 : $DL,2$ *(à gauche) et* $DL,1$ *(à droite) pour une lattice de dimension 2. L'axe des z représente la probabilité.* 156

Figure 4.04 : *Deux itérations de l'algorithme.* 158

Figure 4.05 : *Le LWE en termes d'algèbre linéaire.* 172

Figure 4.06 : *The M-LWE problem can be interpreted in terms of linear algebra.* 174

Figure 5.01 : *Nombre de cycles requis pour l'étape de génération de clé (Key generation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres*

de Kyber. 214

Figure 5.02 : *Nombre de cycles requis pour l'étape d'encapsulation (Encapsulation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres de Kyber.* 214

Figure 5.03 : *Nombre de cycles requis pour l'étape de décapsulation (Decapsulation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres de Kyber.* 215

Figure 5.04 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape de génération de clé (Key generation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.* 216

Figure 5.05 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape d'encapsulation (Encapsulation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.* 217

Figure 5.06 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape de décapsulation (Decapsulation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.* 217

Figure 5.07 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber. Méthodologie noyau-SVP, attaque primitive uniquement.* 223

Figure 5.08 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber. Estimation affinée pour les attaques classiques.* 224

Figure 5.09 : *Nombre de cycles requis pour l'étape de génération de clé (Gen median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.* 232

Figure 5.10 : *Nombre de cycles requis pour l'étape de signature / méthode de calcul moyenne (Sign average cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.* 233

Figure 5.11 : *Nombre de cycles requis pour l'étape de signature / méthode de calcul médian (Sign*

median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium. 234

Figure 5.12 : *Nombre de cycles requis pour l'étape de vérification (Verify median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.* 234

INTRODUCTION GENERALE ET PRESENTATION DU PROBLEME

Ko-nda ko-o ko-a ko-ndeha ko-i ko-la ko-la ko-o. Une de ces phrases dont je me rappelle, utilisée lorsque nous étions enfants. La phrase représente le message *ndao andeha ilalao* en malagasy soit *allons-jouer* en français, à laquelle plus ou moins chaque syllabe a été intercalée par le terme non signifiant *ko* (lire *cô*). Nous (moi et mes camarades) avions voulu à cette époque cacher le message à nos parents. Ces derniers sûrement étaient en mesure de déchiffrer le message mais jouaient le jeu.

Le paragraphe précédent est un exemple du fait que l'être humain était et est fasciné par l'utilité de cacher un message et de ne divulguer celui-ci qu'à ceux ou celles à qui il est destiné. Deux disciplines distinctes existent pour ce faire. D'un côté, *la stéganographie*, consiste à cacher le message dans un contenu. Correspond donc à l'exemple que nous avons vu plus haut. De l'autre, *la cryptographie*, consiste à présenter le message sous une autre forme pour que seul celui à qui il est destiné, soit en mesure de le lire. Le présent ouvrage tournera autour de la cryptographie.

D'une manière générale la cryptographie se définit comme étant l'étude des techniques permettant la communication sécurisée en présence de partie tierce que l'on nomme *Adversaire*. Dans la littérature, l'envoyeur est souvent appelé sous le nom *Alice* ou personne *A*. Le récepteur *Bob* ou personne *B*. L'adversaire est quant à lui dénommé *Eve* de l'anglais *eavesdropper*, l'*eavesdropping* se définissant comme étant l'écoute secrète de conversation privée d'autrui, sans son consentement.

Problématiques

- De nouvelles théories de décryptage ne cessent de faire surface. Comment les méthodes de cryptographie ou cryptosystèmes pourront-ils résister à cet éternel progrès du côté de l'adversaire ?

- La puissance de calcul des ordinateurs disponibles pour le grand public, voire celle que les grandes puissances dans le monde du numérique possèdent, est en perpétuelle évolution/amélioration. De quelle manière cette puissance de calcul élevée rime-t-elle avec réduction du temps nécessaire pour casser une méthode cryptographique donnée ?

Objectifs

- Cartographier les méthodes cryptographiques existant en partant de la première fois où une trace de l'usage de la discipline fut trouvée selon la littérature, pour continuer par celles qui se

trouvent être les plus utilisées en notre ère, et clore par celles qui sont candidates aux futurs exigences du domaine ;

- Mettre en exergue les théories mathématiques sur lesquelles se base l'inviolabilité des méthodes recensées selon notre point précédent.

Dans les paragraphes qui suivent, nous allons énumérer les grandes lignes que suivra notre présent ouvrage.

Dans le premier chapitre, nous allons nous forger une solide base sur ce qu'était la cryptographie depuis son apparition jusqu'à la fin de l'ère pré-informatique. Et en parallèle de s'initier à la cryptanalyse des différentes méthodes citées.

Dans le second chapitre, nous nous étalerons sur des notions mathématiques indispensables à la cryptographie. Il sera divisé en deux grandes parties. Celle relative à la théorie de l'information dans un premier temps. La seconde partie en est la théorie des nombres et la mathématique discrète.

Dans le troisième chapitre, nous allons avancer deux sujets importants. D'une part, la menace qu'apporte les ordinateurs quantiques aux cryptosystèmes largement utilisés en ce moment. D'autre part, nous allons développer les basiques de la lattice, outil mathématique le plus promettant sur lequel se base bon nombre de candidats aux cryptosystèmes post-quantiques, ainsi que deux problèmes mathématiques fondamentaux qui lui sont relatifs.

Dans le quatrième chapitre, nous allons introduire l'objet mathématique MLWE (Learning With Error over Modules). Les module lattices peuvent être considérés comme des lattices situés entre ceux utilisés dans les définitions du problème LWE et ceux utilisés pour le problème R-LWE. Le Module-LWE offre un compromis entre les deux extrêmes que constituent le LWE et le R-LWE.

Nos recherches se clôtureront sur la modélisation et l'analyse de la performance du cryptosystème CRYSTALS. Ce dernier est basé sur la technologie mathématique MLWE. Le cryptosystème CRYSTALS se compose, rappelons-le, de deux sous-systèmes : CRYSTALS-Kyber et CRYSTALS-Dilithium. Le premier est un cryptosystème type KEM si le second est de type signature.

CHAPITRE 1
LA CRYPTOLOGIE, OU CRYPTOGRAPHIE ET CRYPTANALYSE, DE L'ERE PRE-INFORMATIQUE

1.1 Introduction

Avant d'entrer dans le vif du sujet, attardons-nous un peu aux quelques terminologies détaillées dans les paragraphes qui suivent, terminologies qui seront utilisées et réutilisées tout au long du présent ouvrage.

Le texte original, celui qui est compréhensible par tous, sera dénommé *texte en clair* (*plaintext* dans nos bibliographies en anglais). [1.01], [1.03]

Texte chiffré (*ciphertext*) quant à lui, désignera le résultat obtenu après que le processus de chiffrement ait été appliqué au texte en clair.

Le processus de chiffrement, tout simplement désigné par *chiffre* (*cipher*) consistera en l'ensemble d'algorithmes à appliquer pour passer du texte en clair au texte chiffré et vice versa.

Une *clé* (*key*) permettra de faire varier l'action de l'algorithme de chiffrement au texte en clair. Le destinataire doit connaitre la valeur de cette clé pour effectuer le processus inverse sur le texte chiffré ainsi obtenu.

Nous utiliserons également le minuscule *a-z* pour les lettres en clair. Et la majuscule *A-Z* pour les lettres chiffrées.

Le chiffre de Julius Caesar appartient à une classe de méthodes cryptographiques nommée chiffrement par *substitution mono-alphabétique*. *Substitution* car une lettre en clair est remplacée par une lettre chiffrée suivant une règle bien définie. Et *mono-alphabétique* car on fait correspondre une seule et unique lettre chiffrée à chacune des lettres en clair.

1.2 Chiffrement par substitution

1.2.1 Chiffrement mono-alphabétique

1.2.1.1 Chiffrement par addition

Ce qui suit désignera directement le modèle mathématique qui se cache dans le chiffrement par

addition. Cette dernière, est la méthode de chiffrement utilisée du temps de *Julius Caesar*, voici 2000 ans avant notre ère. [1.01], [1.03]

a. *Chiffrement*

Désignons par C le texte chiffré et par P le texte en clair. Posons k la clé de chiffrement, qui sera dans le cas du présent chiffrement par addition (et pour le cas des quelques méthodes de chiffrement qui vont suivre d'ailleurs) un nombre entier. [1.01], [1.02]

$$C \equiv P + k \; modulo \; 26 \qquad \textbf{(}1.01\textbf{)}$$

Notons que le nombre 26 n'est autre que le nombre de lettres dans l'alphabet.

Exemple : En prenant $k = 3$, la lettre *a* après chiffrement devient *D* (*a* est équivalent à 1, $1 + 3$ donne 4, ce qui revient à *D*).

x quant à lui équivaut à 24. Si l'on y ajoute 3, cela donne 27. Après un $modulo\ 26$, on revient à 1 donc *A*. *x* après chiffrement donne *A*.

b. *Déchiffrement*

Le déchiffrement se fait par la formule qui suit : [1.01], [1.03]

$$P \equiv C - k \; modulo \; 26 \qquad \textbf{(}1.02\textbf{)}$$

c. *Faiblesse*

Le nombre de caractères dans l'alphabet étant de 26, le nombre de valeurs que peut prendre notre clé de chiffrement k est limité à 26 (une valeur qui inclue la clé $k = 26$ qui revient au fait de n'appliquer aucun chiffrement au message en clair). Cela rend la méthode de chiffrement par addition sensible à l'*attaque par force brute* (*brute force attack*). Cette attaque consiste à essayer un à un les valeurs de clés possibles jusqu'à obtenir un résultat satisfaisant. [1.03]

1.2.1.2 Chiffrement par multiplication

La présente méthode de chiffrement consiste en l'étape qui permettra de passer de la méthode de chiffrement précédente (chiffrement par addition), à celle qui va suivre (chiffrement affine).

a. *Chiffrement*

La formule utilisée revient à ce qui va suivre : [1.01]

$$C \equiv kP \; modulo \; 26 \qquad \textbf{(}1.03\textbf{)}$$

Notons quand même que la valeur que peut prendre k devra suivre certaines conditions. Etant donné que 26 (le nombre de lettres de l'alphabet pour rappel) si décomposé en nombre premiers donne 2×13. Pour que chacune des lettres en clair de l'alphabet ait une et une seule correspondance, k devrait être ni multiple de 2 ni de 13. k peut-être donc parmi $1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23$ et 25. Soit 13 valeurs possibles au total.

Exemple : Pour $k = 3$. a devient C après chiffrement ($3 \times 1 = 3$). x devient T (soit $3 \times 24 \; modulo \; 26 = 20$).

b. *Déchiffrement*

Le déchiffrement se fait logiquement selon la formule ci-dessous : [1.01]

$$P \equiv \frac{1}{k} C \; modulo \; 26 \qquad \textbf{(}1.04\textbf{)}$$

Il y a cependant un léger problème à utiliser cette formule. En effet, prenons le cas où C n'est pas divisible par k (la lettre chiffrée est *A* par exemple, ce qui équivaut à 1 et $\frac{1}{3}$ n'est pas un entier donc difficilement transformable en lettre).

Nous allons faire entrer la notion d'inverse multiplicative modulaire $\bar{k}$ telle que $\frac{1}{k} C \; modulo \; 26 =$

$\bar{k}C\ modulo\ 26$. Dans ce cas notre formule de déchiffrement finale est donnée par :

$$P \equiv \bar{k}C\ modulo\ 26 \qquad \textbf{(}1.05\textbf{)}$$

Démonstration :

♣

Soit z le nombre de lettres de l'alphabet dans lequel vont être piquées les lettres de notre texte en clair, de même celles de notre texte chiffré. Dans le cas de l'alphabet latin, que nous sommes en train d'utiliser, $z = 26$.

Nous avons vu plus haut que si z était décomposable en nombre(s) premier(s), alors k ne devrait être multiple d'aucun d'entre eux. En d'autres termes, *k* et *z* doivent être premiers entre eux.

D'après le *théorème de Bachet-Bézout*, les nombres entiers a et b tels que $ak + bz = 1$ existent.

Et comme $bz\ modulo\ z \equiv 0$;

$\frac{1}{k} = a = \bar{k}$;

♦

L'*algorithme d'Euclide étendu*, dont les détails feront partis du chapitre 2 du présent ouvrage, nous permet d'arriver à l'égalité qui suit : $9 \times 3 - 1 \times 26 = 1$, d'où notre $a = 9$ et $b = 1$. Soit $a = \bar{k} = 9$.

Pour en revenir au déchiffrement de notre lettre encodée *A*. *A* équivaut à 1 la lettre en clair correspondant est donc $P = 9 \times 1\ modulo\ 26 = 9$, ce qui revient à la lettre *i*.

c. *Faiblesse*

Etant donné que la présente méthode de chiffrement possède uniquement 12 bonnes clés, elle est également vulnérable à l'attaque par force brute. Mais nous souhaitons introduire une autre forme d'attaque, celle connue sous le nom de l'*analyse de la fréquence des lettres*. Cette technique revient à l'arabe *Abu Yusuf Yaqub ibn Ishaq al-Sabbah al-Kindi*, développé aux alentours du IXème siècle.

Dans toutes les langues, certaines lettres se répètent plus souvent que d'autres. Dans la langue anglaise par exemple, les lettres *e*, *t* et *a* sont celles qui apparaissent les plus fréquemment, respectivement de l'ordre de 12%, 9% et 8%. Pour un message encodé relativement long, nous pourrions ainsi étudier la fréquence d'apparition des lettres, en sortir les 3 premières qui reviennent le plus. Elles correspondront probablement aux lettres en clair *e*, *t* et *a*. Pour les messages courts par contre, cette méthode perd en précision. [1.01]

La question qui se pose est alors, comment rendre notre méthode de cryptographie plus résistante aux attaques ? La solution qui vient tout de suite à l'esprit est pourquoi ne pas chiffrer notre message en clair 2 fois de suite ? Nous allons voir ce qu'il en est de cette idée dans la section qui va suivre.

1.2.1.3 Chiffrement affine

Que se passera-t-il si nous allons chiffrer notre message 2 fois de suite avec la méthode de chiffrement par addition de notre paragraphe 1.1 ? Posons k la clé de chiffrement que nous allons appliquer en premier, puis m la seconde. Nous aurons alors :

$$C \equiv P + k + m \; modulo \; 26$$

Ce qui revient, si nous posons $l = k + m$ à

$$C \equiv P + l \; modulo \; 26$$

Le déchiffrement présente dans ce cas la même difficulté que si nous avons effectué un seul chiffrement avec notre clé initial *k*.

De même, si nous choisissons le chiffrement par multiplication, chiffrer 2 fois de suite avec les clés *k* et *m* revient à chiffrer avec une seule clé $l = k \times m$.

$$C \equiv (k \times m)P \; modulo \; 26 \equiv lP \; modulo \; 26$$

Mais qu'en est-il si nous utilisons dans un premier temps le chiffrement par multiplication, puis dans un second temps le chiffrement par addition ? On obtiendra alors ce que nous appellerons chiffrement affine.

a. *Chiffrement*

Le chiffrement se fait par la formule qui suit : [1.01], [1.03]

$$C \equiv kP + m \; modulo \; 26 \qquad \textbf{(}1.06\textbf{)}$$

Exemple : pour $k = 3$ et $m = 13$, notre lettre en clair *x* deviendra *T* ($3 \times 24 \; modulo \; 26 = 20$) après application du chiffrement par multiplication. *T* deviendra par la suite *G* ($20 + 13 \; modulo \; 26 = 7$) après application de la seconde méthode de chiffrement, le chiffrement par addition.

b. *Déchiffrement*

Le déchiffrement se fait en ordre inversé. Effectuer le déchiffrement par addition, puis faire suivre par le déchiffrement par multiplication : [1.01], [1.03]

$$P \equiv \bar{k}(C - m) \; modulo \; 26 \qquad \textbf{(}1.07\textbf{)}$$

Exemple : G après le déchiffrement par addition devient *T* ($7 - 13 \; modulo \; 26 = 20$). Nous avons vu dans notre paragraphe 1.3.3 que $\bar{3} = 9$. Après déchiffrement par multiplication *T* revient à notre lettre en clair *x* ($9 \times 20 \; modulo \; 26 = 24$).

c. *Atbash Cipher*

Cette méthode de chiffrement fut utilisée par les transcripteurs hébreux lors de la copie du livre Jeremiah de la bible. Elle consiste à inverser l'ordre de l'alphabet lors du processus de chiffrement. En d'autres termes, la lettre en clair *A* devient *Z* après chiffrement, de même *B* devient *Y*, *C* devient *X* et ainsi de suite. [1.01], [1.03]

Alors pourquoi est-ce que nous en parlons ? La réponse en est que ladite méthode est une variante du chiffrement affine. Les valeurs des paramètres k et m dans ce cas sont respectivement -1 et 27.

La formule de chiffrement correspondante s'écrit donc :

$$C \equiv -P + 27\ modulo\ 26 \qquad \textbf{(}1.08\textbf{)}$$

d. *Faiblesse*

Nous avons vu que le nombre de clés possibles pour le chiffrement par multiplication était de 12. Celui pour le chiffrement par addition était de 26. Le nombre de clés possibles pour notre chiffrement affine est donc de $12 \times 26 = 312$. Ce qui est déjà mieux, mais reste quand même peu pour résister à la puissance de calcul disponible en notre ère.

Nous avons parlé dans notre sous-titre 1.3.4 de la faiblesse relative à la fréquence des lettres. Cette faiblesse est toujours d'actualité pour le présent chiffrement affine. Nous avons trouvé beaucoup de solutions à ce problème dans nos bibliographies, à commencer par le chiffrement dit homophonique. [1.03]

1.2.2 Chiffrement homophonique

1.2.2.1 Chiffrement

Le *chiffrement homophonique*, consiste à attribuer plus d'une correspondance en lettre chiffrée à une lettre en clair ayant une fréquence d'apparition élevée. A la lettre en clair *e* par exemple, à laquelle la fréquence d'apparition 12% est attribuée, on en fait correspondre 4 (quatre) lettres chiffrées (*V, @, &* et – par exemple). De sorte que la fréquence de la lettre en clair e retombe à 3% ($3\% = 12\%/4$) lorsqu'elle est chiffrée, sous condition que le choix entre les lettres chiffrées *V, @, &* et – se fasse d'une manière totalement aléatoire. Et comme 3% étant approximativement la fréquence d'apparition de toutes les lettres de l'alphabet, si chacune d'entre elles avait la même probabilité d'apparition ($3\% \simeq 1/26$), nous sommes donc en train d'outrepasser le problème lié à la fréquence d'apparition des lettres. [1.01], [1.02], [1.03]

Mais comment Eve pourrait-elle connaître alors qu'un chiffrement mono-alphabétique ait été utilisé, ou s'il s'agissait plutôt d'un chiffrement homophonique ? Cela fera l'objet du paragraphe qui va suivre, soit le *phi test* ou le calcul de l'*indice de coïncidence*. Cette grandeur (l'indice de coïncidence) permet de distinguer un message chiffré contenant des lettres uniformément distribuées (donc probablement la méthode de chiffrement et du genre chiffrement homophonique),

d'un autre dont la distribution des lettres n'est pas uniforme (donc utilisant un simple chiffrement mono-alphabétique).

1.2.2.2 Phi test ou calcul de l'indice de coïncidence

Le *phi test* a été introduit par *William Friedman* dans les années 1920. Il consiste à calculer l'*indice de coïncidence (IC)* correspondant à un message chiffré. L'indice de coïncidence est la probabilité pour que 2 mêmes lettres apparaissent simultanément dans le message. La formule est comme suit :

$$IC = \sum_{i=1}^{z} (\mathbb{P}(c_i))^2 \qquad \textbf{(}1.09\textbf{)}$$

Où z étant le nombre de lettres de l'alphabet ;

c_i le i-ième lettre de l'alphabet du message chiffré ;

$\mathbb{P}(c_i)$ dénote la probabilité d'apparition de la lettre c_i dans le message chiffré.

a. *Indice de coïncidence pour l'alphabet de A à Z et uniformément distribué*

Nous avons dans ce cas $z = 26$. $\mathbb{P}(c_i) = \frac{1}{z} = \frac{1}{26}\ \forall\ 1 \leq i \leq 26,\ soit\ c_i\ \in \{A, \dots, Z\}$: [1.01]

$$IC_{CUD} = \sum_{i=1}^{26} \left(\frac{1}{26}\right)^2 = 26 \times \frac{1}{26^2} = \frac{1}{26} \simeq 0.038 \qquad \textbf{(}1.10\textbf{)}$$

$\mathrm{IC_{CUD}}$, pour Indice de Coïncidence pour Chiffre Uniformément Distribué.

b. *Indice de coïncidence pour l'alphabet de A à Z dans la langue anglaise*

Notre formule sera équivalente à $\mathrm{IC_{anglais}} = (\mathbb{P}(\mathrm{A}))^2 + (\mathbb{P}(\mathrm{B}))^2 + \cdots + (\mathbb{P}(\mathrm{Z}))^2$, qui revient à la valeur ci-dessous selon la bibliographie [1.01] :

$$IC_{anglais} = 0.066 \qquad \textbf{(}1.11\textbf{)}$$

Ce que l'on peut conclure est que, pour tout message chiffré, on devrait avoir $0.038 \leq IC \leq 0.066$. Si IC est plutôt proche de 0.038, c'est que le chiffrement est probable d'avoir utilisé la méthode homophonique. Si plutôt proche de 0.066, le chiffrement est sûrement un simple chiffrement mono-alphabétique où la fréquence d'apparition des lettres a été répercutée dans le message chiffré.

1.2.3 Chiffrement affine de Hill, un chiffrement type polygraphique

Notons le fait que polygraphique signifie que le processus de chiffrement traite plus d'une lettre en clair à la fois.

La présente méthode de chiffrement fut développée par *Lester Hill* dans les années 1929. Elle consiste à subdiviser le texte en clair en plusieurs blocs de *n* lettre chacun. Et d'appliquer le processus de chiffrement tour à tour sur ces blocs de *n* lettres. [1.01], [1.02], [1.03]

1.2.3.1 Chiffrement

La formule pour le chiffrement est comme ce qui suit : [1.01]

$$C \equiv KP + M \; modulo \; 26 \qquad \textbf{(}1.12\textbf{)}$$

Où :

C et P sont des vecteurs de taille n, dont les éléments sont respectivement les n lettres chiffrées et celles en clair.

$C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$ et $P = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$;

K est une matrice carrée inversible de dimension $n \times n$. Elle représente un genre de facteur multiplicatif comme celui que nous avons vu dans notre paragraphe 1.3.1. Cette fois ci, c'est le déterminant de la matrice K, noté $det(K)$ et le nombre de lettre de l'alphabet 26 qui doivent être

premiers entre eux :

$$K = \begin{pmatrix} k_{1,1} & \dots & k_{1,n} \\ \vdots & \ddots & \vdots \\ k_{n,1} & \dots & k_{n,n} \end{pmatrix};$$

M est également un vecteur de taille n et représente le facteur additionnel qui assure le chiffrement par addition, se référer au paragraphe 1.2.1 :

$$M = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix};$$

Exemple : Prenons $n = 2$, $K = \begin{pmatrix} 3 & 5 \\ 6 & 1 \end{pmatrix}$, et $M = \begin{pmatrix} 13 \\ 9 \end{pmatrix}$. Et essayons de chiffrer le digramme en clair *et*, soit $P = \begin{pmatrix} 5 \\ 20 \end{pmatrix}$:

$C \equiv \begin{pmatrix} 3 & 5 \\ 6 & 1 \end{pmatrix}\begin{pmatrix} 5 \\ 20 \end{pmatrix} + \begin{pmatrix} 13 \\ 9 \end{pmatrix} \; modulo \; 26 = \begin{pmatrix} 128 \\ 59 \end{pmatrix} \; modulo \; 26 = \begin{pmatrix} 24 \\ 7 \end{pmatrix}$, soit le digramme chiffré *XG*.

La même procédure appliquée au digramme en clair *es* nous donne le digramme chiffré *SF*. Remarquons la lettre en clair *e* qui devient après chiffrement *X* pour le cas du digramme *et*. Devient par contre *S* pour le cas du digramme *es*.

Et comme rechercher la fréquence d'apparition de digramme s'avère être une tâche assez ardue. N'imaginons pas le cas où $n = 3, 4$ voire 5… Encore une fois, nous sommes en présence de méthode de chiffrement à robustesse améliorée contre la méthode de décryptage basée sur la fréquence d'apparition des lettres.

1.2.3.2 Déchiffrement

Le déchiffrement s'effectue par la formule qui suit : [1.01]

$$P \equiv \overline{\det(K)}\; {}^t com(K)(C - M) \; modulo \; 26 \qquad \textbf{(}1.13\textbf{)}$$

Où :

$\det(K)$ se réfère au déterminant de la matrice K. $\overline{\det(K)}$ donc représente la multiplicative inverse de $\det(K)$ modulo 26 ;

$com(K)$ représente la comatrice de la matrice K et ${}^{t}com(K)$ se réfère à la transposée de $com(K)$;

Exemple : Déchiffrons le premier digramme *XG* que nous avons obtenu dans notre paragraphe précédent 1.5.2.

$det(K) = (3)(-1)^{1+1}(1) + (6)(-1)^{1+2}(5) = 3 - 30 = -27$;

L'inverse multiplicatif de -27 modulo 26 en suivant la méthode détaillée dans le paragraphe 1.3.3 donne $\overline{det(K)} = -1$.

La matrice comatrice de K est :

$$com(K) = \begin{pmatrix} (-1)^{1+1}(1) & (-1)^{1+2}(6) \\ (-1)^{1+2}(5) & (-1)^{2+2}(3) \end{pmatrix} = \begin{pmatrix} 1 & -6 \\ -5 & 3 \end{pmatrix} ;$$

$${}^{t}com(K) = \begin{pmatrix} 1 & -5 \\ -6 & 3 \end{pmatrix} ;$$

Notre digramme en clair s'obtient donc comme suit :

$$P \equiv (-1)\begin{pmatrix} 1 & -5 \\ -6 & 3 \end{pmatrix}\left(\begin{pmatrix} 24 \\ 7 \end{pmatrix} - \begin{pmatrix} 13 \\ 9 \end{pmatrix}\right) modulo\ 26 = \begin{pmatrix} -27 \\ 72 \end{pmatrix} modulo\ 26 = \begin{pmatrix} 5 \\ 20 \end{pmatrix},$$ soit *ET*.

1.2.3.3 Faiblesse

Nous avons connu jusqu'ici des attaques qui se basent uniquement sur la connaissance du texte chiffré. Un autre cas qui pourrait se présenter est que l'attaquant pourrait être en possession d'une portion de texte en clair également. L'attaquant peut venir à bout du présent chiffrement s'il est en possession d'autant de couple de (lettre en clair, lettre chiffrée) que d'inconnus dans notre équation de chiffrement **(**1.12**)**. [1.01]

Prenons le cas par exemple des 2 digrammes que nous avons chiffrés précédemment (*et*, *XG*) et (*es*, *SF*) auxquels nous allons rajouter le couple de (lettre en clair, lettre chiffrée), également obtenu par

le même processus de chiffrement, (*bi*, *LD*). Nous aurons :

(*et*, *XG*) donne $\begin{pmatrix}24\\7\end{pmatrix} = \begin{pmatrix}k_{1,1} & k_{1,2}\\k_{2,1} & k_{2,2}\end{pmatrix}\begin{pmatrix}5\\20\end{pmatrix} + \begin{pmatrix}m_1\\m_2\end{pmatrix}$;

(*es*, *SF*) donne $\begin{pmatrix}19\\6\end{pmatrix} = \begin{pmatrix}k_{1,1} & k_{1,2}\\k_{2,1} & k_{2,2}\end{pmatrix}\begin{pmatrix}5\\19\end{pmatrix} + \begin{pmatrix}m_1\\m_2\end{pmatrix}$;

Et (*bi*, *LD*) donne $\begin{pmatrix}12\\4\end{pmatrix} = \begin{pmatrix}k_{1,1} & k_{1,2}\\k_{2,1} & k_{2,2}\end{pmatrix}\begin{pmatrix}2\\9\end{pmatrix} + \begin{pmatrix}m_1\\m_2\end{pmatrix}$;

Notre attaquant serait alors en possession d'une sorte de 6 équations à 6 inconnues, à l'issue desquelles il est en mesure de reconstituer les paramètres *K* et *M* du chiffrement affine de Hill utilisé.

$$\begin{cases}24 = 5k_{1,1} + 20k_{1,2} + m_1\ modulo\ 26\ (1)\\ 7 = 5k_{2,1} + 20k_{2,2} + m_2\ modulo\ 26\ (2)\\ 19 = 5k_{1,1} + 19k_{1,2} + m_1\ modulo\ 26\ (3)\\ 6 = 5k_{2,1} + 19k_{2,2} + m_2\ modulo\ 26\ (4)\\ 12 = 2k_{1,1} + 9k_{1,2} + m_1\ modulo\ 26\ (5)\\ 4 = 2k_{2,1} + 9k_{2,2} + m_2\ modulo\ 26\ (6)\end{cases};$$

Démonstration :

♣

$(1) - (3)$ nous donne $k_{1,2} = 5$;

$(2) - (4)$ donne $k_{2,2} = 1$;

En ramenant cette valeur de $k_{1,2}$ dans (1) et (5) nous aurons :

$$\begin{cases}24 = 5k_{1,1} + 100 + m_1\ modulo\ 26\\ 12 = 2k_{1,1} + 45 + m_1\ modulo\ 26\end{cases} \text{ soit } \begin{cases}24 = 5k_{1,1} + 22 + m_1\ modulo\ 26\ (7)\\ 12 = 2k_{1,1} + 19 + m_1\ modulo\ 26\ (8)\end{cases};$$

$(7) - (8)$ donne $12 = 3k_{1,1} + 3$ soit $k_{1,1} = 3$;

En ramenant cette valeur de $k_{1,1}$ dans (7) nous aurons :

$24 = 15 + 22 + m_1\ modulo\ 26 = 37 + m_1\ modulo\ 26 = 11 + m_1$ soit $m_1 = 13$;

En ramenant cette valeur de $k_{2,2}$ dans (4) et (6) nous aurons :

$$\begin{cases} 6 = 5k_{2,1} + 19 + m_2 \ modulo\ 26 \\ 4 = 2k_{2,1} + 9 + m_2 \ modulo\ 26 \end{cases} \text{ soit } \begin{cases} -13 = 5k_{2,1} + m_2 \ modulo\ 26\ (9) \\ -5 = 2k_{2,1} + m_2 \ modulo\ 26\ (10) \end{cases}$$

En rajoutant 26 dans notre égalité (9). Cela ne change rien en effet car nous sommes en $modulo\ 26$:

$$\begin{cases} 13 = 5k_{2,1} + m_2 \ modulo\ 26\ (11) \\ -5 = 2k_{2,1} + m_2 \ modulo\ 26\ (10) \end{cases};$$

$(11) - (10)$ donne $18 = 3k_{2,1}$ soit $k_{2,1} = 6$;

En ramenant cette valeur de $k_{2,1}$ dans (11) :

$13 = 30 + m_2 \ modulo\ 26 = 4 + m_2$ soit $m_2 = 9$;

Notre attaquant a pu donc reconstruire les paramètres qui composent notre chiffrement affine de Hill : $K = \begin{pmatrix} 3 & 5 \\ 6 & 1 \end{pmatrix}$, et $M = \begin{pmatrix} 13 \\ 9 \end{pmatrix}$;

♦

Une autre idée qui vient en tête pour passer outre le problème cité plus haut et de faire varier les paramètres de la méthode de chiffrement à appliquer au fur et à mesure où l'on change de lettre en clair à chiffrer. Astuce qui trouvera son application dans le paragraphe 1.5 suivant.

1.2.4 Chiffrement poly-alphabétique

Un chiffre est dit *poly-alphabétique* lorsqu'il fait correspondre plus d'une lettre chiffrée à chacune des lettres en clair existantes.

Remarquons donc que le chiffrement homophonique que nous avons vu un peu plus haut est un passage du chiffrement mono-alphabétique au chiffrement poly-alphabétique. Mono-alphabétique pour les lettres à faible fréquence d'apparition, et poly-alphabétique pour celles à fréquence d'apparition élevée.

1.2.4.1 Chiffre d'Alberti

La présente méthode de chiffrement fut inventée et décrite par *Leon Battista Alberti* durant le 15ème

siècle (vers 1466/67).

a. *Présentation*

Pour chiffrer un texte en clair, Alberti introduisit un outil nommé *Disque d'Alberti*. Il consistait en 2 cercles concentriques. [1.01]

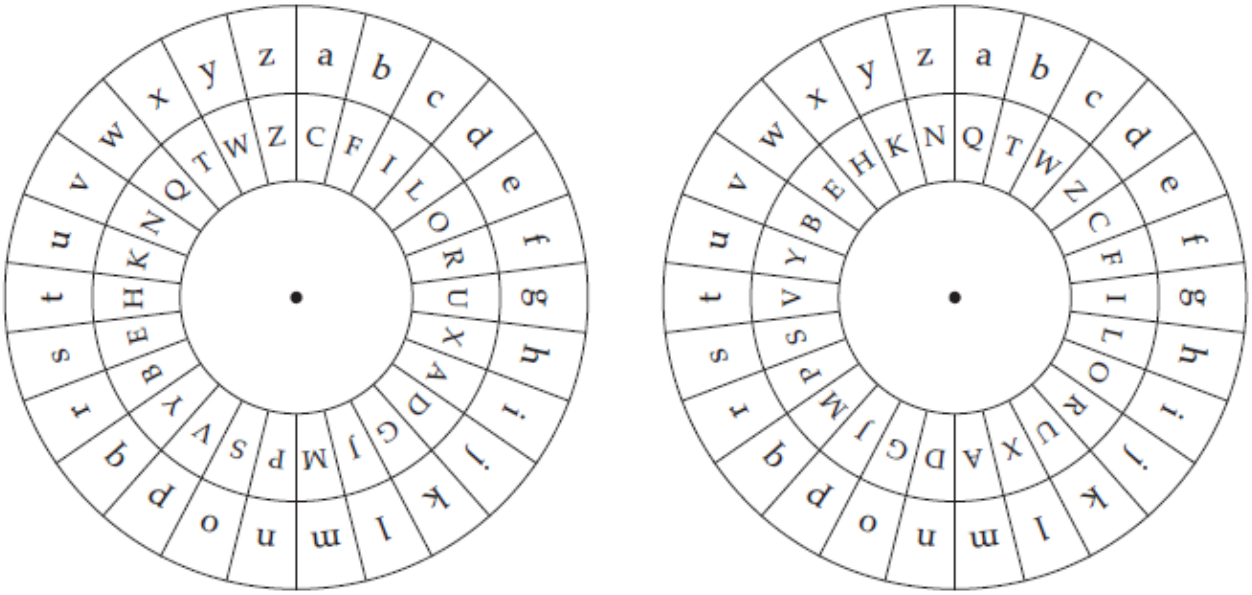


Figure 1.01 : *Disque d'Alberti avec dans un premier temps* $\boldsymbol{m_1 = 0}$ *à gauche, puis* $\boldsymbol{m_2 = -4\ modulo\ 26 = 22}$ *s à droite.*

La partie extérieure est inamovible et contient les lettres en clair de l'alphabet, ordonnées normalement (de *a* à *z*).

La partie intérieure, quant à elle est amovible. Contient les lettres chiffrées ordonnées d'une manière aléatoire, préalablement convenues par l'envoyeur Alice et le récepteur Bob. Notons en passant qu'il y a $26! \simeq 4{,}03 \times 10^{26}$ manières de faire permuter les lettres *a* à *z*, ce qui constitue déjà une première plage de clés non négligeable pour le chiffre d'Alberti. Nous pourrions noter $f(k, P)$ la fonction qui fait cette correspondance, où la clé k prend les valeurs de 1 à 26!.

Pour simplifier, dans ce qui va suivre, nous allons considérer que $f(k, P) = kP\ modulo\ 26$, soit une permutation obtenue par un chiffrement multiplicatif (paragraphe 1.3). Une dernière simplification où $k = 3$ nous amène aux lettres chiffrées *C* à *Z* de la partie interne de notre disque.

b. *Chiffrement*

L'algorithme en utilisant le disque d'Alberti se décrit comme suit : [1.01]

- La première lettre qu'Alice envoie à Bob représente la lettre de la partie externe du disque où la première lettre de la partie interne du disque doit se trouver. Supposons que ça soit *a*. Alice et Bob mettent *C* en dessous de *a* donc (partie gauche de la **Figure 1.01**, $m_1 = 0$).

- Maintenant Alice peut envoyer à Bob les lettres/mots à crypter. Soient *as* ce qu'Alice veut envoyer. Les lettres chiffrées qui correspondent sont *CE* d'après le disque. Le texte chiffré que nous avons jusqu'ici est donc : *aCE*.

- Alberti a proposé qu'après 3 ou 4 mots, Alice choisit une nouvelle clé. Soit *e* cette nouvelle clé (partie droite de la **Figure 1.01**, $m_2 = 22$), et qu'Alice veut par la suite chiffrer *ont*. Le texte chiffré correspondant est *GDV*. Le texte chiffré final devient donc : *aCEeGDV*.

Quant à sa modélisation mathématique, le chiffre d'Alberti est une combinaison d'un chiffre de permutation de clé k, que nous avons noté $f(k, P)$, et d'un chiffre d'addition de clé m_i. La clé du chiffre d'addition, comme nous avons vue plus haut, a été proposée à être changée tous les 3 à 4 mots. D'où l'indice i dans m_i.

$$C \equiv f(k, (P + m_i))\ modulo\ 26 \qquad \textbf{(}1.14\textbf{)}$$

La simplification que nous avons proposée plus haut également $f(k, P) = kP\ modulo\ 26$, nous amène au genre de chiffre affine du paragraphe 1.4 :

$$C \equiv k(P + m_i)\ modulo\ 26 \qquad \textbf{(}1.15\textbf{)}$$

Exemple : Nous allons chiffrer la lettre a de notre premier syllabe *as* : $a \equiv 1, C \equiv 3 \times (1 + 0)\ modulo\ 26 \equiv 3 \equiv C$.

Chiffrons maintenant la première lettre *o* de notre seconde syllabe plus haut *ont*. $o \equiv 15, C \equiv 3 \times (15 + 22)\ modulo\ 26 \equiv 111\ modulo\ 26 \equiv G$.

c. *Faiblesse et/ou avantage*

Nos bibliographies s'accordent au fait que le chiffre d'Alberti avait une certaine sécurité pour son temps. Moyennant le fait qu'Alice devrait changer de disque de temps à autre. [1.01], [1.03]

Mettons en exergue le fait que même si la clé du chiffre par addition m_i est envoyée en clair dans le message chiffré, le message chiffré est en réalité une combinaison d'un chiffre multiplicatif kP et d'un chiffre additif avec la pseudo-clé, cette fois ci inconnue d'Eve, $km_i = l_i$:

$$C \equiv kP + km_i \; modulo \; 26 \equiv kP + l_i \; modulo \; 26 \qquad \textbf{(}1.16\textbf{)}$$

1.2.4.2 Du Tabula Recta au chiffre de Vigenère

a. *Tabula Recta*

Le chiffrement utilisant le *Tabula Recta* (**Figure 1.02**) ou *tableau dans le sens normal* (par opposition à sens inverse) fut introduit par *Johannes Trithemius* dans les années 1508. [1.01], [1.03]

La méthode de chiffrement est dite sans clé car elle consistait à :

- Voir dans l'en-tête du tableau la *première* lettre en clair à chiffrer ;

- Utiliser la lettre juste en dessous, dans la *première* ligne du tableau, comme lettre chiffrée correspondante ;

- Voir dans l'en-tête du tableau la *seconde* lettre en clair à chiffrer ;

- Utiliser la lettre juste en dessous, dans la *seconde* ligne du tableau, comme lettre chiffrée correspondante ;

- Et ainsi de suite.

En d'autres termes la modélisation mathématique est comme suit :

$$c_i \equiv p_i + i \; modulo \; 26 \qquad \textbf{(}1.17\textbf{)}$$

Où c_i étant la i-ième lettre chiffrée, p_i la i-ième lettre en clair et i étant le numéro d'itération en commençant par $i = 1$.

Il va de soi qu'une certaine Eve qui connait que la méthode de chiffrement utilisée par Alice et Bob se trouve être un simple Tabula Recta sans clé, déchiffrera sans problème le texte chiffré. Ce qui nous amène à l'amélioration proposée par *Vigenère*, celle d'introduire une notion de clé dans le présent mode de chiffrement.

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

Figure 1.02 : *Tabula Recta par Johannes Trithemius et amélioré par Vigenère.*

Exemple : Si nous voulons chiffrer la syllabe *as*, cela deviendra *BU*.

b. *Chiffre de Vigenère*

Vigenère a suggéré qu'Alice et Bob devrait s'entendre sur un mot ou une phrase à définir comme clé à utiliser dans leur chiffrement faisant introduire le Tabula Recta. Dans ce cas, le chiffrement

s'effectue comme suit : [1.01], [1.02], [1.03]

- Voir dans l'en-tête horizontal du tableau la *première* lettre en clair à chiffrer ;

- Voir dans l'en-tête vertical du tableau la *première* lettre du mot clé ;

- La lettre chiffrée correspondante est celle qui se trouve à l'intersection des deux dans le tableau ;

- Et ainsi de suite. Lorsque l'on arrive à la dernière lettre du mot clé, revenir au premier.

La modélisation mathématique est donc comme suit :

$$c_i \equiv p_i + m_i \; modulo \; 26 \qquad \textbf{(}1.18\textbf{)}$$

m_i étant la i-ième lettre du mot de passe, revenir à la première lettre à chaque fois que toutes les lettres aient été utilisées.

Exemple : Chiffrons la syllabe *ont* par le chiffre de Vigenère, en utilisant comme clé *as*.

- La lettre en clair *o* avec la lettre clé *a* donnent *P* si directement lu sur le Tabula Recta. Si calculé $c \equiv 15 \; (soit \; o) + 1(soit \; a) \; modulo \; 26 \; \equiv 16 \; modulo \; 26 \equiv 16 \equiv P$;

- *n* et *s* donnent *G* ;

- *t* et *a* (nous étions arrivés à la dernière lettre *s* de la clé et revenons à la première, d'où *a*) donnent *U*.

c. *Faiblesse*

Tout chiffrement poly-alphabétique se base sur l'usage d'un mot ou d'une phrase clé, préalablement convenu entre Alice et Bob. Cette clé est de longueur inférieure à celui du message, de sorte qu'elle se trouve être utilisée à maintes reprises avant que l'intégralité du message soit chiffré. La longueur de cette clé représente la *période*, représentée par l, de la méthode de chiffrement.

Une variante de la formule pour le calcul de l'indice de coïncidence que nous avons vu en paragraphe 1.2.2.2 nous permettra de calculer l pour un message chiffré donné. [1.01]

d. *Indice de coïncidence pour un chiffre à clé périodique*

Indice de Coïncidence pour un Chiffre à Clé Périodique ou IC_{CCP}, est donné par la formule qui suit : [1.01]

$$IC_{CCP} = \frac{n - \frac{n}{l}}{n-1} \times IC_{CUD} + \frac{\frac{n}{l} - 1}{n-1} \times IC_{anglais} = \frac{n - \frac{n}{l}}{n-1} \times 0.038 + \frac{\frac{n}{l} - 1}{n-1} \times 0.066 \qquad \textbf{(}1.19\textbf{)}$$

Où n représente la longueur du message ;

l la longueur de la clé utilisée ;

IC_{CUD} et $IC_{anglais}$ respectivement l'indice de coïncidence d'un chiffre uniformément distribué et celui de la langue anglaise ;

Remarquons que $IC_{CCP} = IC_{CUD} = 0.038$ pour $l = n$, en d'autres termes toutes les lettres du message chiffré ont été chiffrées différemment. Et $IC_{CCP} = IC_{anglais} = 0.066$ pour $l = 1$, en d'autres termes pour un chiffrement mono-alphabétique.

Démonstration :

♣

Notre message comporte n lettres. Il y a n façons de tirer au hasard une lettre, puis $n - 1$ de tirer la seconde. Soit $n(n-1)$ façons de tirer 2 lettres.

Maintenant, subdivisons notre message chiffré en l colonnes, de sorte que toutes les lettres dans la même colonne aient été chiffrées par la même lettre clé. On a dans chaque colonne approximativement n/l lettres donc. Il y a deux possibilités : soit les 2 lettres que nous tirons appartiennent à deux colonnes différentes et nous tombons sur un IC_{CUD}, soit elles appartiennent à la même colonne et nous avons un $IC_{anglais}$.

Pour le premier cas, il y a toujours n façons de tirer notre première lettre et $n - n/l$ façons pour que notre 2ème lettre appartienne aux $l - 1$ colonnes restantes, qui rappelons-le, contiennent chacune n/l lettres. La probabilité pour que l'on tombe sur ce premier cas est donc : $\frac{n(n-\frac{n}{l})}{n(n-1)} \times IC_{CUD} = \frac{n-\frac{n}{l}}{(n-1)} \times$

IC_{CUD}.

Pour le second cas, il y a toujours n façons de tirer notre première lettre et $n/l - 1$ façons pour que notre 2ème lettre appartienne à la même colonne. La probabilité pour le second cas est : $\frac{n(\frac{n}{l}-1)}{n(n-1)} \times IC_{anglais} = \frac{\frac{n}{l}-1}{(n-1)}$.

D'où la probabilité totale résultant de la somme des 2 cas possibles :

$IC_{CCP} = \frac{n-\frac{n}{l}}{n-1} \times IC_{CUD} + \frac{\frac{n}{l}-1}{n-1} \times IC_{anglais}$;

♦

Exemple : Considérons le texte chiffré du ***Tableau 1.01***. Nous pouvons avoir le nombre de lettres du message $n = 235$. Le **Tableau 1.02** par contre nous indique que pour notre message chiffré à clé périodique, $IC_{CCP} = 0.044$. De la formule **(**1.19**)**, nous obtenons un genre d'équation à une inconnue.

Tableau 1.01 : *Exemple de message chiffré ayant une période égale à 5.*

HXJVX	*DMTUX*	*NUOGB*	*USUHZ*	*LFWXK*	*FFJKX*
KAGLB	*AFJGZ*	*IKIXK*	*ZUTMX*	*YAOMA*	*LNBGD*
HZEHY	*OMWBG*	*NZPMA*	*PZHMH*	*KAPGV*	*LASMP*
POFLA	*LTBWI*	*LQQXW*	*PZUHM*	*OQCHH*	*RTFKL*
PEUXK	*DMTKX*	*HPJGZ*	*IGUBM*	*OMEGH*	*WUDMN*
YQTHK	*JAOOX*	*YEBMB*	*VZTBG*	*PFBGW*	*DTBMB*
ZFIXN	*ZQPYT*	*IAPDM*	*OAVZA*	*AMMBV*	*LIJMA*
VGUIB	*JFVKX*	*ZASVH*	*UHFKL*	*HFJHG*	

Tableau 1.02 : *Quelques caractéristiques du message chiffré du Tableau 1.01.*

Lettre chiffrée	**Nombre d'apparition**	**Probabilité d'apparition**	**Probabilité de coïncidence**
A	15	0.064	0.004
B	14	0.060	0.003
C	1	0.004	0.000
D	6	0.026	0.001
E	4	0.017	0.000
F	11	0.047	0.002

G	13	0.055	0.003
H	16	0.068	0.004
I	8	0.034	0.001
J	8	0.034	0.001
K	12	0.051	0.002
L	10	0.043	0.002
M	18	0.077	0.006
N	5	0.021	0.000
O	9	0.038	0.001
P	11	0.047	0.002
Q	5	0.021	0.000
R	1	0.004	0.000
S	3	0.013	0.000
T	9	0.038	0.001
U	11	0.047	0.002
V	8	0.034	0.001
W	6	0.026	0.001
X	13	0.055	0.003
Y	5	0.021	0.000
Z	13	0.055	0.003
Total	235	1.000	0.044

$0.044 = \frac{235-\frac{235}{l}}{235-1} \times 0.038 + \frac{\frac{235}{l}-1}{235-1} \times 0.066$;

Soit après résolution, $l = 4.59$. Rappelons toutefois que c'est une valeur approchée, nous pourrions donc avoir la vraie valeur de l égale à 4 ou 5 avec un peu de chance, et 3 ou 6 aux pires des cas.

Pour ce qui va suivre, supposons qu'Eve a trouvé la période exacte du chiffre qui est de 5, qu'est-ce qu'elle a à faire ensuite pour obtenir le message en clair ?

e. *Superposition et réduction*

Eve a maintenant à grouper chacune des lettres du message chiffré selon qu'elles aient été chiffrées par la 1ère lettre du mot clé, par la 2nde, … jusqu'à la 5ème pour le cas de notre exemple de méthode de chiffrement qui est de période 5. Cette première action est connue sous le terme *superposition*. Se référer au **Tableau 1.03** vu plus bas. [1.01]

On obtient donc une portion de message chiffré par la même lettre clé, en d'autre termes utilisant une méthode de chiffrement mono-alphabétique – voire un chiffre additif – dans le cas du chiffre

de Vigenère. Ce passage du chiffre de Vigenère, chiffre plus ardu à déchiffrer, à un plus simple chiffre qu'est le chiffre additif, se nomme *réduction*.

Eve peut donc vers la fin appliquer soit l'attaque par force brute évoquée dans le paragraphe 1.2.1.1 c, soit l'attaque par l'analyse de fréquence dans le paragraphe 1.2.1.2 c.

Tableau 1.03 : *Le message chiffré du Tableau 1.01 après superposition.*

1ère lettre du mot clé	2nde	3ème	4ème	5ème
H	*X*	*J*	*V*	*X*
D	*M*	*T*	*U*	*X*
N	*U*	*O*	*G*	*B*
U	*S*	*U*	*H*	*Z*
L	*F*	*W*	*X*	*K*
F	*F*	*J*	*K*	*X*
K	*A*	*G*	*L*	*B*
A	*F*	*J*	*G*	*Z*
I	*K*	*I*	*X*	*K*
Z	*U*	*T*	*M*	*X*
Y	*A*	*O*	*M*	*A*
L	*N*	*B*	*G*	*D*
H	*Z*	*E*	*H*	*Y*
O	*M*	*W*	*B*	*G*
N	*Z*	*P*	*M*	*A*
P	*Z*	*H*	*M*	*H*
K	*A*	*P*	*G*	*V*
L	*A*	*S*	*M*	*P*
P	*O*	*F*	*L*	*A*
L	*T*	*B*	*W*	*I*
L	*Q*	*Q*	*X*	*W*
P	*Z*	*U*	*H*	*M*
O	*Q*	*C*	*H*	*H*
R	*T*	*F*	*K*	*L*

(suite)

1ère lettre du mot clé	2nde	3ème	4ème	5ème
P	*E*	*U*	*X*	*K*
D	*M*	*T*	*K*	*X*
H	*P*	*J*	*G*	*Z*
I	*G*	*U*	*B*	*M*
O	*M*	*E*	*G*	*H*
W	*U*	*D*	*M*	*N*
Y	*Q*	*T*	*H*	*K*
J	*A*	*O*	*O*	*X*
Y	*E*	*B*	*M*	*B*
V	*Z*	*T*	*B*	*G*
P	*F*	*B*	*G*	*W*
D	*T*	*B*	*M*	*B*
Z	*F*	*I*	*X*	*N*
Z	*Q*	*P*	*Y*	*T*
I	*A*	*P*	*D*	*M*
O	*A*	*V*	*Z*	*A*
A	*M*	*M*	*B*	*V*
L	*I*	*J*	*M*	*A*
V	*G*	*U*	*I*	*B*
J	*F*	*V*	*K*	*X*
Z	*A*	*S*	*V*	*H*
U	*H*	*F*	*K*	*L*
H	*F*	*J*	*H*	*G*

L'analyse de la fréquence des lettres de la première colonne nous indique que la lettre chiffrée *L* est celle qui revient le plus, elle pourrait correspondre à la lettre en clair *e* donc. Ce qui entraine que la

première lettre du mot clé utilisé est *G*. $7\ (G) \equiv 12(L) - 5\ (e)$.

Quelques itérations de la sorte conduiront au fait que le mot clé utilisé est *GLASS*.

Maintenant que nous avons vu comment déchiffrer un message crypté par le chiffre de Vigenère, la question qui se pose est y aurait-il un moyen pour améliorer la robustesse dudit chiffre contre d'éventuels décryptages effectué par Eve ? Peut-être crypter le message par deux différentes clés ? Quelles conditions les deux clés ainsi utilisées doivent-elles justifier pour que l'effort (le double cryptage) effectué soit récompensé à sa juste valeur ?

f. *Amélioration de la robustesse du chiffre de Vigenère*

Les ouvrages que nous avons lus parlent d'amélioration conséquente de la robustesse du chiffre si utilisé à deux reprises par exemple. Sous conditions que les deux clés utilisées aient une longueur dont les valeurs sont premiers entre elles. [1.01]

Soient l_1 et l_2 respectivement la longueur de la première clé et de la deuxième. Le chiffre de Vigenère faisant intervenir dans un premier temps la première clé de longueur l_1 puis dans un second temps la seconde de longueur l_2 est équivalente à un chiffre de Vigenère unique dont la clé est de longueur l_3 avec :

$$l_3 = \mathrm{PPCM}(l_1, l_2) = \frac{l_1 \times l_2}{\mathrm{PGCD}(l_1, l_2)} \qquad \textbf{(}1.20\textbf{)}$$

PPCM signifie Plus Petit Commun Multiple ;

PGCD, Plus Grand Commun Diviseur ;

On en déduit facilement que le double chiffre (chiffrement avec une clé de longueur l_1, puis une seconde fois avec une autre de longueur l_2) est à son apogée de robustesse (équivalent à un chiffre à long mot clé mais avec effort moindre) lorsque $PGCD(l_1, l_2) = 1$. l_1 et l_2 sont premiers entre eux, en d'autres termes. Dans ce cas, le double chiffre est équivalent à un chiffre unique dont le mot clé

est de longueur l_3 avec :

$$l_3 = l_1 \times l_2 \qquad \textbf{(}1.21\textbf{)}$$

Tableau 1.04 : *Clé résultant de OK et CURIOUSER.*

Clé 1	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*
Equivalent en nombre	15	11	15	11	15	11	15	11	15	11	15	11	15	11	15	11	15	11

Clé 2	*C*	*U*	*R*	*I*	*O*	*U*	*S*	*E*	*R*	*C*	*U*	*R*	*I*	*O*	*U*	*S*	*E*	*R*
Equivalent en nombre	3	21	18	9	15	21	19	5	18	3	21	18	9	15	21	19	5	18

Equivalent en nombre	18	6	7	20	4	6	8	16	7	14	10	3	24	26	10	4	20	3
Clé 3	*R*	*F*	*G*	*T*	*D*	*F*	*H*	*P*	*G*	*N*	*J*	*C*	*X*	*Z*	*J*	*D*	*T*	*C*

Exemple : Considérons les deux mots clés *OK* et puis *CURIOUSER*, de longueurs respectivement 2 et 9. Comme ces deux derniers sont premiers entre eux, la formule **(**1.21**)** dit qu'elles (les deux clés, et par conséquent le chiffre qui en résulte) équivaut à une clé de longueur $18 = 2 \times 9$. Cette clé résultat est *RFGTDFHPGNJCXZJDTC*, le calcul étant explicité dans le **Tableau 1.04**.

1.2.5 Machines cryptographiques

Les groupes de machines cryptographiques que nous allons parler sont ceux qui sont en rapport avec les type de chiffrements que nous avons vus depuis le début de notre paragraphe 1.2 . Nous en énumèrerons 2 principaux : ceux de type pinwheel (roues codeuses) et ceux faisant intervenir les rotors. De telles machines ont vu leur période de gloire au début du XXème siècle jusqu'à son millieu, où ont commencé à surgir les ancêtres des ordinateurs tant répandus de nos jours.

1.2.5.1 Pinwheel

a. *Description*

Les machines cryptographiques de type pinwheel fonctionnent d'une manière uniquement mécanique (contrairement au 2nd type de groupe que nous allons voir un peu plus tard, et pour lequel le fonctionnement est à la fois mécanique et électrique). La création de ce type de machine est

attribuée par la littérature à *Boris Caesar Wilhelm Hagelin* en 1922. Le modèle que nous allons présenter est le C-36, utilisé par les français avant et pendant la deuxième guerre mondiale. L'image correspondante est présentée dans la **Figure 1.03**. [1.01]



Figure 1.03 : *Image réelle du C-36.*

Le C-36 possédait 5 roues, sur chacune d'elles étaient fixées respectivement 25, 23, 21, 19, et 17 pins ou curseurs. Voir la **Figure 1.04**.

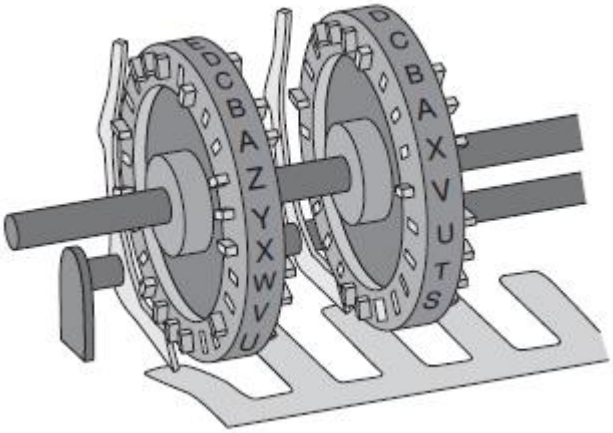


Figure 1.04 : *Schéma modélisé de ses pinwheel ou roues codeuses. A gauche, curseur courant et bras correspondant inactifs. A droite, curseur et bras actifs.*

Il y a un cylindre composé de 25 barres horizontales derrière les 5 roues codeuses. Sur chacune des barres se trouve une sorte d'ergot qui peut se trouver derrière chacune des 5 roues codeuses.

Les curseurs peuvent être inactifs (positionnés à gauche) ou actifs (positionnés à droite). Un curseur inactif conduit au fait que le bras correspondant est également inactif. De même, un curseur actif

entraine un bras inactif. Se référer à la **Figure 1.05**.

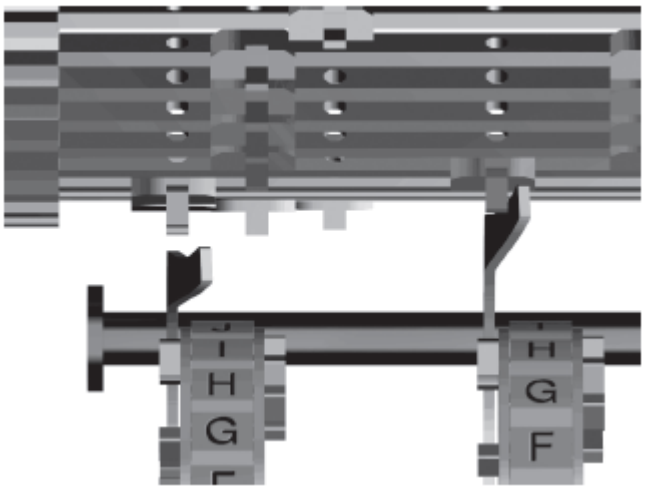


Figure 1.05 : *Schéma modélisé du cylindre à* **25** *barres horizontales et de ses ergots. A gauche, bras inactif. A droite, un ergot d'une barre horizontale rencontrant un bras actif.*

b. *Fonctionnement*

Le chiffrement se fait suivant les étapes ci-dessous : [1.01]

- L'utilisateur paramètre sa première lettre en clair à chiffrer sur un disque prévu par cela ;

- Il fait tourner une manivelle qui fait tourner le cylindre à 25 barres horizontales d'un tour complet ;

- Un tour d'un pas par défaut est appliqué au disque affichant la lettre en clair (et qui va tourner pas à pas pour donner vers la fin la lettre chiffrée). Ceci pour éviter que la lettre chiffrée soit la même que la lettre en clair si tous les curseurs courant des 5 roues codeuses sont inactifs ;

- Pour la j-ième barre (j variant de 1 à 25 donc), si l'ergot correspondant se trouve sur la i-ième roue codeuse (i variant de 1 à 5), et que le curseur courant de cette roue est actif (donc bras correspondant actif également), un tour d'un pas est appliqué au disque affichant la lettre en clair. Pas de tour d'un pas sinon (si le curseur courant est inactif) ;

- Le disque qui affichait la lettre en clair affiche la lettre chiffrée correspondante lorsque le cylindre à 25 barres horizontales a fini de faire le tour complet ;

- Les 5 roues codeuses font toute à leur tour chacun un tour d'un pas ;

- L’utilisateur paramètre sa seconde lettre en clair à chiffrer sur le disque ;

- … ;

c. *Modèle mathématique*

La formule correspondante à l’algorithme que nous avons détaillée dans le paragraphe 1.2.5.1 b est comme suit : [1.01]

$$c_n = 1 + (a_n x_1 + b_n x_2 + c_n x_3 + d_n x_4 + e_n x_5) - p_n \qquad \textbf{(}1.22\textbf{)}$$

Où c_n est la n-ième lettre chiffrée, p_n la n-ième lettre en clair ;

a_n vaut 0 ou 1 selon que le curseur numéro $n\ modulo\ 25$ de la première roue codeuse soit inactif ou actif ;

b_n vaut 0 ou 1 selon que le curseur numéro $n\ modulo\ 23$ de la seconde roue codeuse soit inactif ou actif ;

… ;

x_1 le nombre d’ergots du cylindre à 25 barres se trouvant derrière la première roue codeuse ;

x_2 le nombre d’ergots du cylindre à 25 barres se trouvant derrière la seconde roue codeuse ;

… ;

Exemple : Considérons la configuration de curseurs et d’ergots du **Tableau 1.05**.

Tableau 1.05 : *Exemple de configuration du C-36.*

Numéro barre	**Ergot sur la roue numéro**
1	*1*
2	*2*
3	*2*

	(Numéro roue codeuse)				
Numéro curseur	**1**	**2**	**3**	**4**	**5**
1	*0*	*1*	*0*	*0*	*1*
2	*0*	*1*	*0*	*1*	*1*
3	*0*	*0*	*1*	*0*	*0*

4	*3*
5	*3*
6	*3*
7	*4*
8	*4*
9	*4*
10	*4*
11	*4*
12	*4*
13	*4*
14	*5*
15	*5*
16	*5*
17	*5*
18	*5*
19	*5*
20	*5*
21	*5*
22	*5*
23	*5*
24	*5*
25	*5*

4	*1*	*0*	*0*	*1*	*1*
5	*0*	*0*	*1*	*1*	*0*
6	*1*	*0*	*1*	*0*	*0*
7	*1*	*1*	*0*	*0*	*0*
8	*1*	*1*	*1*	*1*	*1*
9	*0*	*0*	*1*	*1*	*1*
10	*1*	*1*	*0*	*0*	*1*
11	*1*	*0*	*0*	*0*	*0*
12	*1*	*1*	*0*	*0*	*0*
13	*0*	*1*	*1*	*1*	*1*
14	*0*	*1*	*1*	*1*	*0*
15	*1*	*0*	*0*	*0*	*1*
16	*0*	*0*	*1*	*1*	*0*
17	*1*	*0*	*0*	*0*	*1*
18	*0*	*1*	*0*	*0*	
19	*1*	*1*	*1*	*1*	
20	*0*	*1*	*1*		
21	*1*	*1*	*1*		
22	*1*	*0*			
23	*1*	*1*			
24	*1*				
25	*0*				

A partir de la configuration de notre C-36 du **Tableau 1.05**, nous pouvons déduire les valeurs des paramètres (**Tableau 1.06**) de notre formule **(**1.22**)**.

Nous arrivons donc sur les résultats affichés dans le **Tableau 1.07** pour le mot en clair *bork*. Remarquons le fait que même si le mot est répété 3 fois, les mots chiffrés correspondants sont différents.

Tableau 1.06 : *Exemple de valeur des paramètres de notre formule de chiffrement, partant de la configuration du C-36 du Tableau 1.05.*

Numéro lettre à chiffrer n	$a_n x_1$	$b_n x_2$	$c_n x_3$	$d_n x_4$	$e_n x_5$	**Nombre de barres actives modulo 26**	**Formule**
1	*0*	*2*	*0*	*0*	*12*	*14*	*15 -* p_1
2	*0*	*2*	*0*	*7*	*12*	*21*	*22 -* p_2
3	*0*	*0*	*3*	*0*	*0*	*3*	*4 -* p_3

4	*1*	*0*	*0*	*7*	*12*	*20*	*21 - p_4*
5	*0*	*0*	*3*	*7*	*0*	*10*	*11 - p_5*
6	*1*	*0*	*3*	*0*	*0*	*4*	*5 - p_6*
7	*1*	*2*	*0*	*0*	*0*	*3*	*4 - p_7*
8	*1*	*2*	*3*	*7*	*12*	*25*	*26 - p_8*
9	*0*	*0*	*3*	*7*	*12*	*22*	*23 - p_9*
10	*1*	*2*	*0*	*0*	*12*	*15*	*16 - p_{10}*
11	*1*	*0*	*0*	*0*	*0*	*1*	*2 - p_{11}*
12	*1*	*2*	*0*	*0*	*0*	*3*	*4 - p_{12}*
13	*0*	*2*	*3*	*7*	*12*	*24*	*25 - p_{13}*
14	*0*	*2*	*3*	*7*	*0*	*12*	*13 - p_{14}*
15	*1*	*0*	*0*	*0*	*12*	*13*	*14 - p_{15}*
16	*0*	*0*	*3*	*7*	*0*	*10*	*11 - p_{16}*
17	*1*	*0*	*0*	*0*	*12*	*13*	*14 - p_{17}*
18	*0*	*2*	*0*	*0*	*12*	*14*	*15 - p_{18}*
19	*1*	*2*	*3*	*7*	*12*	*25*	*26 - p_{19}*
20	*0*	*2*	*3*	*0*	*0*	*5*	*6 - p_{20}*
21	*1*	*2*	*3*	*7*	*12*	*25*	*26 - p_{21}*
22	*1*	*0*	*0*	*0*	*0*	*1*	*2 - p_{22}*
23	*1*	*2*	*0*	*7*	*0*	*10*	*11 - p_{23}*
24	*1*	*2*	*3*	*7*	*0*	*13*	*14 - p_{24}*
25	*0*	*2*	*0*	*0*	*12*	*14*	*15 - p_{25}*

Tableau 1.07 : *Résultat de chiffrement obtenu pour le mot en clair bork répété 3 fois.*

Valeur de la clé $k = 1 + a_n x_1 + b_n x_2 + c_n x_3 + d_n x_4 + e_n x_5$:	*15*	*22*	*4*	*21*	*11*	*5*	*4*	*26*	*23*	*16*	*2*	*4*
Lettre en clair :	*b*	*o*	*r*	*k*	*b*	*o*	*r*	*k*	*b*	*o*	*r*	*k*
Lettre en clair en nombre :	*2*	*15*	*18*	*11*	*2*	*15*	*18*	*11*	*2*	*15*	*18*	*11*
Lettre chiffrée en nombre :	*13*	*7*	*12*	*10*	*9*	*16*	*12*	*15*	*21*	*1*	*10*	*19*
Lettre chiffrée :	*M*	*G*	*L*	*J*	*I*	*P*	*L*	*O*	*U*	*A*	*J*	*S*

d. *Cryptanalyse des machine de type pinwheel*

Les machines de type pinwheel sont identiques à un type de chiffrement poly-alphabétique. La valeur de la clé étant, une clé de période 25 (obtenue de la première roue codeuse) : [1.01]

a_1x_1, a_2x_1, …, $a_{25}x_1$;

Suivie par une autre, de période 23 :

b_1x_2, b_2x_2, …, $b_{23}x_2$;

… ;

La cryptanalyse qui s'avère être la plus simple étant celle dans le cas où une portion du message en clair correspondant à un message chiffré est connu. Notre formule **(**1.22**)** est équivalente à une autre plus simple, **(**1.23**)**. On peut remarquer qu'il est donc possible d'extraire quelques valeurs correctes correspondant à k_n. La cryptanalyse peut se dérouler suivant les deux étapes qui suivent :

- La valeur possible des 5 curseurs actifs étant de $2^5 = 32$ possibilités, dans le meilleur des cas (la distribution de ces valeurs étant idéalement aléatoire) la chance de retrouver un curseur actif et les 4 curseurs restants inactifs, pour toutes les 5 roues codeuses (soit 10000, 01000, 00100, 00010, 00001) est de une chacun si l'on possède 32 couples de (lettre chiffrée, lettre en clair) donnés. Ces configurations de curseurs courantes permettent de dégager directement la valeur des nombres d'ergots derrière chacune des roues.

$$c_n \equiv k_n - p_n \qquad \textbf{(}1.23\textbf{)}$$

- Comme 25 étant le nombre maximal de curseurs (celui de la première roue codeuse), et que chacune des roues codeuses fasse exactement 1 tour de pas après chaque codage de lettre en clair. Toutes les 5 roues codeuses ont donc chacune fait au moins un tour complet après chiffrement de 25 lettres en clair. La valeur de tous les curseurs peut donc être déduite. Ce qui permet au final de calculer la valeur de k_n quel que soit n.

La cryptanalyse la plus courante est par contre basée sur la connaissance du message chiffré uniquement. Ce second cas est plus ardu à mettre en œuvre, même si relativement simple pour le cas du machine pinwheel de type C-36 (nous faisons allusions à la version améliorée du C-36 qu'est le M-209, pour laquelle la position des ergots est paramétrable). Nous pourrions procéder comme suit :

- En considérant uniquement la roue codeuse numéro 1 (celle ayant 25 curseurs). Nous pouvons donc faire une superposition de 25 colonnes de lettres chiffrées. Dans la première colonne donc la 1ère lettre sur la première ligne, puis la 26ème sur la seconde, la 51ème sur la 3ème, … ;

- Pour les 4 curseurs courants des 4 roues codeuses restantes, nous avons $2^4 = 16$ possibilités. Il y a donc $\frac{1}{16}$ de chance que les lettres chiffrées le soient uniquement par la roue codeuse numéro 1. En d'autres termes, les 4 curseurs des 4 roues codeuses restantes sont simultanément inactifs. Nous allons supposer que nous soyons dans ce cas donc, puis déchiffrer les lettres chiffrées de la première colonne avec les valeurs de curseurs dans $(0, 1)$ et les nombres d'ergots correspondants à la roue codeuse numéro 1 dans $(1, 2, 3, \dots, 13)$. 13 étant le nombre maximal conseillé d'ergots dans le manuel de machine pinwheel de la famille du c-36. Il y a donc exactement 14 essais à faire ;

- Calculer par la suite l'indice de coïncidence (IC) des probables lettres en clair ainsi obtenues et retenir le couple de valeur (curseur, ergot) qui donne un IC élevé. Pour rappel, un IC élevé est synonyme de message en clair (à distribution non uniforme), et un IC à basse valeur est synonyme de message chiffré ou message à lettres aléatoires ;

- Revenir à l'étape numéro 2 et considérer maintenant la 2ème colonne de lettres chiffrées pour déduire la valeur du second curseur sur la roue codeuse. Le nombre d'ergots de la roue codeuse numéro 1 étant déjà obtenu dans l'étape précédente, et utilisé à partir de cette étape. Il y a donc exactement 2 essais à faire (curseur actif ou inactif) ;

- Procéder ainsi pour tous les curseurs de la roue codeuse numéro 1, puis passer aux prochaines roues/curseurs, tout en prenant soin d'utiliser les valeurs trouvées dans les précédentes étapes.

Il va de soi que la cryptanalyse des machines du type C-36 est faisable, mais n'est pas une tâche simple. Surtout pour son époque où l'assistance par ordinateur n'avait pas encore lieu. De plus, comme déjà dit plus haut, la machine c-36 était sujette à des améliorations, notamment le M-209 qui a également connu sa part de succès. Le M-209 possédait 6 roues codeuses à la place des 5 roues du C-36, et des ergots paramétrables selon les envies, comparés aux ergots figés du C-36. Le nombre de barres horizontales du M-209 passe également de 25 (pour le C-36) à 27.

1.2.5.2 Rotor

Les machines à rotors sont le second type de machine cryptographique que nous allons étaler dans le présent ouvrage. Comme déjà mentionné dans le paragraphe 1.2.5.1 , ces machines, en plus d'être en fonctionnement mécanique, sont également à assistance électrique. Les deux principaux

inventeurs, selon notre bibliographie, en sont *Theo A. van Hengel* et *R.P.C. Spengler*.

a. *Description*

Le principe d'un rotor est qu'il consiste en un disque composé des lettres de l'alphabet a à z (les lettres en clair) d'un côté et A à Z (les lettres chiffrées) de l'autre. Chacune des lettres en clair est reliée à la lettre chiffrée qui la substitue à l'aide d'un fils électrique. Un rotor n'est donc autre qu'un genre de disc d'Alberti. [1.01], [1.03]

La configuration d'un rotor est telle qu'elle peut correspondre à n'importe quel chiffre de substitution mono-alphabétique. Mais encore une fois, dans ce qui va suivre, nous allons considérer que ce chiffre par substitution soit un chiffre par multiplication, de clé 3. Rappelons que dans un tel cas, la formule du chiffre est de $C \equiv 3P\ modulo\ 26$.



Figure 1.06 : *Une machine Enigma.*

Nous avons vu la correspondance lettres en clair, lettres chiffrées d'un tel chiffre dans la **Figure 1.01**. La **Figure 1.08** en représente donc l'équivalence en termes de rotor. La règle de substitution

d'un rotor diffère du chiffre d'Alberti suite à son fonctionnement qui consiste à faire un tour de pas après le chiffrement d'une lettre en clair. Et être prêt à chiffrer la lettre en clair suivante. Les Figures **Figure 1.06** et **Figure 1.07** représentent une image réelle d'une machine *Enigma*, et d'un de ses rotors. Nous en parlerons plus en détails dans notre paragraphe 1.2.5.2 c.

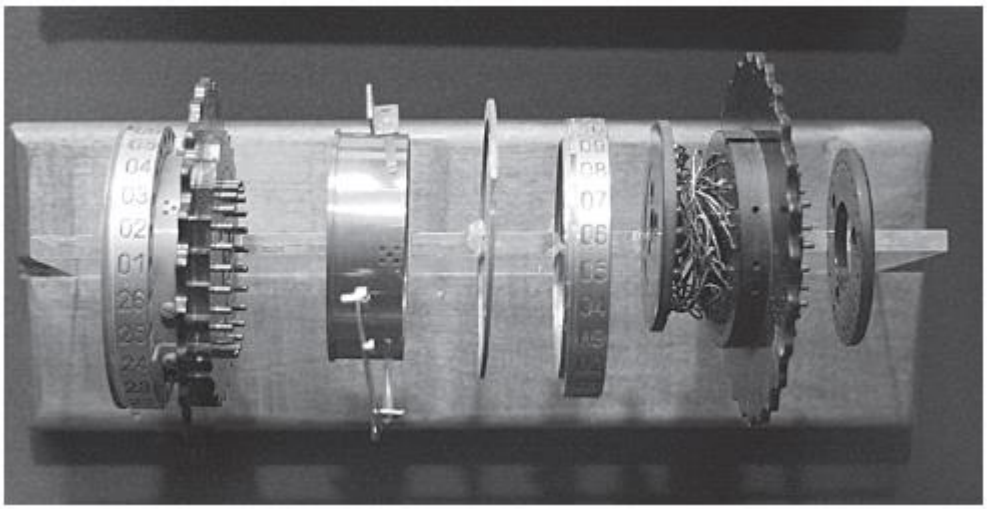


Figure 1.07 : *Image réelle d'un rotor décomposé. Remarquons les fils électriques qui relient l'un à l'autre côté du disque.*

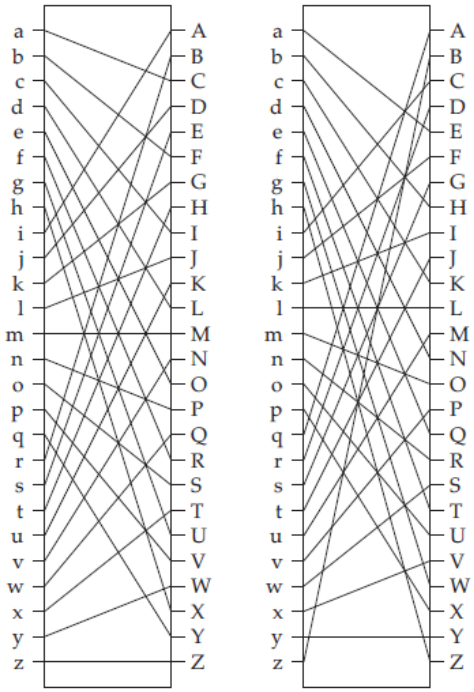


Figure 1.08 : *Représentation des câblages électriques d'un rotor. En position de départ à gauche, puis après un tour de pas à droite.*

b. *Modèle mathématique*

La formule régissant le chiffre performé par un rotor après un tour de pas est comme suit : [1.01]

$$C \equiv 3(P+1) - 1 \; modulo \; 26 \qquad \textbf{(}1.24\textbf{)}$$

Et après *k* tours de pas, nous avons :

$$C \equiv 3(P+k) - k \; modulo \; 26 \qquad \textbf{(}1.25\textbf{)}$$

Nous remarquons tout de suite qu'après 26 tours de pas, le rotor revient à sa position initiale et effectue la même substitution qu'au départ (**Figure 1.09**). Un tel chiffre se trouve être de période 26, et est donc sujet à un fort risque d'être cassé.

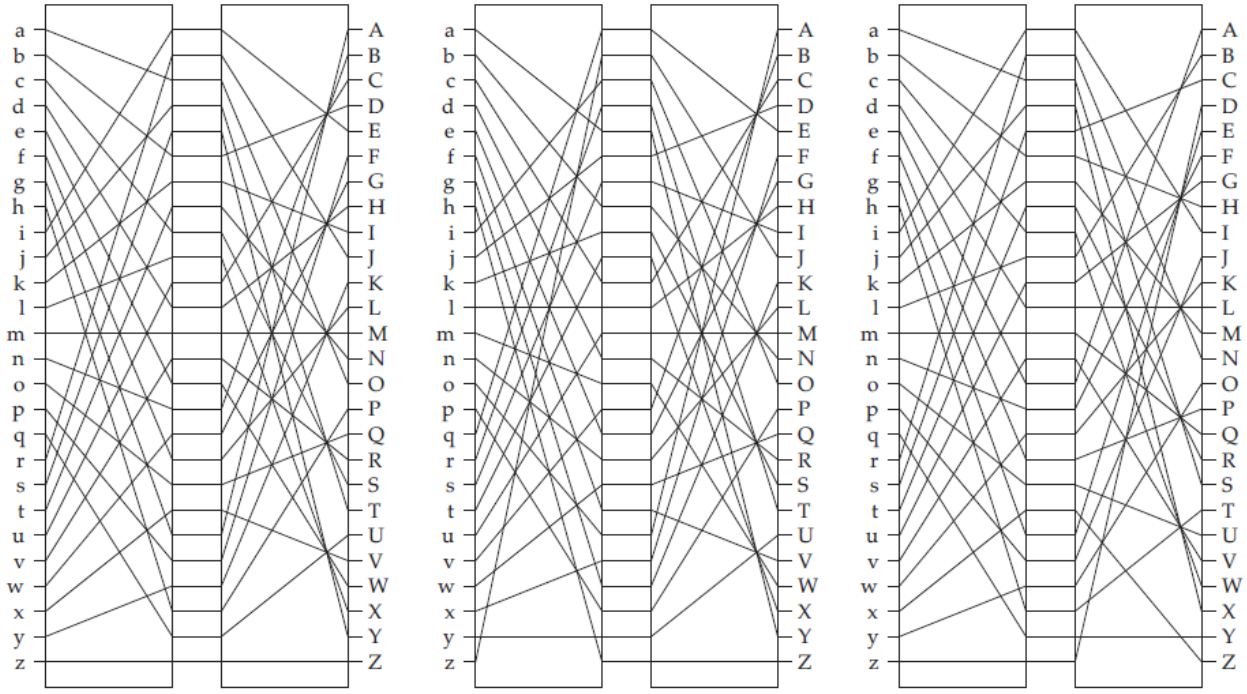


Figure 1.09 : *Deux rotors côte à côte. A gauche, les deux rotors en position initiale pour chiffrer la première lettre en clair. Au milieu, le premier rotor ayant effectué un pas de tour, prêt à chiffrer la seconde lettre en clair. A droite, le premier rotor revenu à sa position initiale et le second rotor ayant effectué un pas de tour, prêt à chiffrer la 26ème lettre en clair.*

Les machines à rotors deviennent intéressantes lorsqu'on commence à associer plusieurs rotors pour

effectuer un chiffre à la chaine. Supposons que le second rotor soit un rotor effectuant un chiffre par multiplication de clé 5, soit $C \equiv 5P\ modulo\ 26$. L'association des 2 rotors nous conduit à la formule de chiffrement **(**1.26**)** pour la k-ième lettre en clair.

$$C \equiv 5(3(P+k)-k)+\left\lfloor\frac{k}{26}\right\rfloor)-\left\lfloor\frac{k}{26}\right\rfloor\ modulo\ 26 \qquad \textbf{(}1.26\textbf{)}$$

Où $\left\lfloor\frac{k}{26}\right\rfloor$ dénote l'entier juste en dessous de la division de k par 26. En d'autres termes, le rotor correspondant au chiffre par multiplication de clé 5 effectue un tour de pas lorsque le premier rotor correspondant au chiffre par multiplication de clé 3 ait fait un tour complet.

La présente machine cryptographique composée de 2 rotors a donc $26 \times 26 = 26^2$ comme période. En règle générale, une machine à n rotors a comme période 26^n.

c. *Enigma, un type de machine cryptographique à rotor*

La machine *Enigma* est une machine cryptographique à rotors dont la première version fut inventée vers la fin de la première guerre mondiale par un ingénieur allemand connu sous le nom d'*Arthur Scherbius*. Toujours dans cette première version, la machine possédait : [1.01], [1.02]

- 3 *rotors* qui vont être dans un premier temps traversés, de droite vers la gauche, par le courant électrique représentant la première étape de chiffrement ;

- Un *réflecteur* qui effectue, encore une fois, un chiffre par substitution. Le réflecteur subdivise l'alphabet de 26 lettres en 2 groupes de 13 lettres chacun. Fait correspondre par la suite chacune des lettres du premier groupe, à une autre dans le second groupe. Le réflecteur assure dans un premier temps que chacune des lettres en clair ne correspond pas à une lettre chiffrée égale à elle (la lettre en clair) même. Et dans un second temps, de rendre réciproque le chiffre effectué par la machine Enigma ;

- Le courant électrique après avoir passé dans le réflecteur, ré-entre dans le labyrinthe des 3 rotors cités dans notre premier point, mais cette fois ci d'un ordre inversé de gauche vers la droite ;

- Dans les versions améliorées de la machine, on trouvait un *tableau de connexion* qui une n-ième fois encore, effectue une substitution. Le tableau de connexion permet de définir manuellement

le comportement du chiffre par substitution qui est appliqué. Le chiffre effectué par le tableau de connexion enveloppe (Effectue un chiffrement de la lettre en clair en premier. Passe la lettre ainsi chiffrée à l'ensemble rotors-réflecteur puis rotors. Effectue le déchiffrement) tous les chiffres précédemment effectués dans nos précédents points ;

La machine Enigma est schématisée d'une manière simple dans la **Figure 1.10**.

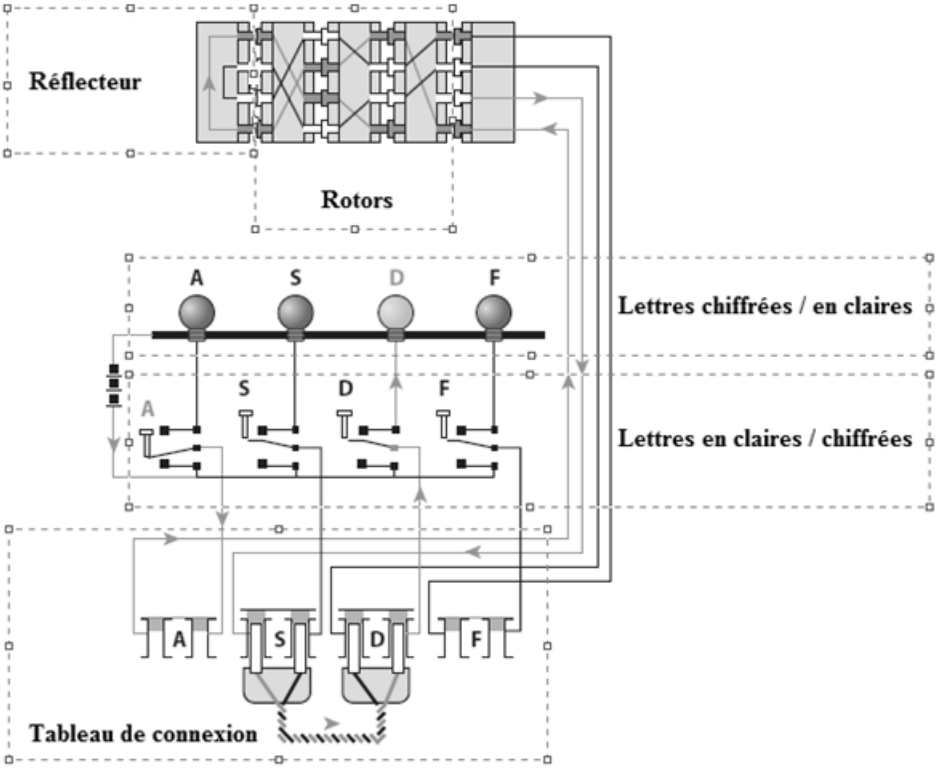


Figure 1.10 : *Le couple de (lettre en clair, lettre chiffré) ou (lettre chiffrée, lettre en clair) (A, D) traité par la machine Enigma.*

La formule du chiffrement effectué par une machine Enigma est représentée comme suit :

$$E \equiv TRMLUL^{-1}M^{-1}R^{-1}T^{-1} \qquad \textbf{(}1.27\textbf{)}$$

Où E représente le chiffrement (Encryption) de la machine Enigma dans sa globalité ;

T et T^{-1} représentent le chiffre du tableau de connexion (Table of connection ou plugboard) et son inverse ;

R et R^{-1} représentent le chiffre du rotor de droite (R pour Right) et son inverse ;

M et M^{-1} représentent le chiffre du rotor du milieu (M pour Middle) et son inverse ;

L et L^{-1} représentent le chiffre du rotor de gauche (L pour Left) et son inverse ;

U représente le chiffre symétrique du réflecteur.

d. *La bombe ou machine d'aide à la cryptanalyse d'Enigma*

La bombe (**Figure 1.11**) est une machine électromécanique utilisée par les cryptologues britanniques afin de casser les codes allemands d'Enigma pendant la Seconde Guerre mondiale. La version la plus connue fut celle conçue par *Alan Turing* en 1939. Celle-ci (la version d'Alan Turing) tire toute fois son idée d'un même genre de machine créé par le cryptologue et mathématicien polonais *Marian Rejewski* un an plus tôt (en 1938). Pour être, un an plus tard (en 1940), améliorée avec la participation cette fois ci de *Gordon Welchman*, un mathématicien anglais. [1.01], [1.04]

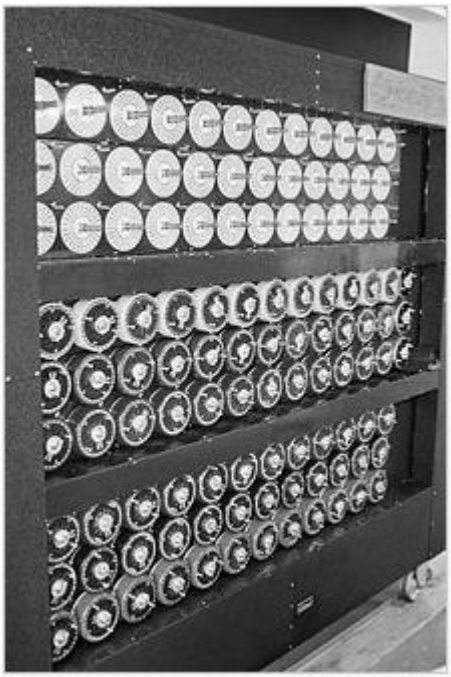


Figure 1.11 : *Une version reconstruite de la bombe qui marche réellement, exposée au musée du parc de Bletchley. La bombe possède* 3 *groupes de* 36 *rotors. Chaque rotor de la bombe simulant l'action d'un rotor de la machine Enigma.*

Il est toutefois à noter que la tâche de la bombe n'était pas de permettre de déchiffrer la configuration exacte de la machine Enigma, mais plutôt de réduire le nombre de configurations possibles à un

nombre acceptable, des configurations qui seront par la suite sujettes à des analyses plus poussées.

Les configurations possibles d'une machine Enigma en général se divisent en 2 groupes. Les configurations internes (celles qui peuvent être changées uniquement en ouvrant la machine) et celles externes (celles qui peuvent se faire machine Enigma fermée).

Les configurations internes sont :

- La sélection des 3 rotors utilisés parmi un ensemble de 5 rotors voire de 8 rotors. On a dans le cas le plus ardu donc $A_3^8 = 8 \times 7 \times 6 = 336$ possibilités où A_k^n représente l'arrangement de k éléments parmi un ensemble de n éléments ;

- Le paramétrage de la bague du rotor (ou *ring setting*). Ce dernier modifie le positionnement du câblage électrique à l'intérieur du rotor. Dans le **Tableau 1.08** par exemple, pour le rotor numéro I s'il n'y a pas de modification, *A* est encodé en *E*. Après une modification du paramétrage de la bague sur la position *B*. *A* devient encodé en *K*. Il y a 26 possibilités pour ce second point.

Tableau 1.08 : *Substitution éffectuée selon le numéro de rotor pour la machine Enigma I.*

Numéro du rotor	ABCDEFGHIJKLMNOPQRSTUVWXYZ	Modèle de la machine à rotor
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Enigma I
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	Enigma I
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	Enigma I
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	M3 Army
V	VZBRGITYUPSDNHLXAWMJQOFECK	M3 Army
VI	JPGVOUMFYQBENHZRDKASXLICTW	M3 & M4 Naval (Fév. 1942)
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT	M3 & M4 Naval (Fév. 1942)
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV	M3 & M4 Naval (Fév. 1942)

Et les configurations externes :

- Les raccords effectués sur le tableau de connexion. 10 couples de lettres peuvent être raccordés. On a donc $\frac{C_2^{26} \times C_2^{24} \times C_2^{22} \times \ldots \times C_2^8}{10!} = 1.5074 \times 10^{14}$ (où C_k^n représente la combinaison de k éléments choisi parmi un ensemble de n éléments) possibilités ;

- La position initiale des rotors. Il y a $26^3 = 17{,}756$ possibilités.

Pour pouvoir fonctionner, la bombe a besoin que le cryptanalyste entre un certain menu obtenu suite à la connaissance d'une portion de texte en clair qui pourrait correspondre à une partie du message chiffré. Dans la pratique, ledit menu consiste en un paramétrage du câblage de la bombe avant que celle-ci (la bombe) ne teste chacune des possibilités de configuration des rotors. Un rotor de la bombe qui a trouvé une bonne configuration (une configuration sans contradiction avec le paramétrage du menu dicté par le supposé couple de texte en clair/texte chiffré) s'arrête de tourner. Le cryptanalyste a droit de noter la position avant de relancer la recherche. Si la position du rotor de la bombe présente une contradiction, elle est ignorée. Le rotor continue de tourner pour tester la position suivante.

1.3 Chiffrement par transposition

Ce que nous avons vu jusqu'ici est que pour une lettre en clair donnée, ou un groupe de lettres en clair donné, on associe une lettre chiffrée, ou groupe de lettres chiffrées. C'est ce qui a été connu sous le terme chiffrement par substitution.

Une autre manière de procéder est de laisser les lettres en clair telles qu'elles sont, mais d'uniquement chambouler leur position, pour donner vers la fin le message chiffré. Une telle méthode est connue sous le nom de chiffrement par transposition.

1.3.1 Scytale



Figure 1.12 : *Une scytale*

La scytale est considéré par nos bibliographies comme étant le chiffre par transposition le plus ancien, utilisé vers l'an -404 par *Lysandre* de Sparte. Elle consistait en un ensemble de parchemin (du cuir dans le cas de l'illustration de la **Figure 1.12**), sur lequel était inscrit le message, et un bâton de diamètre spécifique, sur lequel on enroule le parchemin pour chiffrer (écrire le message en clair

à encoder) et déchiffrer (lire le message chiffré). [1.01], [1.03]

La clé du chiffre s'avère être donc le nombre de lettres en clair qui tient sur la circonférence du bâton. Comme d'habitude, désignons par *k* cette valeur, soit $k = 4$. Nous aurons donc l'arrangement de la **Figure 1.13**. Une lecture horizontale, de haut en bas, du message nous ramène au suivant : *I am hurt very badly HELP*. Si le parchemin est par contre non enroulé sur le bâton, la succession de lettres, qui constitue donc le message chiffré, s'avère être *IryyatbHmvaEhedLurlP*.

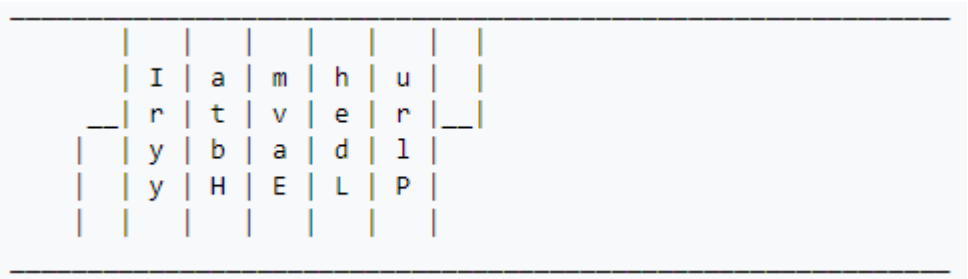


Figure 1.13 : *Une vue modélisée de la scytale.*

1.3.2 Méthodes de chiffrement dérivées du remplissage d'une forme géométrique

1.3.2.1 Idée de base

Le cas du chiffre scytale du paragraphe 1.3.1 peut être interprété comme étant un remplissage vertical d'une forme rectangulaire. Il existe différentes manières de s'y prendre. Le colonel américain *Parker Hitt* en a fait l'objet de son ouvrage sorti durant la première guerre mondiale. Se référer au **Tableau 1.09**. [1.01], [1.03]

Tableau 1.09 : *Quelques manières de remplir une forme rectangulaire.*

(a) Horizontal simple			
ABCDEF	FEDCBA	STUVWX	XWVUTS
GHIJKL	LKJIHG	MNOPQR	RQPONM
MNOPQR	RQPONM	GHIJKL	LKJIHG
STUVWX	XWVUTS	ABCDEF	FEDCBA

(b) Vertical simple			
AEIMQU	DHLPTX	UQMIEA	XTPLHD
BFJNRV	CGKOSW	VRNJFB	WSOKGC
CGKOSW	BFJNRV	WSOKGC	VRNJFB
DHLPTX	AEIMQU	XTPLHD	UQMIEA

(c) Horizontal alterné			
ABCDEF	FEDCBA	XWVUTS	STUVWX
LKJIHG	GHIJKL	MNOPQR	RQPONM
MNOPQR	RQPONM	LKJIHG	GHIJKL
XWVUTS	STUVWX	ABCDEF	FEDCBA

(d) Vertical alterné			
AHIPQX	DELMTU	XQPIHA	UTMLED
BGJORW	CFKNSV	WROJGB	VSNKFC
CFKNSV	BGJORW	VSNKFC	WROJGB
DELMTU	AHIPQX	UTMLED	XQPIHA

1.3.2.2 Chiffre Rail fence

Sur la base d'idées de Parker Hitt, *Friedman* a élargi le concept sur d'autres formes géométriques telles que les trapèzes, les triangles, les croix et les zigzags. Ces derniers étant un les chiffres par transposition les plus connus. Ils sont connus sous le nom de chiffre *Rail fence* (**Tableau 1.10**). La profondeur (le nombre des lignes horizontales à faire traverser par le zigzag) la plus courante est de 2 (cette valeur pouvant être interprétée comme étant la clé du chiffre). [1.01], [1.03]

Tableau 1.10 : *Exemple de chiffrement par Rail fence, pour le message en clair the rail splitter for president.*

Message en clair :	t		e		a		l		p		i		t		r		o		p		e		i		e		t
		h		r		i		s		l		t		e		f		r		r		s		d		n	

Message chiffré :	TEALP ITROP EIETH RISLT EFRRS DN

La manière de procéder étant de remplir une forme en zigzag avec le message en clair. Le message chiffré s'obtient en extrayant le message obtenu d'une lecture horizontale.

1.3.3 Chiffre par permutation

1.3.3.1 Définition et fonctionnement

Le chiffre par permutation consiste à assigner une autre position à toutes les lettres en clair données du message. Pour la praticité, le message est subdivisé en un nombre donné de lettres, puis la permutation est effectuée sur chaque groupe de lettres. Encore pour la praticité, la longueur de subdivision du message en clair ainsi que l'ordre de permutation proviennent d'un mot que l'on pourrait désigner comme clé du chiffre. [1.01]

Dans le **Tableau 1.11**, le mot clé *TALE* a été utilisé. *4321* étant respectivement l'ordre d'apparition des lettres *T*, *A*, *L* et *E* selon leur ordre d'apparition dans l'alphabet normal de *A* à *Z*. Chaque groupe de lettre chiffrée est donc obtenu en triant suivant l'ordre croissant le numéro ainsi attribué au groupe de lettre en clair. Evidemment, le message chiffré final est groupé suivant un nombre qui diffère de la longueur de la clé utilisé, pour rendre un peu difficile la tâche à Eve.

Tableau 1.11 : *Exemple de chiffre par permutation en utilisant le mot clé TALE.*

	4132	4132	4132	4132	4132	4132	4132	4132	4132	4132
Mot clé :	tale	tale	tale	tale	tale	tale	tale	tale	tale	tale
Message en clair :	theb	attl	eand	thes	word	thep	aper	andt	hepe	nllu
Message chiffré :	HBET	TLTA	ADNE	HSET	ODRW	HPET	PREA	NTDA	EEPH	LULN

Message chiffré réarrangé par bloc de 5 lettres :	HBETT LTAAD NEHSE TODRW HPETP REANT DAEEP HLULN

1.3.3.2 Modélisation mathématique

Le chiffrement est représenté sous la forme d'une matrice à 2 lignes et l colonnes, l étant la longueur du mot clé utilisé. La première ligne représente tout simplement un nombre croissant allant de 1 au nombre de lettres du mot clé (4 dans notre cas). La seconde, la succession de chiffre vu dans le paragraphe 1.3.3.1 . [1.01], [1.02]

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix} \quad \textbf{(}1.28\textbf{)}$$

La première colonne de la matrice dans la formule **(**1.28**)** dicte donc que la première lettre du groupe de lettre du message chiffré est en fait la $4^{\text{ème}}$ lettre du groupe de lettre correspondant dans le message en clair.

Pour obtenir la matrice de déchiffrement, échanger la première et la seconde ligne de la matrice. Ce qui donne $\begin{pmatrix} 4 & 1 & 3 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix}$. Puis trier suivant la première ligne, ce qui revient à la formule **(**1.29**)**.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix} \quad \textbf{(}1.29\textbf{)}$$

La plage de clés (le nombre de clé possible) pour un chiffre par permutation est donnée par la valeur $l!$ (l factorielle) pour une clé de longueur l. Cette valeur est égale à $4! = 24$ pour un chiffre par permutation dont la clé est de longueur 4.

$$l! = \prod_{1 \leq n \leq l} n = l \times (l-1) \times \ldots \times 2 \times 1 \qquad \textbf{(}1.30\textbf{)}$$

1.3.3.3 Renforcement d'un chiffre par permutation

a. *Extension*

Par extension, nous entendons le fait que le message chiffré pourrait être de longueur supérieure au message en clair. On pourrait donc réutiliser une ou plusieurs lettres du message en clair pour former le message chiffré. La formule **(**1.31**)** signifie par exemple que le message en clair est traité par bloc de 5 lettres (il y a 5 positions distinctes dans la seconde ligne de la matrice) et que dans un second temps, on retrouve la 2nde lettre d'un bloc de message en clair donné à la fois à la 2nde et à la 5ème position du bloc de message chiffré correspondant (**Tableau 1.12**). [1.01]

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 2 & 5 & 4 & 2 & 1 \end{pmatrix} \qquad \textbf{(}1.31\textbf{)}$$

L'avantage de cette méthode est qu'elle modifie l'indice de coïncidence du message chiffré généré. Un chiffre par transposition pour rappel, garde la valeur de l'indice de coïncidence du texte en clair.

Tableau 1.12 : *Message en clair et message chiffré utilisant la formule d'encodage* ***(****1.31****)****.*

Message en clair :	westw	ardho

Message chiffré :	SEWTEW	DROHRA

La matrice de décodage est donnée par la formule **(**1.32**)**. Remarquons le 5 ou le 2 dans la (2nde ligne, 2nde colonne) des deux matrices équivalentes, étant donné qu'à ces positions a été répliquée la 2nde lettre du bloc de message en clair.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 5 & 1 & 4 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 2 & 1 & 4 & 3 \end{pmatrix} \qquad \textbf{(}1.32\textbf{)}$$

b. *Produit de chiffre par permutation*

Nos bibliographies ont mis en exergue le bénéfice obtenu en chiffrant 2 fois un message en clair

avec 2 mots clés de tailles différentes voire à tailles premiers entre elles. La formule **(**1.20**)** que nous avons vu un peu loin plus haut, est toujours d'actualité. Deux chiffres par permutation utilisant une clé de longueur $l1$ puis $l2$ est équivalent à un seul chiffre dont la clé $l3$ est donnée par la formule **(**1.20**)**. [1.01]

c. *Combinaison d'un chiffre colonnaire ou columnar cipher avec un chiffre par permutation*

Une telle combinaison est connue sous le nom de chiffre colonnaire à clé ou *keyed columnar cipher* (**Tableau 1.13**). Cette méthode fut proposée par le cryptographe anglais *John Falconer* au cours du XVIIème siècle.

Un avantage du chiffre colonnaire à clé est que contrairement au chiffre par addition, par multiplication, affine ou par permutation – pour lesquels un double chiffrement résulte en un chiffre du même type, même si la clé correspondante est tout de même différente – un double chiffrement revient à un chiffre totalement différent. Une robustesse accrue est également d'actualité si la clé du premier chiffre colonnaire appliquée est différente (en termes de longueur) de celle du second chiffre colonnaire utilisé. [1.01].

Tableau 1.13 *: Exemple de chiffre colonnaire à clé utilisant un chiffre colonnaire de clé* $k = 4$ *(nombre de colonnes du chiffre colonnaire) combiné à un chiffre par permutation de clé* $m \equiv TALE \equiv 4132$. *Rapellons le groupement par* 5 *lettres du message chiffré pour embrouiller Eve.*

	Message en clair	**Message chiffré**	
	4132 TALE		
	theb	HBET	
	attl	TLTA	
	eand	ADNE	
	thes	HSET	
	word	ODRW	
	thep	HPET	
	aper	PREA	
	andt	NTDA	
	hepe	EEPH	
	nllu	LULN	
HTAHO HPNEL BLDSD PRTEU ETNER EEDPL TAETW TAAHN			

1.3.4 Cryptanalyse d'un chiffre de type colonnaire, à permutation ou la combinaison des deux

Dans un premier temps, Eve devrait être sûre sur le fait que le chiffre utilisé est de type transposition. Comme maintes fois énoncé, un chiffre par permutation ne change pas la fréquence d'apparition des lettres dans le message chiffré, son indice de coïncidence reste donc inchangé. Se référer au paragraphe 1.2.2.2 pour ce faire.

Pour le décryptage, on peut procéder suivant les paragraphes qui suivent.

1.3.4.1 Passer de la clé k désignant le nombre de lignes pour le chiffre scytale, à une clé k représentant le nombre de colonnes pour un chiffre colonnaire

Du fait que la connaissance du nombre de lignes est équivalente à la connaissance du nombre de colonnes, nous allons plutôt considérer que la clé k du chiffre est le nombre de colonnes. Effectuer également un déchiffrement par force brute pour k allant de 4 à 20. La clé k étant dans la pratique donnée par un mot clé pour faciliter la mémorisation. [1.01]

1.3.4.2 Proportion voyelles/consonnes pour mieux aider à la détection de la clé k

La proportion voyelles/consonnes dans la langue anglaise que nous allons noter $PVC_{anglais}$ est de 38.1% : [1.01]

$$PVC_{anglais} = 38.1\% \quad \textbf{(}1.33\textbf{)}$$

Cela veut dire qu'en prenant k lettres d'une manière aléatoire au sein d'un message en anglais, il est probable que $0.381 \times k$ soient des voyelles. Cela est loin de dire donc que l'on aurait toujours $0.381 \times k$ voyelles si l'on prenait k lettres. Mais plutôt, si l'on prenait plusieurs fois (soit n ce nombre d'essais, avec $n = 100$ par exemple donc) un échantillon de k lettres, il y aurait en moyenne $0.381 \times k$ voyelles. Une manière de raisonner selon notre bibliographie est donc de raisonner en termes de variance de nombre de voyelles.

En addition, il est également énoncé que cette variance est égale à la probabilité d'avoir une voyelle, que l'on multiplie par la probabilité d'avoir une consonne, que l'on multiplie une dernière fois par le nombre de lettres tirées aléatoirement, soit k.

$$\sum_{n}\left(\frac{v}{v+c}-PVC_{anglais}\times k\right)^{2}\leq PVC_{anglais}\times(1-PVC_{anglais})\times k \qquad \textbf{(}1.34\textbf{)}$$

Où v et c sont respectivement le nombre de voyelles et de consonnes pour chacune des n essais de tirer k lettres.

L'égalité $(=)$ est atteinte lorsque le tirage se fait d'une manière aléatoire sur un message en anglais.

Quant à l'infériorité $(<)$, elle est d'actualité s'il s'agit de k lettres successives issues d'un message en anglais. Pour appuyer ceci, mentionnons qu'il est pratiquement impossible d'avoir $k = 10$ lettres successives issue d'un message en anglais qui ne contient pas de voyelles. Par contre il y a une infime probabilité $(1 - PVC_{anglais})^{10} = 0.008$ de tirer $k = 10$ lettres d'une manière aléatoire au sein d'un message en anglais, qui sont toutes des consonnes.

Une valeur de k correcte est donc donnée par la superposition qui minimise le plus la variance donnée par la partie gauche de la formule **(**1.34**)**.

Tableau 1.14 : *Exemple de message chiffré par un chiffre colonnaire à clé.*

OHIVR	SVAHT	BLRHL	HLBIT	MBETM	NOEIO
ITETK	ROWTN	ATHIG	NSDEN	UPBLN	TSEMA
TADAA	ERARI	AOWSA	YIAPT	NAEOW	BCDRE
WAHMT	GEDER	HFDDT	EAEHA	TEHME	IELBO
HIUSI	EKIUE	UHESL	MTKSE	CREP	

Tableau 1.15 : *Calcul de la variance du nombre de voyelles pour* $k = 6$.

I	**II**	**III**	**IV**	**V**	**VI**	**Nombre voyelles réelle**	**Nombre de voyelles théorique**	**Ecart**	**Variance**
O	M	E	W	E	H	3	2.286	0.714	0.510
H	N	N	S	D	I	1	2.286	−1.286	1.654
I	O	U	A	E	U	6	2.286	3.714	13.794
V	E	P	Y	R	S	1	2.286	−1.286	1.654
R	I	B	I	H	I	3	2.286	0.714	0.510
S	O	L	A	F	E	3	2.286	0.714	0.510
V	I	N	P	D	K	1	2.286	−1.286	1.654

A	T	T	T	D	I	2	2.286	−.286	0.082
H	E	S	N	T	U	2	2.286	−.286	0.082
T	T	E	A	E	E	4	2.286	1.714	2.938
B	K	M	E	A	U	3	2.286	0.714	0.510
L	R	A	O	E	H	3	2.286	0.714	0.510
R	O	T	W	H	E	2	2.286	−.286	0.082
H	W	A	B	A	S	2	2.286	−.286	0.082
L	T	D	C	T	L	0	2.286	−2.286	5.226
H	N	A	D	E	M	2	2.286	−.286	0.082
L	A	A	R	H	T	2	2.286	−.286	0.082
B	T	E	E	M	K	2	2.286	−.286	0.082
I	H	R	W	E	S	2	2.286	−.286	0.082
T	I	A	A	I	E	5	2.286	2.714	7.366
M	G	R	H	E	C	1	2.286	−1.286	1.654
B	N	I	M	L	R	1	2.286	−1.286	1.654
E	S	A	T	B	E	3	2.286	0.714	0.510
T	D	O	G	O	P	2	2.286	−.286	0.082
Variance sur la totalité du message chiffré :									**1.725**

Exemple : Prenons le cas du message chiffré du **Tableau 1.14**. Une valeur de $k = 6$ nous emmène à une variance égale à 1.725. Les détails de calcul sont dans notre **Tableau 1.15**. Ce qui est supérieure à la valeur de la variance correspondante à un tirage aléatoire de 6 lettres au sein d'un message en anglais donné : $0.381 \times (1 - 0.381) \times 6 = 1.415$. Une estimation fausse de la valeur de k donc.

En répétant les mêmes étapes mais cette fois-ci pour $k = 8$, nous obtenons une variance égale à 0.462 pour la totalité du message, comparé à une variance égale à $0.381 \times (1 - 0.381) \times 8 = 1.887$ si le tirage s'est fait aléatoirement. La valeur $k = 8$ pourrait donc être la bonne.

1.3.4.3 Détection de l'ordre de permutation correcte ou la clé de permutation m

Notre analyse du paragraphe 1.3.4.2 revient donc à effectuer une superposition de $k = 8$ colonnes. Voir le **Tableau 1.16**. Il nous reste donc à détecter, quelle colonne va en premier, suivie par laquelle, et ainsi de suite jusqu'à l'épuisement de toutes les colonnes. [1.01]

Pour ce faire, *William Friedman* a proposé ce que l'on connait par *méthode de contact*. Cette méthode consiste à calculer la probabilité pour que les lettres de 2 colonnes données se suivent. En effet, on connait déjà pour la langue anglaise la probabilité d'apparition de tout digraphe donné. Le

produit de chacune de ces valeurs, ou la somme de leur logarithme donne la probabilité pour que chacune des lettres de 2 colonnes données se suivent. Une probabilité proche de 0 (soit 1 en valeur pour laquelle le logarithme n'a pas été appliqué) équivaut à deux colonnes qui ont une forte probabilité de se suivre. Dans le cas du **Tableau 1.17**, le couple de colonnes (I, VIII) a une plus forte chance de se suivre.

Tableau 1.16 : *Superposition pour* $k = 8$.

I	II	III	IV	V	VI	VII	VIII
O	I	O	N	W	W	H	K
H	T	W	T	S	A	A	I
I	M	T	S	A	H	T	U
V	B	N	E	Y	M	E	E
R	E	A	M	I	T	H	U
S	T	T	A	A	G	M	H
V	M	H	T	P	E	E	E
A	N	I	A	T	D	I	S
H	O	G	D	N	E	E	L
T	E	N	A	A	R	L	M
B	I	S	A	E	H	B	T
L	O	D	E	O	F	O	K
R	I	E	R	W	D	H	S
H	T	N	A	B	D	I	E
L	E	U	R	C	T	U	C
H	T	P	I	D	E	S	R
L	K	B	A	R	A	I	E
B	R	L	O	E	E	E	P

Tableau 1.17 : *Méthode de contact appliqué aux couples de colonnes (I, VII) et (I, VIII). (I, VIII) a une plus forte probabilité de se suivre.*

I	VII	Probabilité d'apparition	Logarithme - Probabilité d'apparition
O	H	0.001	-3.301
H	A	0.013	-1.886
I	T	0.010	-2.000
V	E	0.008	-2.097
R	H	0.001	-3.000
S	M	0.001	-3.301

I	VIII	Probabilité d'apparition	Logarithme - Probabilité d'apparition
O	K	-	
H	I	0.006	-2.222
I	U	-	
V	E	0.008	-2.097
R	U	0.002	-2.824
S	H	0.005	-2.301

V	E	0.008	-2.097
A	I	0.001	-3.000
H	E	0.017	-1.783
T	L	0.002	-2.824
B	B	-	
L	O	0.002	-2.699
R	H	0.001	-3.000
H	I	0.006	-2.222
L	U	0.002	-2.824
H	S	-	
L	I	0.005	-2.347
B	E	0.006	-2.260
Probabilité sur la totalité du message :			**-40.640**

V	E	0.008	-2.097
A	S	0.008	-2.097
H	L	0.001	-3.301
T	M	0.001	-3.301
B	T	0.001	-3.301
L	K	-	
R	S	0.005	-2.347
H	E	0.017	-1.783
L	C	0.002	-2.699
H	R	0.001	-3.000
L	E	0.009	-2.046
B	P	-	
Probabilité sur la totalité du message :			**-35.415**

1.4 Conclusion

Ce premier chapitre nous a permis de se former une solide base sur ce qu'était la cryptographie depuis son apparition jusqu'à la fin de l'ère pré-informatique. Et en parallèle de s'initier à la cryptanalyse des différentes méthodes citées.

Rappelons que les méthodes citées dans le présent premier chapitre constituent le fondement de ceux qui vont se faire en termes d'amélioration dans la discipline qu'est la cryptologie, cette fois-ci, pour l'ère de l'arrivée de l'informatique, notre ère donc.

Mais le domaine de l'informatique étant surtout caractérisé par une capacité de faire faire des calculs complexes en un temps réduit, il est primordial pour nous de passer par des fondements mathématiques indispensable à la cryptologie, pour ne citer que la théorie de l'information, celle des nombres ainsi que les mathématiques discrètes.

CHAPITRE 2
THÉORIE DE L'INFORMATION, THÉORIE DES NOMBRES ET MATHÉMATIQUE DISCRÈTE

2.1 Introduction

L'ère informatique, dont la cryptologie qui lui est caractéristique va être décrite dans le prochain chapitre, est synonyme de l'arrivée des ordinateurs. *Ordinateur* ou *personal computer* en anglais, il est question de traitement de volume conséquent d'informations. Contrairement à ce que nous avons décrit dans le chapitre premier de cette ouvrage, où les calculs ont été effectués par l'homme lui-même. Et au mieux à l'aide de machine mécanique et/ou électrique (nous faisons référence au machine pinwheel c-36 et à celui à rotor qu'est l'Enigma).

Et qui dit volume conséquent d'informations, présume que nous devrions avoir une bonne base de théorie là-dessus pour ne pas faire une avancée aveugle sur le sujet. Entamons donc dans nos prochains paragraphes ce que l'on pourrait en développer.

2.2 Théorie de l'information

La littérature attribue les premiers travaux sur la théorie de l'information à *Claude Shannon*. Ce dernier était un mathématicien/ingénieur qui vivait dans les années 1916. Il travaillait notamment sur l'application de la mathématique dans l'électricité – une application qui a donné naissance aux ordinateurs digitaux – mais aussi à l'un des deux piliers qui constituent la cryptologie : la cryptanalyse. [1.03]

2.2.1 Claude Shannon

2.2.1.1 Quelques définitions

Pour traiter le sujet (théorie de l'information), Claude Shannon introduisait quelques concepts – certains desquels ne sont pas si loin de ce que nous avions déjà énumérés dans notre paragraphe 1.1 dont l'énumération est ce qui suit : [1.03]

- *La source d'information* : dans le domaine de la cryptographie, la source d'information produit à la fois le texte en clair et en applique un chiffre donné pour générer le texte chiffré ;

- *L'émetteur* : prépare le message chiffré pour que celui-ci soit prêt à être transporté par le canal ;

- *Le canal* : c'est le support qui va transporter le signal correspondant au message chiffré de l'émetteur au récepteur ;

- *Le récepteur* : il transforme le signal en message chiffré qui lui correspond. On acceptera également que le récepteur effectue aussi le décryptage du message chiffré pour donner le message en clair ;

- *Le destinataire* : c'est celui (mais cela pourrait être également un objet, une machine par exemple) pour lequel le message a été envoyé. Il se pourrait donc des fois que le récepteur et le destinataire se fusionnent ;

- *L'entropie de l'information* : disons simplement pour le moment que celle-ci représente la quantité d'informations présente dans un message donné. Nous aborderons en détails cette dernière énumération dans nos prochains paragraphes ;

En addition aux concepts que nous avons précédemment énumérés, Claude Shannon énumérait quelques importants théorèmes.

2.2.1.2 Théorèmes

a. *Premier théorème ou théorème du codage source*

Le théorème stipule qu'il est impossible de compresser une chaine de variables aléatoires indépendantes, identiquement distribuées (i.i.d.) – la longueur de la chaîne tendant donc vers l'infini – de telle sorte que le taux du code (la longueur moyenne des codes des variables) soit inférieur à l'entropie de la source, sans avoir la certitude que l'on perd de l'information. On peut cependant avoir une compression avec un taux de code proche de ladite entropie. [1.03], [2.01]

Démontrons tout d'abord la seconde moitié de l'affirmation. Comme quoi, on peut recouvrir le message original en utilisant un taux de code proche voire égale à l'entropie de la source.

Démonstration :

♣

Soit $x = x_1 \dots x_N$ un message aléatoire de N lettres, indépendamment tirées d'un alphabet $\mathcal{A} = \{a_1, \dots, a_K\}$, avec chacune une probabilité d'apparition de :

$$p(a_k) = p_k \in (0,1] \text{ soit } 0 < p_k \leq 1 \text{ pour } k = 1, \dots, K \qquad \textbf{(}2.01\textbf{)}$$

avec $\sum_k p_k = 1$. Un message x est donc *une instance* ou *une réalisation* de l'ensemble de messages $X \equiv X_1 \dots X_N$ où chaque lettre aléatoire X_n est identique à un ensemble fixé de lettres X :

$$X_n = X \text{ avec } n = 1, \dots, N \qquad \textbf{(}2.02\textbf{)}$$

On pourrait tomber sur un message particulier $x = x_1 \dots x_N$ ayant une probabilité :

$$p(x_1 \dots x_N) = p(x_1) \dots p(x_N) \qquad \textbf{(}2.03\textbf{)}$$

qui confirme le fait que toutes les lettres sont statistiquement indépendantes entre elles.

Maintenant considérons un assez long message x. Typiquement la lettre a_k va apparaître avec une fréquence $N_k \approx Np_k$. Ainsi, la probabilité d'un tel message typique est de :

$$p(x) = p_{typ} \equiv p_1^{N_1} \dots p_K^{N_K} = \prod_{k=1}^{K} p_k^{N_k} = \prod_{k=1}^{K} p_k^{Np_k} \qquad \textbf{(}2.04\textbf{)}$$

Nous voyons donc que de tels messages typiques sont uniformément distribués par la probabilité p_{typ}. Ce qui nous emmène au fait qu'un message typique T a pour longueur :

$$|T| = \frac{1}{p_{typ}} \qquad \textbf{(}2.05\textbf{)}$$

Si nous encodons chaque membre de T par un message binaire, nous avons besoin de :

$$I_N = \log|T| = -N \sum_{k=1}^{K} p_k \log p_k = NH(X) \quad \textbf{(2.06)}$$

bits, où $H(X)$ étant l'entropie de Shannon de l'ensemble de lettres sources. Ainsi pour un message suffisamment long, le nombre moyen de bits par lettre revient à :

$$I = \frac{1}{N} I_N = H(X) \quad \textbf{(2.07)}$$

♦

Maintenant démontrons la première moitié. Comme quoi l'information est perdue si l'on essaye d'encoder le message avec un taux de code inférieur à l'entropie de la source.

Démonstration :

♣

Pour pouvoir faire la présente démonstration, nous avons besoin de faire entrer le concept de variable aléatoire couplé avec la loi des grands nombres. Soit X toujours un ensemble de lettres, la fonction $f: \mathcal{A} \to \mathbb{R}$ définit une *variable aléatoire* à valeur réelle, discrète. Les réalisations de $f(X)$ sont les nombres réels $f(x), x \in \mathcal{A}$. La moyenne de $f(X)$ est définie par :

$$\langle f(X) \rangle \coloneqq \sum_{x \in A} p(x) f(x) = \sum_{k=1}^{K} p_k f(a_k) \quad \textbf{(2.08)}$$

La variance est donnée par :

$$\Delta^2 f(X) \coloneqq \langle f^2(X) \rangle - \langle f(X) \rangle^2 \quad \textbf{(2.09)}$$

Pour la séquence $f(X) \equiv f(X_1), \ldots, f(X_N)$ nous définissons sa moyenne arithmétique par :

$$A \coloneqq \frac{1}{N}\sum_{n=1}^{N} f(X_n) \tag{2.10}$$

qui est également une variable aléatoire. Etant donné que X_n n'est autre qu'une copie identique de l'ensemble de lettres X, la moyenne de A est égale à la moyenne de $f(X)$:

$$\langle A \rangle \coloneqq \frac{1}{N}\sum_{n=1}^{N} \langle f(X_n) \rangle = \langle f(X) \rangle \tag{2.11}$$

La variance en est donc :

$$\Delta^2 A = \langle A^2 \rangle - \langle A \rangle^2 \tag{2.12}$$

$$= \frac{1}{N^2}\sum_{n,m} \langle f(X_n) f(X_m) \rangle - \frac{1}{N^2}\sum_{n,m} \langle f(X_n) \rangle \langle f(X_m) \rangle \tag{2.13}$$

$$= \frac{1}{N^2}\sum_{n} \{\langle f^2(X_n) \rangle - \langle f(X_n) \rangle^2\} \tag{2.14}$$

$$\Delta^2 A = \frac{1}{N}\Delta^2 f(X) \tag{2.15}$$

L'écart type relatif de A est donné par :

$$\frac{\Delta A}{\langle A \rangle} = \frac{1}{\sqrt{N}}\left(\frac{\Delta f(X)}{\langle f(X) \rangle}\right) \tag{2.16}$$

Ce qui nous emmène à la conclusion qui suit. Pour un grand nombre N la moyenne arithmétique de

la séquence $f(X)$ et l'ensemble de la moyenne de $f(X)$ coïncident. C'est la loi des grands nombres. Cette loi est responsable de la validation des expériences statistiques.

En faisant intervenir le langage ε, δ dans cette loi des grands nombres, on arrive à ce qui va suivre. Pour $\delta > 0$, nous définissons l'ensemble typique T de la séquence aléatoire X en tant comme l'ensemble des réalisations $x \equiv x_1, \dots, x_N$ de telle sorte que :

$$f(X) - \delta \leq \frac{1}{N}\sum_{n=1}^{N} f(x_n) \leq f(X) + \delta \qquad \textbf{(}2.17\textbf{)}$$

La loi des grands nombres implique que pour tout $\varepsilon, \delta > 0$ il y a un nombre entier N_0 tel pour tout $N > N_0$ la probabilité totale de toutes les séquences typiques vérifie :

$$P_T \equiv \sum_{x \in T} p(x) \geq 1 - \varepsilon \qquad \textbf{(}2.18\textbf{)}$$

La probabilité totale P_T représente la probabilité pour qu'une séquence x choisie aléatoirement soit contenue dans l'ensemble typique T. Maintenant, considérons la variable aléatoire :

$$f(X) \coloneqq -\log p(X) \qquad \textbf{(}2.19\textbf{)}$$

La moyenne de $f(X)$ est égale à l'entropie de Shannon de de l'ensemble X :

$$\langle f(X) \rangle \coloneqq -\sum_{x \in \mathcal{A}} p(x) \log p(x) = H(X) \qquad \textbf{(}2.20\textbf{)}$$

L'ensemble typique contient donc tous les messages x dont la probabilité vérifie :

$$H - \delta \leq -\frac{1}{N}\sum_{n=1}^{N} \log p(x_n) \leq H + \delta \qquad \textbf{(}2.21\textbf{)}$$

Ce qui équivaut à :

$$2^{-N(H+\delta)} \leq p(x) \leq 2^{-N(H-\delta)} \quad \textbf{(}2.22\textbf{)}$$

Où $H \equiv H(X)$. D'après la loi des grands nombres, la probabilité pour un message x tiré d'une manière aléatoire tombe dans l'ensemble typique T est :

$$P_T \equiv \sum_{x \in T} p(x) \geq 1 - \varepsilon \quad \textbf{(}2.23\textbf{)}$$

Si nous encodons seulement les séquences typiques, la probabilité d'erreur :

$$P_{err} \coloneqq 1 - P_T \leq \varepsilon \quad \textbf{(}2.24\textbf{)}$$

peut-être aussi petite que l'on veut en choisissant N suffisamment grand. Maintenant, déterminons combien d'ensemble typique existe-t-il. La partie gauche de **(**2.22**)** nous donne :

$$p(x) \geq 2^{-N(H+\delta)} \quad \textbf{(}2.25\textbf{)}$$

$$\Leftrightarrow \sum_{x \in T} p(x) \geq |T| 2^{-N(H+\delta)} \quad \textbf{(}2.26\textbf{)}$$

La partie droite de **(**2.22**)** nous donne :

$$p(x) \leq 2^{-N(H-\delta)} \quad \textbf{(}2.27\textbf{)}$$

$$\Leftrightarrow \sum_{x \in T} p(x) \leq |T| 2^{-N(H-\delta)} \quad \textbf{(}2.28\textbf{)}$$

qui avec la formule **(**2.23**)** conduit à :

$$|T|2^{-N(H-\delta)} \geq 1-\varepsilon \quad \textbf{(}2.29\textbf{)}$$

$$\Leftrightarrow |T| \geq (1-\varepsilon)2^{N(H-\delta)} \quad \textbf{(}2.30\textbf{)}$$

Les équations **(**2.28**)** et **(**2.30**)** nous conduisent à :

$$(1-\varepsilon)2^{N(H-\delta)} \leq |T| \leq 2^{N(H+\delta)} \quad \textbf{(}2.31\textbf{)}$$

Pour $N \to \infty$ nous pouvons choisir $\varepsilon, \delta = 0$ et obtenir l'équation souhaitée :

$$|T| \to 2^{NH(X)} \quad \textbf{(}2.32\textbf{)}$$

Ainsi, nous avons besoin de $I_N \to NH(X)$ bits pour encoder le message. En d'autres termes, l'information contenue par lettre est donc $I = H(X)$ bits.

Entrons maintenant dans le vif du sujet, comme quoi s'il serait possible d'augmenter le taux de compression. D'aller en deçà de $H(X)$ pour encoder chacune des lettres de l'ensemble X. Posons $\varepsilon' > 0$ et encodons uniquement les séquences appartenant au sous ensemble typique $T' \subset T$ dont le cardinal s'obtient par :

$$|T'| \leq (1-\varepsilon)2^{N(H-\delta-\varepsilon')} < 2^{N(H-\delta-\varepsilon')} \quad \textbf{(}2.33\textbf{)}$$

La partie droite de **(**2.22**)** la probabilité d'une séquence typique est majorée par :

$$p(x) < p_{max} \equiv 2^{-N(H-\delta)} \quad \textbf{(}2.34\textbf{)}$$

Si nous encodons uniquement les séquences typiques dans le sous ensemble typique T', la

probabilité pour que la dite séquence soit contenue dans T'est donnée par :

$$\mathrm{P}_{T'} = \sum_{x \in T'} p(x) \quad \textbf{(}2.35\textbf{)}$$

$$\leq |T'|.p_{max} = 2^{N(H-\delta-\varepsilon')}2^{-N(H-\delta)} \quad \textbf{(}2.36\textbf{)}$$

$$\leq 2^{-N\varepsilon'} \quad \textbf{(}2.37\textbf{)}$$

Parce que $\varepsilon' > 0$, la probabilité que l'encodage s'effectue avec succès tend vers 0 lorsque $N \to \infty$.

$$\mathrm{P}_{T'} \to 0 \quad \textbf{(}2.38\textbf{)}$$

En conclusion, si nous compressons le message en dessous de $NH(X)$ bits, nous ne serons pas en mesure d'encoder tous les messages typique et lorsque $N \to \infty$, nous perdrons de l'information.

♦

b. *Deuxième théorème ou théorème du codage canal*

Ce théorème statue qu'il est possible de transmettre des données numériques sur un canal, même bruité, presque sans erreur à un débit maximum calculable.

La limite ou capacité de Shannon d'un canal est le débit maximal de transfert d'information sur ce canal pour un certain niveau de bruit donné. [1.03], [2.02]

Démonstration :

♣

La formulation mathématique du théorème est attribuée par la littérature au binôme *Shannon-*

Hartley et est donnée par ce qui suit :

$$C = B \log\left(1 + \frac{S}{N}\right) \quad \textbf{(}2.39\textbf{)}$$

Où :

- C est la capacité du canal en bits par seconde (bits/s) ;

- B est la bande passante du canal en hertz (Hz) ;

- S/N est le rapport signal sur bruit (*SNR* ou *Signal to Noise Ratio* en anglais) ;

Ce théorème peut être justifié comme suit. Supposons que nous recevions un signal accompagné par du bruit à un moyenne quadratique σ, et que le signal a été quantifié avec des niveaux séparés par $a = \lambda\sigma$. Si à λ a été assignée une valeur suffisamment large, nous pouvons espérer être en mesure de reconnaitre le niveau du signal avec un probabilité d'erreur acceptable. En supposant par la suite que chaque message soit représenté par un niveau de voltage. S'il y a M messages possibles, il devra y avoir M niveaux. La puissance moyenne du signal est donc :

$$S = \frac{M^2 - 1}{12}(\lambda\sigma)^2 \quad \textbf{(}2.40\textbf{)}$$

Le nombre de niveaux pour une puissance donnée de signal est donc :

$$M = \left(1 + \frac{12}{\lambda^2}\frac{S}{N}\right)^{\frac{1}{2}} \quad \textbf{(}2.41\textbf{)}$$

Où $N = \sigma^2$ la puissance du bruit. Si tous les messages ont la même probabilité d'apparaitre, alors chacun contient la même quantité d'information :

$$H = \log M = \frac{1}{2}\log\left(1 + \frac{12}{\lambda^2}\frac{S}{N}\right) \quad \textbf{(}2.42\textbf{)}$$

Où H est en bits par message ;

Pour trouver le taux d'information, nous avons besoin d'estimer combien de messages peut être transportés, par unité de temps, par un signal sur le canal. Etant donné que la réponse d'un filtre passe bas (*LPF* ou *Low Pass Filter* en anglais) idéal, de bande passante B, au niveau unité, a un temps d'attaque $\tau = 0.44/B$ dans 10 à 90% des cas. Si nous posons alors $T = 0.5/B \approx \tau$, nous pouvons avoir estimer le niveau. Le taux de message est donc :

$$r = \frac{1}{T} = 2B \qquad \textbf{(}2.43\textbf{)}$$

Où r est en messages/s.

Le taux suivant lequel le message est transféré par le canal est donc :

$$R = rH = B\log\left(1 + \frac{12}{\lambda^2}\frac{S}{N}\right) \qquad \textbf{(}2.44\textbf{)}$$

Ceci est équivalent au théorème de Shannon-Hartley en posant $\lambda^2 = 12$. R aura pour unité bits/s.

♦

2.2.2 Concepts de base autour de la théorie de l'information

2.2.2.1 Entropie d'une information

a. *Définition*

L'entropie d'une information se définit comme étant la mesure de la quantité d'information contenue dans un message. Cette définition équivaut au fait que l'entropie est également la mesure de l'incertain dans un message. [1.03], [2.03], [2.04]

Elle est donnée par la formule **(**2.45**)** qui suit. L'origine de la formule a déjà été évoquée par les

équations **(**2.06**)** et **(**2.07**)** :

$$H(X) = -\sum_{k=1}^{K} p_k \log p_k \qquad \textbf{(}2.45\textbf{)}$$

Nous entamerons dans les sous sous-titres qui vont suivre quelques caractéristiques de l'entropie.

b. *Continuité*

La mesure est continue, de telle sorte qu'une petite modification dans les valeurs de probabilités se traduit par une petite variation de la valeur de l'entropie. [2.04]

Démonstration :

♣

Une fonction donnée $f(x)$ est continue en un point donné c si $\forall\ \varepsilon > 0, \exists\ \delta > 0$ tel que :

$|x - c| < \delta \Rightarrow |f(x) - f(c)| < \varepsilon$. (1)

En posant $x = p_k$, on aura à démontrer donc la continuité de $H(X) = -\sum_k x \log x$.

Une propriété des fonctions continues stipule que la somme d'un nombre fini (nous faisons référence à k, le cardinal de l'alphabet $\mathcal{A}$ dans lequel est tirée chaque lettre du message x_n. Se référer au paragraphe 2.2.1.2 a). D'après cette propriété donc, $H(X) = -\sum_k x \log x$ est continue $\Leftrightarrow x \log x$ est continue.

Une autre propriété des mêmes fonctions (fonctions continues) stipule également que le produit d'un nombre fini de fonctions continues est continu. $x \log x$ est donc continue $\Leftrightarrow x$ est continue et $\Leftrightarrow \log x$ est continue.

La continuité de la fonction $f(x) = x$ n'est plus à démontrer. $\delta = \varepsilon$ rempli la condition nécessaire énoncée plus haut (1).

La fonction $f(x) = \log x$ est continue pour $x \in]0,1]$. En effet :

$|\log x - \log c| < \varepsilon \ \Rightarrow -\varepsilon < \log x - \log c < \varepsilon$;

$\Rightarrow -\varepsilon < \log\frac{x}{c} < \varepsilon \Rightarrow ce^{-\varepsilon} < x < ce^{\varepsilon}$;

$\Rightarrow c(e^{-\varepsilon} - 1) < x - c < c(e^{\varepsilon} - 1) \Rightarrow |x - c| < \min\{c(e^{\varepsilon} - 1), |c(e^{-\varepsilon} - 1)|\}$;

$\Rightarrow |x - c| < 1 - e^{-\varepsilon}$ soit $\delta = 1 - e^{-\varepsilon}$;

Pour $x \longrightarrow 0$, la fonction $f(x) = x \log x = \frac{\log x}{1/x}$ tend vers $\frac{1/x}{-1/x^2} = -x$ d'après la règle d l'hôpital. Qui est également une fonction continue.

♦

c. *Symétrie*

La valeur de l'entropie ne change pas si la probabilité d'apparition de chacune des lettres a_k se retrouve être permutée entre elles. [2.04]

Démonstration :

♣

L'entropie se définissant d'une part par $H(X) = -\sum_{k=1}^{K} p_k \log p_k$, et que d'autre part, l'addition étant une opération ayant une propriété commutative suffisent à démontrer cette propriété.

♦

d. *Maximum*

L'entropie est à son maximum lorsque toutes les lettres de la source sont équiprobables. [2.04]

$$H(a_1, \ldots, a_K) \leq H\left(\frac{1}{K}, \ldots, \frac{1}{K}\right) = \log K$$

Démonstration :

♣

L'inégalité de Jensen énonce que :

$H(X) = E\left[\log\frac{1}{p(X)}\right] \leq \log\left(E\left[\frac{1}{p(X)}\right]\right) = \log K$;

♦

Pour les évènements équiprobable, l'entropie augmente lorsque le cardinal (ou nombre de lettres) de la source augmente.

Démonstration :

♣

$H\left(\frac{1}{K}, \dots, \frac{1}{K}\right) = \log K$; (1)

$H\left(\frac{1}{K+1}, \dots, \frac{1}{K+1}\right) = \log(K+1)$; (2)

Comme on a $\log(K+1) \geq \log K$, ce qui démontre $H\left(\frac{1}{K+1}, \dots, \frac{1}{K+1}\right) \geq H\left(\frac{1}{K}, \dots, \frac{1}{K}\right)$;

♦

e. *Additivité*

La valeur de l'entropie est indépendante du fait de subdiviser le processus de génération d'information en plusieurs sous parties. Cette propriété implique que l'on peut calculer l'entropie d'un système si à partir de celle des sous-systèmes qui le composent, et ce si l'on sait comment les sous-systèmes interagissent entre eux. [2.04]

Démonstration :

♣

Supposons que notre alphabet de K éléments soit subdivisé en L sous-ensemble de K_l éléments

chacun, avec $l = 1, \dots, L$. La propriété de l'additivité se formalise simplement comme suit pour un alphabet à évènements équiprobables :

$$H_K = H_L + \sum_{l=1}^{L} \frac{K_l}{K} H_{K_l}$$

Où $H_K = H\left(\frac{1}{K}, \dots, \frac{1}{K}\right) = \log K$, $H_L = H\left(\frac{K_1}{K}, \dots, \frac{K_L}{K}\right)$ et $H_{K_l} = H\left(\frac{1}{K_l}, \dots, \frac{1}{K_l}\right) = \log K_l$;

Pour la démonstration, on sait que :

$$H_L = -\sum_{l=1}^{L} \frac{K_l}{K} \log \frac{K_l}{K} = \sum_{l=1}^{L} \frac{K_l}{K} (\log K - \log K_l) = \sum_{l=1}^{L} \frac{K_l}{K} \log K - \sum_{l=1}^{L} \frac{K_l}{K} \log K_l$$

$$= \log K - \sum_{l=1}^{L} \frac{K_l}{K} \log K_l$$

D'où :

$$\log K = H_K = H_L + \sum_{l=1}^{L} \frac{K_l}{K} \log K_l = H_L + \sum_{l=1}^{L} \frac{K_l}{K} H_{K_l}$$

♦

f. *Positive*

L'entropie d'une variable aléatoire discrète est positive. [2.04]

Démonstration :

♣

Comme $0 \leq p_k \leq 1$ on a $\log p_k \leq 0$ donc $-\log p_k \geq 0$ de même pour $-p_k \log p_k \geq 0$.

Ce qui nous emmène à $H(X) = -\sum_{k=1}^{K} p_k \log p_k \geq 0$;

♦

g. *Evènement à probabilité 0 n'influence pas la valeur de l'entropie*

Enlever ou ajouter un évènement de probabilité 0 n'affecte pas la valeur de l'entropie. [2.04]

Démonstration :

♣

Supposons que $p_{K+1} = 0$, on a $p_{K+1} \log p_{K+1} = 0$;

Ce qui nous emmène à $H_{K+1} = H(p_1, \dots, p_K, p_{K+1}) = H_{K+1}(p_1, \dots, p_K, 0)$;

Donc $H_{K+1} = -\sum_{k=1}^{K} p_k \log p_k - p_{K+1} \log p_{K+1} = -\sum_{k=1}^{K} p_k \log p_k = H_K$;

♦

2.2.2.2 Confusion et diffusion

a. *Confusion*

Confusion et diffusion sont deux propriétés de méthode cryptographique énumérées par Claude Shannon. [1.03]

La confusion se veut d'être une propriété selon laquelle la relation entre le texte en clair et la clé de chiffrement doit être difficilement décelable. Une lettre chiffrée devrait donc dépendre de plusieurs lettres de la clé de chiffrement à la fois. Par opposition au fait de dépendre uniquement d'une seule lettre de la clé de chiffrement.

b. *Diffusion*

La diffusion quant à elle est une propriété qui garantit que toute redondance que l'on pourrait trouver dans le texte en clair ne le soit plus dans le texte chiffré. Soit en d'autres termes, elle (la redondance) est dissipée d'une manière convenable à une partie du texte chiffré. [1.03]

2.2.2.3 Transformation complète et effet d'avalanche

a. *Transformation complète*

Une transformation est dite complète si chacune des lettres du texte chiffré dépend de toutes les

lettres du texte en clair. Ainsi, s'il est possible de trouver une version simplifiée de ladite transformation pour chacune des lettres chiffrées, alors elle dépendra en son entrée de toutes les lettres en clair. [2.04]

b. *Effet d'avalanche*

L'avalanche est une propriété de méthode cryptographique qui se veut qu'une petite modification dans le texte en clair ait une répercussion importante dans le texte chiffré, donc comme une avalanche. Cette propriété a une similitude à celle qu'est la diffusion de Claude Shannon, l'avalanche est cette fois-ci attribuée par la littérature à *Horst Feistel*. Ce dernier était un chercheur américano-allemand en cryptographie, recherches qui ont culminé avec l'apparition de la *Data Encryption Standard* ou *DES* dans les années 1970. [2.04]

c. *Critère d'avalanche strict*

Le critère d'avalanche strict est une formalisation de l'effet d'avalanche. Il est satisfait si le fait de modifier un seul bit du texte en clair conduit à une modification de tous les bits de sortie du message chiffré avec une probabilité de 0.5, soit la moitié des bits chiffrés change en d'autres termes. Le concept de critère d'avalanche strict a été introduit par *Webster* et *Tavares* en 1985. [2.04]

2.2.2.4 Distance et poids de Hamming

a. *Distance de Hamming*

La distance de Hamming est donnée par le nombre de caractères qui diffèrent entre deux textes donnés. Elle est dénotée par $h(x, y)$. La distance de Hamming est en d'autres termes le nombre de substitutions nécessaires pour aller d'un texte à un autre. [1.03]

Pour notre ère, qui est caractérisée par la cryptographie moderne, l'information représentée sous forme de bits. La distance de Hamming est donc donnée par le nombre de bits 1 résultant de l'opération eXclusive OR ou XOR entre les deux textes x et y.

$$h(x, y) = x \oplus y \qquad \textbf{(}2.46\textbf{)}$$

Où $\oplus$ représente l'opération XOR.

b. *Poids de Hamming*

Le poids de Hamming est un peu identique au concept de distance de Hamming. Il s'agit de compter le nombre de bits égal à 1 dans un texte donné. Elle équivaut donc à calculer la distance de Hamming du texte à un second autre dont chacun des bits est égal à 0. [1.03]

2.2.2.5 Principe de Kerckhoffs et/ou maxime de Shannon

Le principe de Kerckhoffs se veut que « la sécurité d'un chiffre (méthode cryptographique) dépende uniquement du fait que la clé de chiffrement est tenue secrète et non du fait que les détails de l'algorithme lui-même soient tenus secrets ». Shannon, de par sa maxime a évoqué la formulation qui suit : « Nous devrions construire notre chiffre (méthode cryptographique toujours) sous la connaissance que l'ennemi pourrait être en possession de ladite algorithme ». [1.03]

Le principe de Kerckhoffs et la maxime de Shannon sont donc d'accord sur un point : seule la clé doit être tenue secrète. Le fait de connaitre donc que quelqu'un a utilisé du SHA-256 bits, du Blowfish, du Serpent ou quoi que ce soit d'autres comme type d'algorithme, ne met en rien en périls la sécurité de son message chiffré tant qu'il garde bien secrète sa clé de chiffrement.

2.2.3 Mathématique binaire

La mathématique binaire se définit comme étant la mathématique qui utilise la base 2 au lieu de la base ordinaire 10. La base 10 est plus connue par la plupart des personnes lambda suite au fait qu'il est plus facile de compter 10 par 10, le nombre de doigts et d'orteils que nous avons aidant sûrement.

2.2.3.1 Correspondance nombre binaire nombre décimal

Les nombres décimaux utilisent des chiffres de $0, 1, \dots, 9$. Dans la base binaire, ils (ces même nombres) sont représentés sous forme de 0 et de 1 uniquement. Le **Tableau 2.01** montre la correspondance entre les dix premiers chiffres décimaux et leur représentation en binaire. [1.03]

Maintenant pour passer de la base binaire vers la base décimale, nous pourrions procéder par la méthode qui suit. Chaque bit d'un nombre binaire équivaut à 2^n où n étant le rang du bit donné en comptant de la droite vers la gauche (bien sûr) et en commençant par 0. Cette valeur de 2^n est considérée si le bit au n-ième place est égal à 1, si non elle ignorée. Ensuite, il faut additionner le

tout. Le **Tableau 2.02** illustre ces explications.

Tableau 2.01 : *Correspondance entre nombre décimal et nombre binaire.*

Nombre en base 10 (décimal)	Nombre en base 2 (binaire)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tableau 2.02 : *Conversion nombre binaire vers nombre décimal.*

Rang du bit n	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valeur du n-ieme bit	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur de 2^n	128	64	32	16	8	4	2	1
Nombre binaire	1	0	1	1	1	0	0	1
Nombre décimal correspondant	128	0	32	16	8	0	0	1
					= 128 + 32+ 16 + 8 +1 = 185			

2.2.3.2 Opérations binaires

Les opérations binaires nécessaires à la cryptographie sont principalement le OR, le AND et le XOR. On pourrait également ajouter l'opération NOT. Cette dernière, contrairement au OR, AND and XOR, requiert uniquement une entrée, si deux sont nécessaires pour les 3 premières opérations citées. [1.03]

a. *OR*

Tableau 2.03 : *Opération binaire OR.*

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Le résultat du OR est égal à 1 si au moins un de ses deux entrées est égale à 1. 0 sinon. Se référer à notre **Tableau 2.03**.

b. *AND*

Le résultat du AND est égal à 1 si toutes ses deux entrées sont égales à 1. 0 sinon. Se référer à notre **Tableau 2.04**.

Tableau 2.04 : *Opération binaire AND.*

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

c. *XOR*

Le résultat du XOR est égal à 1 ses deux entrées sont différentes. 0 sinon. Se référer à notre **Tableau 2.05**.

Tableau 2.05 : *Opération binaire XOR.*

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

L'opération XOR est très importante dans le domaine de la cryptographie. Ceci est dû au fait que cette opération est non seulement réversible, mais que son inverse aussi n'est autre qu'elle-même. En d'autres termes, si l'on a $a\ XOR\ b\ =\ c$, alors l'on pourra retrouver a en évaluant $b\ XOR\ c$.

d. *NOT*

Tableau 2.06 : *Opération binaire NOT.*

a	NOT a
0	1
1	0

Le NOT consiste à complémenter un bit donné. Si son entrée est 0, la sortie en est 1. Si à l'entrée on a 1, sa sortie est 0. Se référer à notre **Tableau 2.06**.

2.3 Théorie des nombres et mathématique discrète

La théorie des nombres se définit comme étant l'étude des nombres entiers positifs. L'étude des nombres premiers et de leur factorisation constitue un aspect important de la théorie des nombres, du moins pour son application dans le domaine de la cryptographie.

Comme avant-goût, on pourrait citer par exemple l'usage des nombres premiers (2 dans notre cas) dans la constitution du corps de Galois (ou Galois Field an anglais) $\mathrm{GF}(2^8)$ sur lequel se base la S-box (boîte à substitution) du chiffre SHARK. SHARK étant un algorithme de cryptographie prédécesseur à l'AES (Advanced Encryption Standard).

2.3.1 Classification des nombres

Nous citerons dans cet ouvrage 5 classifications usuelles des nombres. Les nombres entiers naturels dénotés par $\mathbb{N}$, les entiers relatifs $\mathbb{Z}$, les nombres rationnels $\mathbb{Q}$, les nombre réels $\mathbb{R}$ et les nombres complexes $\mathbb{C}$. [1.03]

2.3.1.1 Nombres entiers naturels

Ce sont les nombres utilisés naturellement pour compter $\{1, 2, 3, \dots\}$. Certaines sources incluent 0, d'autres pas. On pourrait considérer comme ensemble des entiers naturels donc $\{0, 1, 2, 3, \dots\}$.

2.3.1.2 Nombres entiers relatifs

Cet ensemble est constitué par l'ensemble des entiers naturels auquel on rajoute une extension du côté négatif $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$.

2.3.1.3 Nombres rationnels

Les nombres rationnels sont constitués par les nombres qui peuvent être représentés sous forme de ratio de deux nombres entiers a et b.

Exemple : $0.5 = \frac{1}{2}; 0.33333 \dots = \frac{1}{3}; -0.7 = \frac{-7}{10}$;

2.3.1.4 Nombres réels

Les nombres réels sont constitués par les nombres rationnels, auxquels on rajoute les nombres irrationnels dénotés $\mathbb{I}$. Ces derniers (les nombres irrationnels) sont constitués par des nombres qui ne sont pas représentables sous forme de fraction de nombres entiers. Mais peuvent par contre en avoir une approximation pour une marge d'erreur donnée.

Exemple : A deux décimaux près $\pi = 3.14 = \frac{314}{100}$; de même $\sqrt{2} = 1.41 = \frac{141}{100}$.

2.3.1.5 Nombres complexes

Les nombres complexes sont de la forme $c = a + ib$ où a et b sont des nombres réels et i est par définition un nombre imaginaire qui élevé au carré donne -1 soit $i \coloneqq \sqrt{-1}$.

Exemple : $1 + i\sqrt{2}$; ou encore – i ;

2.3.2 Quelques opérations et/ou propriétés importantes

Nous avons vu les différentes classifications des nombres dans notre paragraphe précédent. Nous verrons dans ce qui va suivre les opérations que l'on pourrait effectuer sur ces derniers. Ou encore des propriétés que ces derniers peuvent avoir. On remarquera que seuls les nombres entiers naturels, et relatifs dans une moindre mesure (nous faisons référence au nombre -1 par exemple, mais qui équivaut à $n - 1$ dans l'opération $(\mathrm{mod}\ n)$), seront majoritairement manipulés dans le domaine de la cryptographie.

2.3.2.1 Divisibilité et algorithme de division

a. *Divisibilité*

On dit qu'un nombre entier b divise a si l'on a $a = mb$ où m est également un nombre entier. La notation $b|a$ est majoritairement utilisée pour exprimer le fait que b divise a. b est dans ce cas également appelé *diviseur* de a. [1.02]

$$b|a \Leftrightarrow a = mb \qquad \textbf{(}2.47\textbf{)}$$

Exemple : Les diviseurs positifs de 24 sont $1, 2, 3, 4, 6, 8, 12$ et 24 lui-même.

Quelques propriétés :

- Si $a|1$ alors $a = \pm 1$;

- Si $a|b$ et $b|a$ alors $a = \pm b$;

- Tout $a \neq 0$ divise 0 ;

- Si $a|b$ et $b|c$ alors $a|c$;

- Si $a|b$ et $a|c$ alors $a|mb + nc$ pour m et n des entiers arbitraires ;

b. *Algorithme de division*

Etant donné un nombre entier n, et un autre non négatif entier a, si nous divisons a par n, nous aurons deux entiers q (pour quotient) et r (pour reste) qui obéissent à la formulation : [1.02]

$$a = qn + r \text{ où } 0 \leq r < n \text{ et } q = \lfloor {}^{a}/_{n} \rfloor \qquad \textbf{(}2.48\textbf{)}$$

Où l'opérateur $\lfloor x \rfloor$ représente l'arrondissement a l'entier directement inférieur ou égal à x.

Exemple : $a = 11$, $n = 7$, on a $11 = 1 \times 7 + 4$ donc $q = 1$ et $r = 4$.

2.3.2.2 Plus Grand Commun Diviseur ou PGCD et Algorithme d'Euclide

a. *Plus Grand Commun Diviseur ou PGCD*

Le *Plus Grand Commun Diviseur* de deux nombres entiers a et b est noté $\mathrm{PGCD}(a, b)$. Comme son nom l'indique, c'est le plus grand nombre entier c qui divise à la fois a et b. [1.02]

$$c = \mathrm{PGCD}(a, b) = \max[k \; tel \; que \; k|a \; et \; k|b] \qquad \textbf{(}2.49\textbf{)}$$

Une définition équivalente en est, c est le PGCD de a et b si :

- c divise a et b ;

- Tout diviseur de a et b divise c ;

Exemple : $PGCD(24{,}60) = 12.$

a et b sont premiers entre eux si $\mathrm{PGCD}(a, b) = 1.$

Et comme tout nombre entier non nul divise 0, on a $\mathrm{PGCD}(a, 0) = a$. Mentionnons en passant que nous travaillerons la majorité des cas, sauf explicitement mentionné, avec des nombres positifs.

b. *Algorithme d'Euclide*

L'algorithme d'Euclide est un algorithme permettant de calculer le PGCD de deux nombres a et b donnés. Ci-dessous les étapes qui le constituent : [1.02]

- Supposons que nous aimerions déterminer le nombre c qui est le PGCD de deux nombres a et b. Choisissons a et b de tel sorte que l'on ait $a \geq b > 0$. Intervertir a et b si ce n'est pas le cas ;

- Trouver un nombre r_1 de tel sorte que :

$$a = q_1 b + r_1, \text{ où } 0 \leq r_1 < b \qquad \textbf{(}2.50\textbf{)}$$

- Considérons en premier le cas que $r_1 = 0$. Dans ce cas, b divise a et plus aucun plus grand nombre ne peut diviser à la fois a et b. On a en d'autres termes $c = \mathrm{PGCD}(a, b) = b$;

- Le second cas possible étant celui où $r_1 > 0$. Dans ce cas, nous pouvons affirmer que $c|r_1$. Ceci découle du fait que si $c|a$ et que $c|b$ alors, $c|(a - q_1 b)$, soit $c|r_1$;

- Maintenant remarquons que si d est un diviseur commun arbitraire de b et r_1 alors, on aura $d|q_1 b$ et donc $d|(q_1 b + r_1)$ soit $d|a$ donc. Ce qui veut dire que $d \leq c$. On peut conclure alors $\mathrm{PGCD}(b, r_1) = \mathrm{c} = \mathrm{PGCD}(a, b)$.

- En d'autres termes, rechercher $\mathrm{PGCD}(a, b)$ revient à rechercher le $\mathrm{PGCD}(b, r_1)$. Ou encore d'une manière plus générale, à rechercher le $\mathrm{PGCD}(r_i, r_{i+1})$

$$\mathrm{PGCD}(a, b) = \mathrm{PGCD}(r_n, r_{n+1}) \text{ où } a \leq b, n \geq 0, r_0 = b\,; \qquad \textbf{(}2.51\textbf{)}$$

$$\mathrm{PGCD}(a,b) = \mathrm{PGCD}(r_n, r_{n+1}) = r_n \Leftrightarrow r_{n+1} = 0\,; \qquad \textbf{(}2.52\textbf{)}$$

Le diagramme ci-dessous (**Figure 2.01**) représente comment les étapes précédemment listées de l'algorithme d'Euclide s'enchaînent.

Exemple : Recherchons le $\mathrm{PGCD}(24, 60)$.

On a donc $a = 24$ et $b = 60$. Comme $a < b$, on a à intervertir a et b. Donc nous allons continuer avec $a = 60$ et $b = 24$.

Maintenant, nous avons $60 = 2 \times 24 + 12$. On a donc $r = 12$. Comme $r = 12 > 0$, on a à continuer l'algorithme donc, cette fois ci en prenant $a = 24$ et $b = 12$.

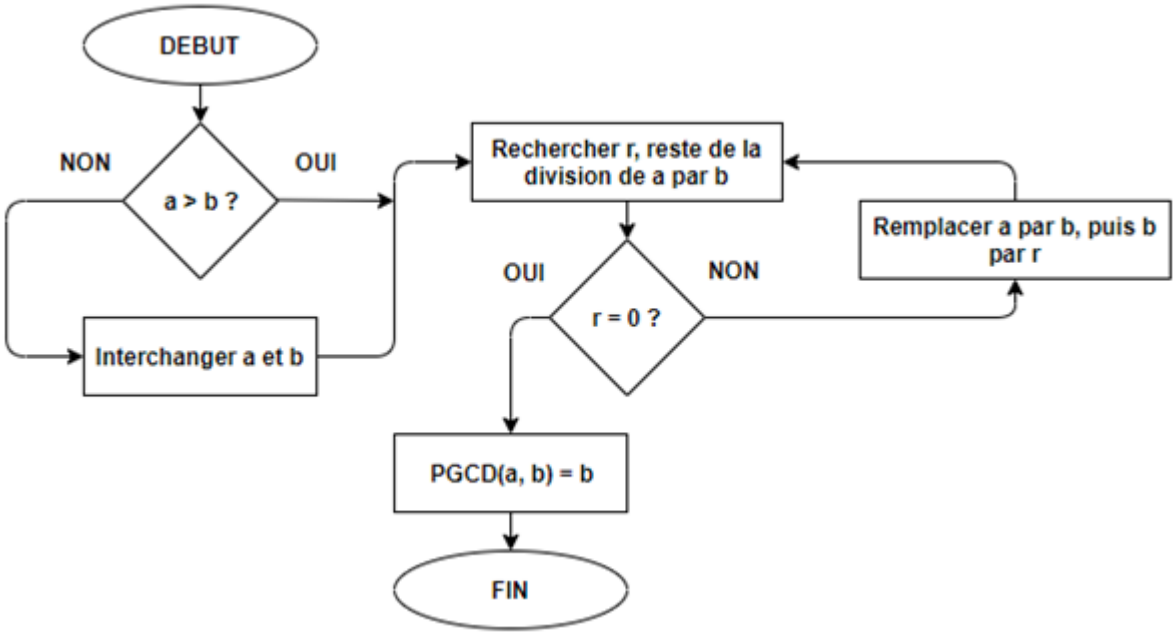


Figure 2.01 : *Algorithme d'Euclide.*

On a $24 = 2 \times 12 + 0$. Comme le reste $r = 0$, le PGCD recherché est donc la dernière valeur de b qui est 12.

$\mathrm{PGCD}(24, 60) = 12$.

2.3.2.3 Arithmétique modulaire

a. *Modulo*

Si a et n sont deux entiers, alors $a \bmod n$ se définit comme étant le reste de la division de a par n. On a : [1.02]

$$a = q \times n + r \text{ avec } q = \lfloor {}^{a}/_{n} \rfloor \text{ et que } 0 \leq r < n$$
$$a = \lfloor {}^{a}/_{n} \rfloor \times n + (a \bmod n) \quad \textbf{(2.53)}$$

Où $\lfloor x \rfloor$ représente le nombre entier immédiatement inférieur ou égal à x.

Exemple : $11 \mod 7 = 4$. Etant donné que $11 = 1 \times 7 + 4$.

Deux nombres a et b sont dits congruent modulo n si $a \bmod n = b \bmod n$. Cette propriété est dénotée par $a \equiv b \pmod n$.

$$a \equiv b \pmod n \Leftrightarrow a \bmod n = b \bmod n \quad \textbf{(2.54)}$$

Exemple : $73 \equiv 4 \pmod{23}$.

b. *Quelques propriétés de congruence*

Ci-dessous quelques propriétés importantes de congruences : [1.02]

- Si $a \equiv 0 \pmod n$ alors $n|a$;

- Si $a \equiv b \pmod n$ alors $n|(b - a)$;

- Si $a \equiv b \pmod n$ alors $b \equiv a \pmod n$;

- Si $a \equiv b \pmod n$ et $b \equiv c \pmod n$ alors $a \equiv c \pmod n$;

Démonstration :

♣

- Si $a \equiv 0 \pmod{n}$ alors on a $a = q \times n + 0 = q \times n$ d'après la formule **(**2.53**)**. Ce qui revient à dire également que $n|a$ selon cette fois ci la formule de divisibilité **(**2.47**)** ;

- Si $a \equiv b \pmod{n}$ alors on a $a = q \times n + b$. Ce qui revient à $(a - b) = q \times n$, d'où l'affirmation $n|(a - b)$;

- Si $a \equiv b \pmod{n}$ alors on a $a = q \times n + b$, soit encore $b = (-q) \times n + a$. Ce qui revient à $b \equiv a \pmod{n}$;

- Si $a \equiv b \pmod{n}$ on a $a = q \times n + b$. Et si $b \equiv c \pmod{n}$ on a $b = q' \times n + c$. Ce qui revient à $a = q \times n + q' \times n + c = (q + q') \times n + c$. Ce qui revient à $a \equiv c \pmod{n}$;

♦

c. *Opération arithmétique modulaire*

Rappelons que d'après la formule **(**2.53**)**, l'opération modulo associe à tout entier a un nombre entier de l'ensemble réduit $\{0,1, \dots, n-1\}$. La question se pose alors s'il serait possible d'effectuer des opérations dans cette ensemble. Et il s'avère que la réponse à cette question est oui. Elles (lesdites opérations) se définissent comme suit : [1.02]

- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$;

- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$;

- $[[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$;

Démonstration :

♣

Nous entamerons la démonstration du premier tiret soit la propriété liée à l'addition. La démonstration des deux tirets restant peut-être faite en adoptant la même démarche :

$$(a+b) \bmod n = (q_a n + r_a + q_b n + r_b) \bmod n$$

$$= (r_a + r_b) \bmod n$$

$$= [(a \bmod n) + (b \bmod n)] \bmod n$$

♦

d. *Quelques propriétés de l'arithmétique modulaire*

Nous allons énumérer dans le tableau ci-dessous (**Tableau 2.07**) quelques propriétés relatives à l'arithmétique modulaire : [1.02]

Tableau 2.07 : *Quelques propriétés de l'arithmétique modulaire.*

Propriétés	**Expression**
Loi commutatives	$(w+x) \bmod n = (x+w) \bmod n$; $(w \times x) \bmod n = (x \times w) \bmod n$;
Loi associative	$[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$; $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$;
Loi distributive	$[w \times (x+y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$;
L'identité	$(0+w) \bmod n = w \bmod n$; $(1 \times w) \bmod n = w \bmod n$;
Existence d'une additive inverse	Pour tout $w \in \mathbb{Z}_n$, $\exists\ z$ tel que $w + z = 0 \bmod n$;
Existence d'une multiplicative inverse, si $\boldsymbol{w}$ et $\boldsymbol{n}$ sont premiers entre eux	Pour tout $w \in \mathbb{Z}_n$, et que w et n sont premiers entre eux, $\exists\ z$, tel que $w \times z \bmod n = 1$

Exemple :

- Pour l'additive inverse dans $\mathbb{Z}_8$, 2 est l'additive inverse de 6. De même 5 est l'additive inverse de 3. En effet, on a respectivement $(2+6) \bmod 8 = 8 \bmod 8 = 0$ et $(5+3) \bmod 8 = 8 \bmod 8 = 0$;

- Pour la multiplicative inverse, dans le même ensemble $\mathbb{Z}_8$, la multiplicative inverse de 3 est 3. En effet, $(3 \times 3) \bmod 8 = 9 \bmod 8 = 1$. Rappelons que 3 a une multiplicative inverse dans $\mathbb{Z}_8$ car 3 et 8 sont premiers entre eux. Ainsi, 2 par exemple n'a pas de multiplicative inverse dans $\mathbb{Z}_8$ car 2 et 8 ne sont pas premiers entre eux. En effet $\mathrm{PGCD}(2, 8) = 2$.

2.3.2.4 Algorithme d'Euclide revisité et Algorithme d'Euclide Etendu

a. *Algorithme d'Euclide revisité*

Après avoir introduit la notion d'arithmétique modulaire dans notre paragraphe précédent 2.3.2.3 , nous sommes à présent en mesure de redéfinir d'une manière plus simple l'algorithme d'Euclide mentionné dans notre paragraphe 2.3.2.2 b. Ci-après la nouvelle formule donc : [1.02]

$$\mathrm{PGCD}(a, b) = \mathrm{PGCD}(b, a \bmod b) \qquad \textbf{(}2.55\textbf{)}$$

Pour voir d'où vient notre formule **(**2.55**)**, nous avons déjà vu dans notre paragraphe 2.3.2.2 b que $\mathrm{PGCD}(b, r_1) = \mathrm{c} = \mathrm{PGCD}(a, b)$. Où $0 \leq r_1 < b$ avec $a = q_1 b + r_1$, ce qui revient à dire $a \equiv r_1 \pmod b$, ou encore $r_1 = a \bmod b. On$ obtient finalement notre formule **(**2.55**)** soit $\mathrm{PGCD}(a, b) = \mathrm{PGCD}(b, a \bmod b)$.

Si l'on considère $\mathrm{PGCD}(a, b)$ comme étant une fonction avec respectivement a et b comme premier et second paramètre, la fonction peut se définir d'une manière récursive comme suit :

$\mathbf{PGCD}(\boldsymbol{a}, \boldsymbol{b})$:

Si $\boldsymbol{b} = \mathbf{0}$ alors $\mathbf{PGCD}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}$;

Sinon $\mathbf{PGCD}(\boldsymbol{a}, \boldsymbol{b}) = \mathbf{PGCD}(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$;

Exemple : $\mathrm{PGCD}(55, 22) = \mathrm{PGCD}(22, 55 \bmod 22) = \mathrm{PGCD}(22, 11) = 11.$

b. *Algorithme d'Euclide étendu*

L'algorithme d'Euclide étendu, en plus de calculer le $\mathrm{PGCD}(a, b)$, calcul également les deux entiers

x et y justifiant l'équation ci-dessous : [1.02]

$$ax + by = \text{PGCD}(a, b) \qquad \textbf{(}2.56\textbf{)}$$

Pour ce faire, en plus de calculer les quotients et restes q_n et r_n, l'algorithme d'Euclide étendu calcule également les entiers additionnels x_n et y_n. Les étapes à suivre sont décrites dans le **Tableau 2.08**. L'algorithme s'arrête à l'itération $n + 1$ où l'on obtient un reste $r_{n+1} = 0$. Ainsi, les valeurs de r_n, x_n et y_n justifient notre formule **(**2.56**)** précédemment citée, soit $ax_n + by_n = r_n = \text{PGCD}(a, b)$.

Démonstration :

♣

Nous avons déjà vu dans notre paragraphe 2.3.2.2 b détaillant l'algorithme d'Euclide qu'il existe r_n tel que $\text{PGCD}(a, b) = \text{PGCD}(r_n, r_{n+1}) = r_n$ avec $r_{n+1} = 0$;

Sans oublier la première égalité de division d'Euclide entre a et b soit $a = q_1 b + r_1$, où $0 \leq r_1 < b$. Qui équivaut à $r_1 = a - q_1 b \equiv ax_1 + by_1$ avec $x_1 = 1$ et $y_1 = q_1$;

Ou encore d'une manière générale et par récurrence pour un n donné : $r_{n-2} = q_n r_{n-1} + r_n$. Qui encore une fois équivaut à $r_n = r_{n-2} - q_n r_{n-1} = (ax_{n-2} + by_{n-2}) - q_n(ax_{n-1} + by_{n-1}) = a(x_{n-2} - q_n x_{n-1}) + b(y_{n-2} - q_n y_{n-1}) = ax_n + by_n$ avec $x_n = (x_{n-2} - q_n x_{n-1})$ et $y_n = y_{n-2} - q_n y_{n-1}$.

♦

Tableau 2.08 : *Algorithme d'Euclide étendu.*

$\boldsymbol{i}$	$\boldsymbol{q_i}$	$\boldsymbol{r_i}$	$\boldsymbol{x_i}$	$\boldsymbol{y_i}$
$\mathbf{-1}$	$-$	a	1	0
$\mathbf{0}$	$-$	b	0	1
$\mathbf{1}$	$\lfloor a/b \rfloor$	$a \bmod b$	$x_{-1} - q_1 x_0$	$y_{-1} - q_1 y_0$
$\mathbf{2}$	$\lfloor b/r_1 \rfloor$	$b \bmod r_1$	$x_0 - q_2 x_1$	$y_0 - q_2 y_1$
$\mathbf{3}$	$\lfloor r_1/r_2 \rfloor$	$r_1 \bmod r_2$	$x_1 - q_3 x_2$	$y_1 - q_3 y_2$

…	…	…	…	…
$\boldsymbol{n}$	$\lfloor r_{n-2}/r_{n-1} \rfloor$	$r_{n-2} \bmod r_{n-1}$	$x_{n-2} - q_n x_{n-1}$	$y_{n-2} - q_n y_{n-1}$
$\boldsymbol{n+1}$	$\lfloor r_{n-1}/r_n \rfloor$	$r_{n-1} \bmod r_n = 0$		

Exemple : effectuons l'algorithme d'Euclide étendu pour $a = 1212$ et $b = 630$. Le tableau ci-après illustre notre exemple :

$\boldsymbol{i}$	$\boldsymbol{q_i}$	$\boldsymbol{r_i}$	$\boldsymbol{x_i}$	$\boldsymbol{y_i}$
$\mathbf{-1}$	−	1759	1	0
0	−	630	0	1
1	1	582	1	-1
2	1	48	-1	2
3	12	6	13	-25
4	8	0		

On a donc $n = 3$, $r_n = r_3 = 6$, $x_n = x_3 = 13$ et $y_n = y_3 = -25$. Soit $13 \times 1212 - 25 \times 630 = 6$.

2.3.2.5 Théorème des restes chinois

Le théorème des restes chinois peut s'écrire de différentes manières, mais celle qui nous concerne se détaille comme ce qui va suivre. Soit un entier *N* tel que : [1.02]

$$N = \prod_k n_k \text{ avec } 1 \leq k \leq K \qquad \textbf{(}2.57\textbf{)}$$

Où les n_k sont deux à deux premiers entre eux, soit $\forall\ 1 \leq k, l \leq K$ avec $k \neq l$, on a $\mathrm{PGCD}(n_k, n_l) = 1$. Nous pouvons représenter tout entier A dans $\mathbb{Z}_N$ par un K-uplet dont chaque éléments appartient à $\mathbb{Z}_{n_k}$.

$$A \leftrightarrow (a_1, \dots, a_K) \qquad \textbf{(}2.58\textbf{)}$$

Où $A \in \mathbb{Z}_N$, $a_k \in \mathbb{Z}_{n_k}$et $a_k = A \bmod n_k, \forall\, 1 \leq k \leq K$. Deux assertions en découlent :

- La correspondance dans l'équation **(**2.58**)** est bijective entre $\mathbb{Z}_N$ et l'ensemble produit cartésien $\mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_K}$. Ainsi, pout tout entier $A \in \mathbb{Z}_N$ il existe un unique K-uplet $(a_1, \ldots, a_K) \in \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_K}$ qui le représente. Et inversement tout K-uplet $(a_1, \ldots, a_K) \in \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_K}$ correspond à un unique entier $A \in \mathbb{Z}_N$;

- Tout opération effectuée sur un élément $A \in \mathbb{Z}_N$ peut être effectuée indépendamment sur les K-uplet $(a_1, \ldots, a_K) \in \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_K}$.

Démonstration :

♣

Démontrons pour commencer notre première affirmation.

Pour l'unicité de la solution, supposons dans un premier temps qu'il existe deux nombres A et B solutions à toutes les congruences. Comme par définition A et B retournent le même reste quand divisés par tous les n_k, leur différence $A - B$ est donc multiple de chacun des n_k. Comme tous les n_k sont deux à deux premiers entre eux, leur produit N divise donc également $A - B$. Ainsi A et B sont donc congruents modulo N. Mais comme A et B sont des entiers positifs inférieur à N, alors leur différence est multiple de N si et seulement si $A = B$ soit $A - B = 0$.

Comme nous avons vu donc qu'à un K-uplet $(a_1, \ldots, a_K)$ correspond un seul entier A soit $A \longleftarrow (a_1, \ldots, a_K)$, le fait que le cardinal de $\mathbb{Z}_N$ et l'ensemble produit cartésien $\mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_K}$ est le même suffit à justifier la bijection de la correspondance. Soit $A \longleftrightarrow (a_1, \ldots, a_K)$.

Pour l'existence de l'entier A, nous procèderons par sa construction. Soit $N_k = {}^{N}/_{n_k}$ pour $1 \leq k \leq K$. Remarquons que $N_k = n_1 \times \ldots \times n_{k-1} \times n_{k+1} \times \ldots \times n_K$. Soit $N_l \equiv 0 \pmod{n_k}\ \forall\, l \neq k$. Maintenant soit :

$$m_k = N_k \times (N_k^{-1} \bmod n_k) \equiv 1 \pmod{n_k} \text{ pour } 1 \leq k \leq K \qquad \textbf{(}2.59\textbf{)}$$

L'entier m_k de notre formule **(**2.59**)** est bien défini et est unique dû au fait que N_k est premier par

rapport à n_k et donc possède une multiplicative inverse $\bmod n_k$. Maintenant, A peut se calculer comme suit :

$$A \equiv \sum_{k=1}^{K} (a_k m_k) \bmod M \qquad \textbf{(}2.60\textbf{)}$$

Pour montrer que notre formulation de A dans l'équation **(**2.60**)** est juste, nous devons montrer que $A \bmod n_k = a_k$ pout tout $1 \leq k \leq K$. Rappelons que $m_l \equiv 0 \pmod{n_k}$ pour tout $l \neq k$. Ce qui revient à dire que $A \bmod n_k \equiv a_k m_k \bmod n_k = a_k$ étant donné que $m_k \bmod n_k = 1$ d'après la définition de m_k dans la formule **(**2.59**)**.

Passons maintenant à la seconde affirmation.

En effet, elle découle de la propriété sur les opérations arithmétiques modulaires (se référer à notre paragraphe 2.3.2.3 c). Soient deux entiers A et B de $\mathbb{Z}_N$:

$$A \leftrightarrow (a_1, \dots, a_K)$$

$$B \leftrightarrow (b_1, \dots, b_K)$$

On a :

$$(A + B) \bmod N \leftrightarrow \big((a_1 + b_1) \bmod n_1, \dots, (a_K + b_K) \bmod n_K\big)$$

$$(A - B) \bmod N \leftrightarrow \big((a_1 - b_1) \bmod n_1, \dots, (a_K - b_K) \bmod n_K\big)$$

$$(A \times B) \bmod N \leftrightarrow \big((a_1 \times b_1) \bmod n_1, \dots, (a_K \times b_K) \bmod n_K\big)$$

♦

Une des propriétés utiles du théorème des reste chinois est qu'il permet de manipuler un nombre entier très large N en utilisant à sa place ses K-uplet, qui sont des entiers plus petits. Le prérequis en est cependant la nécessité de connaitre la factorisation de N.

Exemple : Considérons le cas où $N = 1813 = 37 \times 49$ soient $n_1 = 37$ et $n_2 = 49$, $A = 973$.

Nous avons donc aussi $N_1 = 49$ et $N_2 = 37$. En appliquant l'algorithme d'Euclide étendu, on a $N_1^{-1} = 34 \bmod n_1$ et $N_2^{-1} = 4 \bmod n_2$. En prenant le modulo de 973 respectivement par 37 et 49,

on a la représentation de $A = 973$ qui est $(11, 42)$.

Maintenant, supposons que nous aimerions additionner 678 à 973. On a $678 \leftrightarrow (678 \bmod 37, 678 \bmod 49) \equiv (12, 41)$. Il nous suffit d'additionner les 2-uplets un à un et de les réduire. Soit $(11 + 12 \bmod 37, 42 + 41 \bmod 49) = (23, 34)$. Pour voir que notre calcul est correct, recherchons l'équivalent de $(23, 34)$ dans $\mathbb{Z}_{1813}$. Selon le formule **(**2.60**)**, on a $(23, 34) \leftrightarrow (23 \times 49 \times 34 \ + 34 \times 37 \times 4) \bmod 1813 = 43350 \bmod 1813 = 1651$.

2.3.3 Nombres premiers

Un nombre premier se définit comme étant un nombre positif ayant exactement deux diviseurs : 1 et lui-même. 1 n'est pas considéré comme étant un nombre premier.

2.3.3.1 Factorisation

Tout nombre entier $a > 1$ peut être écrit d'une manière unique sous la forme [1.02], [1.03] :

$$a = p_1^{a_1} \times p_2^{a_2} \times \ldots \times p_K^{a_K} \qquad \textbf{(}2.61\textbf{)}$$

Où $p_1 < p_2 < \cdots < p_K$ sont des nombres premiers et $a_1, a_2, \ldots, a_K$ sont des nombres entiers positifs.

Exemple : $3600 = 24^4 \times 32^2 \times 52^2$; $11011 = 7 \times 11^2 \times 13$;

Soit en d'autres termes si P étant l'ensemble regroupant tous les nombres premiers p_k. On aura :

$$a = \prod_{p_k \in P} p_k^{a_k} \qquad \textbf{(}2.62\textbf{)}$$

Où $a_k \geq 0$;

2.3.3.2 Quelques propriétés

Dans la présente section, nous allons voir quelques propriétés de nombres premiers.

a. *Factorisation de produit de deux nombres entiers*

Soient a et b deux nombres entiers tels que $a = \prod_{p_k \in P} p_k^{a_k}$ et $b = \prod_{p_k \in P} p_k^{b_k}$. On a : [1.02]

$$c = a \times b = \prod_{p_k \in P} p_k^{c_k} = \prod_{p_k \in P} p_k^{(a_k + b_k)} \quad \textbf{(}2.63\textbf{)}$$

Démonstration :

♣

On a :

$c = a \times b = \prod_{p_k \in P} p_k^{a_k} \times \prod_{p_k \in P} p_k^{b_k} = \prod_{p_k \in P} (p_k^{a_k} \times p_k^{a_k})$;

$c = \prod_{p_k \in P} p_k^{(a_k + b_k)} = \prod_{p_k \in P} p_k^{c_k}$;

d'où $c_k = a_k + b_k$.

♦

b. *Divisibilité de deux nombres*

Soient deux nombre entiers a et b tels que a divise b soit $a|b$. Pour rappel $a|b$ signifie qu'il existe un nombre entier c tel que $\frac{b}{a} = c$). On a : [1.02]

$$a \left| b = \prod_{p_k \in P} p_k^{a_k} \right| \prod_{p_k \in P} p_k^{b_k} \Rightarrow a_k \leq b_k \quad \textbf{(}2.64\textbf{)}$$

Démonstration :

♣

Les p_k étant tous des nombres premiers, aucun des p_k ne divise p_l $\forall l \neq k$;

Ce qui nous emmène au fait que si $a|b$ alors $p_k^{a_k} | p_k^{b_k}$;

Donc $p_k^{a_k} | p_k^{b_k} \Longrightarrow \frac{p_k^{b_k}}{p_k^{a_k}} = p_k^{(b_k - a_k)}$ qui devrait être un entier;

Enfin $p_k^{(b_k - a_k)}$ est entier $\Leftrightarrow b_k - a_k \geq 0 \Rightarrow a_k \leq b_k$;

♦

c. *Plus grand commun diviseur ou PGCD de deux nombres*

Soient toujours deux nombres entiers a et b. Si c est le PGCD de a et b alors aon a : [1.02]

$$c_k = \min(a_k, b_k) \qquad \textbf{(}2.65\textbf{)}$$

Démonstration :

♣

c est le PGCD de a et b si et seulement si $\frac{a}{c}$ ne divise plus $\frac{b}{c}$;

En posant $c_k = \min(a_k, b_k)$, si l'on a $d = \frac{a}{c}$ et $e = \frac{b}{c}$ alors on a :

$\begin{cases} d_k = a_k - \min(a_k, b_k) \\ e_k = b_k - \min(a_k, b_k) \end{cases}$ donc $\forall k$ soit $d_k = 0$ soit $e_k = 0$;

En d'autres termes, soit $p_k^{d_k} = 1$ soit $p_k^{e_k} = 1$;

D'où les nombres d et e qui sont respectivement $\frac{a}{c}$ et $\frac{b}{c}$ sont premiers entre eux.

♦

2.3.3.3 Théorème de Fermat, indicatrice (ou *Totient function* en anglais) et théorème d'Euler

Les deux théorèmes (théorème de Fermat et d'Euler) jouent un rôle important dans la cryptographie à clé publique.

a. *Théorème de Fermat*

Le théorème de Fermat statue que, si p est premier et que a est un nombre entier non divisible par p, on a : [1.02], [1.03]

$$a^{p-1} \equiv 1 \pmod{p} \qquad \textbf{(}2.66\textbf{)}$$

Démonstration :

♣

Soit l'ensemble des nombres entiers positifs plus petit que p : $\{1,2, \dots, p-1\}$. En multipliant chaque élément par $a \bmod p$ on obtient l'ensemble $X = \{a \bmod p, 2a \bmod p, \dots, (p-1)a \bmod p\}$. Aucun élément de X n'est égal à 0 parce que p ne divise pas a. De même, il n'y a pas deux élément de X qui sont égaux. Pour voir cela, supposons que deux éléments i et j de X sont égaux, avec $1 \leq i < j \leq p-1$. On devrait donc avoir $ia \equiv ja \pmod{p}$. Parce que a et p sont premiers entre eux, on pourra enlever a dans cette équation. Et donc avoir vers la fin $i = j \pmod{p}$. Ce qui est impossible car i et j sont inférieurs à p et que p est premier. En multipliant tous les éléments de X, on aura :

$a \times 2a \times \dots \times (p-1)a \equiv 1 \times 2 \times \dots \times (p-1) \pmod{p}$;

$a^{p-1}(p-1)! = (p-1)! \pmod{p}$;

On pourra enlever $(p-1)!$ dans cette équation car ce facteur n'est divisible par p. D'où l'on a : $a^{p-1} = 1 \pmod{p}$

♦

b. *Indicatrice d'Euler*

L'indicatrice d'Euler est une fonction qui pour tout nombre entier n, retourne le nombre d'entiers inférieurs à n qui ne divisent pas n. Elle est notée $\phi(n)$. Pour un nombre premier donc, on a $\phi(n) = n - 1$. Par convention, on accepte $\phi(1) = 1$. [1.02]

Soit $n = pq$ le produit de deux nombres premiers p et q avec $p \neq q$, on a :

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1)(q-1) \quad \textbf{(}2.67\textbf{)}$$

Démonstration :

♣

Pour la démonstration, considérons l'ensemble des nombres entiers plus petits que n : $X = \{1,2, \dots, pq-1\}$. Parmi ces nombres entiers, les seuls qui ne sont premiers par rapport à $n = pq$, étant donné que p et q sont premiers, donc que les dits nombres (ceux non premiers par rapport à n) doivent avoir soit p soit q comme facteur, sont : $Y = \{p, 2p, \dots, (q-1)p\}$ et $Z = \{q, 2q, \dots, (p-1)q\}$. De plus ces deux ensembles ne s'écrasent pas, c'est-à-dire aucun membre du premier ne se trouve dans le second. Le nombre d'éléments des deux ensembles est donc : $(q-1) + (p-1)$. D'où le nombre d'élément restant, correspondant à $\phi(n)$ est :

$\phi(n) = \text{card}\big(X \text{ minus } (Y \cup Z)\big) = pq - 1 - (q - 1 + p - 1)$;

$\phi(n) = pq - q - p + 1 = (p-1)(q-1)$;

♦

c. *Théorème d'Euler*

Le théorème d'Euler statue que pour tout nombre entier a et n premiers entre eux, on a : [1.02]

$$a^{\phi(n)} \equiv 1 \pmod{n} \quad \textbf{(}2.68\textbf{)}$$

Démonstration :

♣

Si n est un nombre entier alors on aura $\phi(n) = n - 1$. Le théorème d'Euler revient à celui de Fermat. Pour les autres cas où n n'est pas premier, rappelons le fait que $\phi(n)$ est le nombre d'entiers

inférieurs à n qui sont premiers par rapport à n. Dénotons par X cet ensemble :

$$X = \{x_1, x_2, \dots x_{\phi(n)}\}$$

Chaque élément de x_i de X est donc un entier inférieur à n tel que $\mathrm{PGCD}(n, x_i) = 1$. Maintenant, multiplions chaque élément par $a \pmod n$. On aura l'ensemble Y :

$$Y = \{ax_1 \pmod n, ax_2 \pmod n, \dots, ax_{\phi(n)} \pmod n\}$$

L'ensemble Y n'est autre qu'une permutation de X. En effet :

- a est premier par rapport à n, de même x_i, donc ax_i également. $ax_i \pmod n$ est de ce fait un nombre entier inférieur à n qui lui est premier ;

- Il n'y a pas de duplication dans Y. En effet s'il y a i et j justifiant le fait que $ax_i \equiv ax_j \pmod n$ alors $x_i = x_j$ donc $i = j$;

Ce qui nous emmène à :

$$\prod_{i=1}^{\phi(n)} ax_i \pmod n \equiv \prod_{i=1}^{\phi(n)} x_i$$

$$a^{\phi(n)} \prod_{i=1}^{\phi(n)} x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

D'où :

$a^{\phi(n)} \equiv 1 \pmod n$

♦

2.3.3.4 Test de primalité, algorithme de Miller-Rabbin

Plusieurs méthodes cryptographiques ont besoin de la génération d'un ou plusieurs grands nombres premiers aléatoires. Il est donc nécessaire de pouvoir être en mesure de connaitre si tels ou tels grands nombres générés aléatoirement sont premiers ou non.

Il y a deux types de méthodes. Le premier, probabiliste. En d'autres termes, garantit avec une

marge d'erreur que l'on peut réduire à volonté, qu'un grand nombre a une forte chance d'être premier. Le second, déterministe, confirme qu'un tel nombre est premier ou non.

Dans la pratique, les algorithmes déterministes sont coûteux (en termes de temps d'exécution entre autres), sont donc de ce fait moins bien adoptés que leurs confrères probabilistes.

Par souci de manque de place pour étaler la majorité des tests de primalités existant, nous allons directement détailler une méthode de test de primalité, probabiliste évidemment, qui a prouvé une efficacité concluante (par opposition par exemple au fait que le test de primalité de Fermat présente la faille de déclarer faussement les nombres de Carmichael en tant que premiers).

a. *Quelques propriétés utiles*

Avant d'entamer les détails de cet algorithme, il est nécessaire d'énumérer les quelques notions qui vont suivre. [1.02], [2.05]

Pour tout nombre impair n, on peut écrire :

$$n - 1 = 2^k q \qquad \textbf{(}2.69\textbf{)}$$

Où k est entier et q est impair.

Démonstration :

♣

n est impair, donc $n - 1$ est pair. Par la suite, on pourrait diviser le nombre pair $n - 1$, k fois jusqu'à ce que le résultat ne soit plus divisible par 2, impair en d'autres termes. Ce nombre impair, on pourra le nommer q.

♦

Une première propriété utile des nombres premiers que nous aurons besoin également est que si p est un nombre premier et que a est un nombre entier inférieur à p, alors on a :

$$a^2 \bmod p = 1 \Leftrightarrow a \bmod p = 1 \text{ ou } a \bmod p = -1 \equiv p - 1 \qquad \textbf{(}2.70\textbf{)}$$

Où p est premier et $a < p$.

Démonstration :

♣

D'après la règle de l'opération modulo $a^2 \bmod p = (a \bmod p)(a \bmod p)$ qui est égale à 1 si ses deux identiques facteurs $(a \bmod p)$ sont tous les deux égaux à 1 ou -1.

Et inversement, si $a^2 \bmod p = 1$, alors $(a \bmod p)^2 = 1$. Une égalité qui tient si $(a \bmod p) = 1$ ou $(a \bmod p) = -1$.

♦

La deuxième propriété d'un nombre premier que nous aurons besoin également est que, si p est un nombre premier plus grand que 2, nous pouvons écrire $p - 1 = 2^k q$ avec k positif et q impair. Soit a un nombre entier tel que $1 < a < p - 1$. Alors, une des deux conditions suivantes est vraie :

- a congrue vers $1 \bmod p$. En d'autres termes, $a^q \bmod p = 1$, ce qui équivaut à $a^q \equiv 1 \pmod p$;

- Un des nombres $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}$ congrue vers $-1 \bmod p$. En d'autres termes, il y a un nombre j tel que $1 \leq j \leq k$ tel que $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$. Ce qui équivaut à $a^{2^{j-1}q} \equiv -1 \bmod p$.

Démonstration :

♣

Le théorème de Fermat énonce que $a^{n-1} \equiv 1 \bmod n$ si n est premier. Nous avons $p - 1 = 2^k q$. Ainsi, nous savons que $a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$. Ainsi, si nous regardons les séquences de nombres qui suivent :

$$a^q \bmod p, a^{2q} \bmod p, a^{4q} \bmod p, \ldots, a^{2^{k-1}q} \bmod p, a^{2^k q} \bmod p$$

Nous savons que le dernier de la liste a comme valeur 1. Tous les nombres restants de la liste est le

carré de celui qui le précède. Ainsi, une des deux possibilités ci-dessous doit être vraie :

- Le premier nombre de la liste, et par conséquent, tous les autres restant sont égaux à 1 ;

- Certains nombres de la liste ne sont pas égaux à 1, mais le dernier parmi cette liste aura une valeur égale à 1 quand il est élevé au carré modulo p, 1. Mais selon la propriété des nombres premiers, nous savons que le seul nombre qui satisfait cette condition est $p-1$. Dans ce cas, il est certain que la liste contient un élément égal à $p-1$.

♦

b. *Implémentation de l'algorithme*

Nous avons vu dans le paragraphe précédent que si n est premier, alors soit le premier élément de la liste $\left(a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}, a^{2^k q}\right) \bmod n$ est égal à 1, soit un élément de la liste est égal à $n-1$; sinon n est un nombre composite. D'un autre côté, si un nombre n vérifie une des deux conditions précédemment énumérées, cela ne signifie pas forcément que n est premier. Considérons par exemple le nombre $n = 2047 = 23 \times 89$. $n-1 = 2046 = 2 \times 1023$. On a $2^{1023} \bmod 2047 = 1$, qui justifie une de nos conditions, alors que 2047 n'est pas premier. [1.02], [2.05]

Nous pouvons cependant utiliser le présent algorithme pour diviser le fait qu'un nombre est définitivement composite, ou qu'il pourrait être premier ou non. Dénommons $TEST(n)$ cet algorithme, les étapes correspondantes sont comme suit :

$TEST(n)$:

- Trouver les entiers k et q, avec $k > 0$ et q impair tel que $n-1 = 2^k q$;

- Sélectionner un nombre aléatoire a, tel que $1 < a < n-1$;

- Si $a^q \bmod n = 1$ alors retourner « n pourrait être premier » ;

- Pour $j = 0$ à $k-1$ faire si $a^{2^j q} \bmod n = n-1$ alors retourner « n pourrait être premier » ;

- Retourner « n est composite » ;

Exemple : Essayons de voir si 29 est premier. On a $29-1 = 28 = 2^2 \times 7 = 2^k q$. Premièrement,

essayons $a = 10$. On a $10^7 \bmod 29 = 17$. Ce qui n'est ni 1 ni 28. On continue avec $(10^7)^2 \bmod 29 = 28$. Ce qui nous emmène au résultat que 29 pourrait être premier. En essayant encore avec $a = 2$, on trouve $2^7 \bmod 29 = 12$ et $(2^7)^2 \bmod 29 = 28$. Ce qui nous emmène, encore une fois, au résultat que 29 pourrait être premier. Si nous réitérons le calcul pour tout $1 < a < 28$, nous obtiendrons toujours le résultat comme quoi n pourrait être premier.

Maintenant, essayons le test avec le nombre composite $n = 13 \times 17 = 221$. On a $221 - 1 = 220 = 2^2 \times 55 = 2^k q$. En essayant $a = 5$, on a $5^{55} \bmod 221 = 112$, qui n'est ni 1 ni 220. Maintenant, $(5^{55})^2 \bmod 221 = 168$. Parce que nous avons épuisé toutes les valeurs de j, $0 \leq j \leq 1$, notre algorithme $TEST(n)$ retourne dans sa dernière étape « n pourrait être composite ». Mais supposons que nous avons commencé par $a = 21$. Nous avions eu $21^{55} \bmod 221 = 200$ et $(21^{55})^2 \bmod 221 = 220$, l'algorithme aurait retourné « n pourrait être premier ». En effet, parmi les 228 choix possibles du nombre a (a allant de 2 à 219), 4 d'entre eux retournent le même résultat : « n pourrait être premier ».

c. *Usage répété de l'algorithme de Miller-Rabin*

L'algorithme de Miller-Rabin est une méthode à approche probabiliste, comme nous l'avons déjà mentionné un peu plus haut. Il est avancé dans la littérature que cet algorithme a une probabilité inférieure à ¼ de ne pas réussir à détecter qu'un nombre composite l'est (retourne en d'autres termes un résultat comme quoi le nombre pourrait être premier). Répéter l'algorithme de Miller-Rabin t fois revient donc à réduire à $(1/4)^t$ la probabilité de cette erreur. [1.02], [2.05]

Procéder donc comme ce qui va suivre. Pour un nombre impair n donné, choisir aléatoirement un nombre $1 < a < n - 1$. Si $TEST(n)$ retourne « composite » alors n est composite. Si par contre l'algorithme retourne « n pourrait être premier », choisir une autre valeur de a. Si après un nombre assez élevé de t tests l'algorithme continue à retourner « n pourrait être premier », assumer que n est premier.

La démonstration de ce qui a été annoncé précédemment (celle qui avance la valeur ¼ comme borne supérieure de la probabilité d'erreur de l'algorithme de Miller-Rabin, pour rappel) s'avère être assez longue. Nous renvoyons ainsi nos lecteurs dans notre bibliographie [2.05], si besoin est.

2.4 Conclusion

Ce chapitre nous a permis de s'étaler sur des notions mathématiques indispensables à la cryptographie. Il s'est divisé en deux grandes parties. Celle relative à la théorie de l'information dans un premier temps. La seconde partie en est la théorie des nombres et la mathématique discrète.

Dans la théorie de l'information, nous avons vu entre autres les deux principaux théorèmes de Shannon : celui du codage source et du codage canal. Le premier stipule que le taux du code ne devrait être inférieur à l'entropie de la source, sinon l'on perdrait l'information. Le second affirme que pour un niveau de bruit donné, il existe un débit suivant lequel, on pourrait transmettre un message dans le canal.

Dans la seconde partie qu'est la théorie des nombres et la mathématique discrète, nous avons remis en exergue des notions d'arithmétiques modulaires comme quoi les propriétés (commutativité, associativité, distributivité, identité et inverse) que nous connaissons avec les opérations standards $(+, -\text{ et } \times)$ tiennent également dans l'ensemble des restes de la division par N, l'ensemble $\mathbb{Z}_N$. Nous avons également développé la méthode de Miller-Rabin qui est une approche probabiliste en termes de confirmation ou non si un nombre n donné est premier.

CHAPITRE 3
LATTICE, UN DES FONDEMENTS MATHEMATIQUES DES CRYPTOSYSTEMES POST-QUANTIQUES

3.1 Introduction

Les cryptosystèmes du monde moderne, de l'ère informatique pour le dire d'une autre manière, sont basés sur des problèmes mathématiques difficiles à résoudre. Nécessitant un temps de calcul non négligeable, même pour un ordinateur assez puissant. Parmi ces problèmes mathématiques, on peut citer : [1.01]

- La factorisation d'un grand nombre entier. Produit de deux grands nombres premiers pour le cas du cryptosystème *RSA* – des initiales du nom de leurs inventeurs *Rivest, Shamir* et *Adleman* ;

- Le problème lié au logarithme discret. Celui-ci est l'outil mathématique derrière *l'échange de clés de Diffie-Hellman*, la dernière partie du nommage venant toujours du nom de leurs inventeurs *Whitfield Diffie* et *Martin Hellman* ;

Pour le moment, personne n'est parvenu à un méthode facile pour résoudre ces problèmes. Mais rien ne dit que demain, ou dans un futur proche, quelqu'un ne vienne avec un algorithme ou une solution facile pour les surmonter.

Une autre voie, pourrait être qu'une autre famille d'ordinateur soit en train de voir le jour. C'est justement le cas des ordinateurs quantiques, des types d'ordinateurs tout à fait nouveaux, basés sur les phénomènes physiques connus sou le nom de *physique quantique*. En effet, des chercheurs ont déjà pu trouver comment écrire des programmes sur ces types d'ordinateur, même si ceux-ci (ces types d'ordinateurs) n'ont pas encore vu le jour. Ce qui pourrait apporter de sérieux problèmes aux cryptosystèmes en cours d'usage en ce moment, quand le tant mentionné jour viendra.

La plus importante propriété de la physique quantique est la *superposition*. Soit une particule quantique peut avoir deux états en même temps. Nous allons nous attarder sur le sujet (physique quantique et ordinateur quantique) dans un de nos paragraphes qui vont suivre.

Nous allons également détailler les algorithmes qui peuvent venir à bout des cryptosystèmes en cours d'usage en ce moment, notamment ceux de la famille des deux cryptosystèmes sus-cités.

Mais la plus grande partie de notre présent chapitre sera axée sur les théories mathématiques autour

de la *lattice* (nous décidons d'utiliser le terme lattice, même si c'est un mot plutôt anglais. Les recherches bibliographiques sur google montrent par exemple des résultats plus conséquents en utilisant le terme *« lattice based cryptography »*, que son homologue français *« cryptographie à base de treillis »*). Cet objet mathématique se trouve être en effet, selon notre bibliographie, un type de problèmes qui peut résister aux ordinateurs quantiques. D'où l'intérêt que nous y portons au cours de cette recherche.

3.2 Ordinateur quantique, cryptographie quantique et post-quantique

3.2.1 Ordinateur quantique

Comme déjà mentionné dans notre introduction, les ordinateurs quantiques utilisent la physique quantique pour calculer d'une manière différente ce que les ordinateurs classiques ne peuvent faire. Ils ne sont donc pas des ordinateurs classiques très rapides. [1.01], [3.01]

Les ordinateurs quantiques sont basés sur la mécanique quantique, une branche de la physique qui étudie le comportement des particules subatomiques. Ces dernières se comportent d'une manière réellement aléatoire. Contrairement aux ordinateurs classiques, qui opèrent sur des *bits* qui sont soit 0 soit 1, les ordinateurs quantiques sont basés sur des *bits quantiques* (ou *qubits*) qui peuvent être simultanément 0 et 1, ce phénomène est connu sous le nom de *superposition*. En effet, dans le monde microscopique, avant que l'on observe l'électron, il ne se trouve pas à un endroit fixe, mais en plusieurs endroits en même temps (c'est la superposition que nous avons vue plutôt). Mais une fois qu'on l'observe – une opération appelée *mesure quantique* – alors il (l'électron) se place à un endroit fixe et ne bouge plus. C'est cette magie qui a permis de parvenir à la création des qubits des ordinateurs quantiques.

Mais si ces-dits qubits quantiques fonctionnent comme ils le sont, c'est également à cause du phénomène connu sous le nom d'*intrication quantique* (*quantum entanglement* en anglais). Deux particules peuvent être connectées de telle sorte que si l'on connaît la valeur de l'une, cela nous emmène à savoir la valeur de l'autre. Cette propriété tient indépendamment de la distance qui sépare les deux particules. Ce comportement est décrit par le *paradoxe EPR* (du nom de leurs inventeurs *Einstein-Podolsky-Rosen*).

Les deux prochaines sections de notre ouvrage détailleront les qubits (l'élément qui compose la partie hardware de l'ordinateur quantique) et les *portes quantiques* ou *quantum gates* en anglais

(l'élément qui compose le software de l'ordinateur quantique).

3.2.1.1 Bits quantiques ou qubits

Les bits quantiques sont caractérisés par des nombres appelés *amplitudes*. Un amplitude est un nombre complexe de la forme $a + b \times i$ ou tout simplement $a + bi$. Où a et b sont des nombres réels et i représente l'unité imaginaire, $i^2 = -1$.

Le théorème de Pythagore nous permet de calculer la longueur ou *module* du nombre complexe. Cette grandeur sera dénotée $|a + bi|$:

$$|a + bi| = \sqrt{a^2 + b^2} \quad \textbf{(}3.01\textbf{)}$$

Pour un ordinateur quantique, les enregistrements sont constitués d'un ou plusieurs qubits dans un état de superposition, composés de tel ensemble d'amplitudes. Mais nous allons voir par la suite que ces nombres complexes – amplitudes – ne peuvent être n'importe quel nombre.

a. *Amplitude d'un simple qubit*

Un simple qubit Ψ est constitué de deux amplitudes, que nous nommerons α (alpha) et β (beta). Nous pouvons exprimer l'état du qubit par $\alpha\,|0\rangle + \beta\,|1\rangle$ où le "$|\,\rangle$" est utilisé pour dénoter les vecteurs dans un état quantique. Cette notation signifie que quand on observe le qubit, celui s'affiche en tant que $|0\rangle$ pour une probabilité égale à $|\alpha|^2$ et $|1\rangle$ pour une probabilité égale à $|\beta|^2$. En effet, pour que la loi de la probabilité tienne, il faut que $|\alpha|^2$ et $|\beta|^2$ soient des nombres entre 0 et 1 et que $|\alpha|^2 + |\beta|^2 = 1$. [3.01]

$$\Psi = \alpha\,|0\rangle + \beta\,|1\rangle,\ |\alpha|^2 + |\beta|^2 = 1 \quad \textbf{(}3.02\textbf{)}$$

Exemple : Soit dans un premier temps le qubit $\Psi = 1/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle$. La probabilité d'observer Ψ en tant que $|0\rangle$ est de $|\alpha|^2 = \left(\sqrt{\left(1/\sqrt{2}\right)^2}\right)^2 = 1/2$.

Soit par la suite le qubit $\Phi = i/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle$, que l'on pourra également noter $|\Phi\rangle =$

$\left(i/\sqrt{2}\ , -1/\sqrt{2}\ \right)$. La chance pour voir Φ en tant que $|0\rangle$ est donc de $|\alpha|^2 = \left(\sqrt{\left(1/\sqrt{2}\right)^2}\right)^2 = 1/2$.

Remarquons que les qubits Ψ et Φ ont la même chance d'être observés en tant que $|0\rangle$. Mais ils ne représentent pas pour autant les mêmes qubits (ce sont deux différents qubits). Ce qui nous emmène au fait que les probabilités de pouvoir observer un qubit en tant que $|0\rangle$ ou $|1\rangle$ ne caractérisent que partiellement le qubit. Pour les qubits, une dimension cachée est renfermée dans leurs amplitudes.

b. *Amplitude d'un ensemble de qubits*

Pour un ensemble de qubits, par exemple un *octet quantique*, formé de 8 qubits, quand placés dans un état quantique dans lequel chacun des 8 qubits est lié entre eux (nous avons parlé du phénomène quantique d'intrication). Un tel octet quantique peut être décrit comme suit, où les αs sont les amplitudes associés à chacun de ses 256 possibles valeurs :

$$\Psi = \alpha_0\,|00000000\rangle + \alpha_1\,|00000001\rangle + \alpha_2\,|00000010\rangle + \alpha_3\,|00000011\rangle + \cdots + \alpha_{255}\,|11111111\rangle,\ |\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 + \cdots + |\alpha_{255}|^2 = 1 \quad \textbf{(}3.03\textbf{)}$$

Notre groupe de 8 qubits équivaut donc à $2^8 = 256$ amplitudes, du fait qu'il peut avoir 256 configurations, chacune d'entre elles ayant sa propre amplitude. En réalité cependant, nous avons uniquement 8 objets physiques et non pas 256.

En généralisant, un groupe de n qubits est caractérisé par un ensemble de 2^n nombres complexes. Un nombre qui augmente exponentiellement au fur et à mesure que n augmente. C'est cette propriété qui fait que les ordinateurs classiques ne peuvent simuler les ordinateurs quantiques. Pour en être capable, ils (les ordinateurs classiques) doivent posséder un volume conséquent de mémoires (de l'ordre de 2^n) pour sauvegarder le même volume d'informations contenu dans seulement n qubits.

3.2.1.2 Quantum gates ou portes quantiques

Le concept d'amplitude et de portes quantiques sont spécifiques aux ordinateurs quantiques. Si les ordinateurs quantiques utilisent des registres, de la mémoire et des microprocesseurs pour effectuer des séquences d'instructions sur des données, les ordinateurs quantiques transforment un groupe de

qubits d'une manière réversible en appliquant une série de portes quantiques. Et par la suite mesure la valeur d'un ou plusieurs qubits. Les ordinateurs quantiques de ce fait apportent plus de puissance de calcul dû au fait qu'avec seulement n qubits, ils peuvent manipuler 2^n nombres (les amplitudes des qubits).

D'un point de vue mathématique, les algorithmes quantiques sont essentiellement un circuit de portes quantiques qui transforme un ensemble de nombres quantiques (les amplitudes) avant la mesure finale où la valeur de 1 ou plusieurs qubits est observée. Se référer à la **Figure 3.01**.

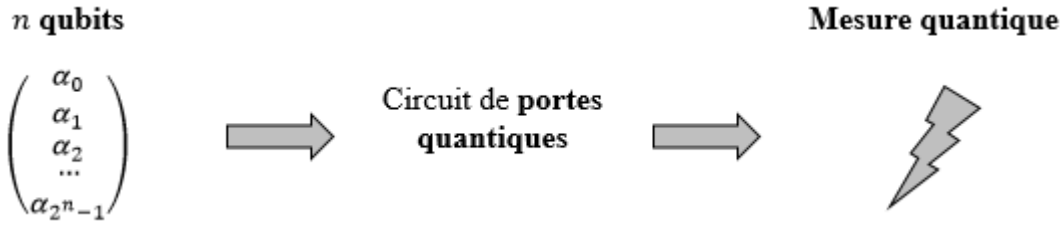


Figure 3.01 : *Principe d'un algorithme quantique.*

a. *Portes quantiques comme multiplication dans les matrices*

Tout comme les portes booléennes des ordinateurs classiques (AND, XOR, …), une porte quantique agit sur un groupe d'amplitudes tout comme une matrice se comporte quand celle-ci est multipliée par un vecteur. Dans l'exemple qui va suivre, nous allons voir l'effet de la plus simple des portes quantique, la matrice identité $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, quand celle-ci est appliquée aux amplitudes du qubit $|\Phi\rangle = \left(i/\sqrt{2}, -1/\sqrt{2}\right)$:

$$I|\Phi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} i/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1 \times i/\sqrt{2} + 0 \times \left(-1/\sqrt{2}\right) \\ 0 \times i/\sqrt{2} + 1 \times \left(-1/\sqrt{2}\right) \end{pmatrix} = \begin{pmatrix} i/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} = |\Phi\rangle ;$$

Notons que dans la pratique un ordinateur quantique ne va pas effectuer la multiplication matrice-vecteur. En effet, les matrices seront trop grandes à manipuler. C'est pour cela d'ailleurs que les ordinateurs classiques ne peuvent simuler les ordinateurs quantiques. Au lieu de cela, les ordinateurs quantiques transforment les qubits en tant que particules physiques par le biais de transformations physiques qui sont équivalentes à la multiplication matricielle demandée.

b. *Porte quantique d'Hadamard*

La porte quantique d'Hadamard est sans doute l'une des plus utilisées. Elle est souvent notée *H* :

$$\mathrm{H} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \quad \textbf{(}3.04\textbf{)}$$

Voyons ce qui se passe lorsque l'on applique cette porte quantique au qubit $|\Phi\rangle = \left(1/\sqrt{2}, 1/\sqrt{2}\right)$:

$$H|\Phi\rangle = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1/2 + 1/2 \\ 1/2 - 1/2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \, ;$$

En appliquant la porte d'Hadamard au qubit $|\Phi\rangle$, nous obtenons le qubit $|0\rangle$, pour lequel la valeur $|0\rangle$ a une amplitude égale à 1 et la valeur $|1\rangle$ une amplitude 0. Ceci nous dit que le qubit résultant va se comporter d'une manière déterministique : nous allons toujours le voir en tant que $|0\rangle$ et jamais $|1\rangle$. En d'autres termes, la caractéristique aléatoire du qubit $|\Phi\rangle$ a été perdue.

Qu'est ce qui va se passer si nous réappliquons la porte d'Hadamard au qubit $|0\rangle$?

$$H|0\rangle = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} + 0 \\ 1/\sqrt{2} + 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = |\Phi\rangle \, ;$$

Cela nous ramène à notre qubit initial $|\Phi\rangle$ et le fait que celui-ci se comporte d'une manière aléatoire. En effet, la porte d'Hadamard est souvent utilisée dans les algorithmes quantiques pour passer d'un état déterministe à un état uniformément aléatoire.

c. *Portes quantiques, des matrices unitaires*

Ce ne sont pas toutes les matrices qui sont des portes quantiques. Il existe une propriété que celles-ci (les matrices) doivent vérifier pour en être une. A titre de rappel, un qubit est constitué de deux amplitudes α et β, vérifiant la condition $|\alpha|^2 + |\beta|^2 = 1$. Si la matrice porte quantique, après multiplication par notre qubit initial, mène à un vecteur dont les amplitudes ne vérifient pas la condition précédemment énoncée ($|\alpha|^2 + |\beta|^2 = 1$), cette matrice-là ne peut être considérée comme porte quantique. Les matrices qui peuvent effectuer de telle transformation sont dites *matrices unitaires*.

Les matrices unitaires (et donc les portes quantiques par définition), sont des matrices inversibles. En effet, une matrice est dite unitaire si son inverse n'est autre que sa *matrice transposée conjuguée*. On pourra donc calculer le qubit original en utilisant la *matrice inverse*.

3.2.1.3 Accélération quantique ou quantum speedup en anglais

a. *Accélération quadratique dans le cas de la recherche d'un élément parmi n éléments donnés*

Une accélération quantique ou *quantum speedup* en anglais apparait lorsque l'on peut résoudre un problème donné avec un ordinateur quantique en un temps moindre que si la même opération ait été effectuée avec un ordinateur classique. Prenons l'exemple du cas de la recherche d'un élément parmi n éléments donnés, non arrangés. On a en moyenne besoin de $n/2$ opérations pour trouver l'élément recherché sur un ordinateur classique. En effet, on a besoin de tester en moyenne la moitié des n éléments avant de pouvoir trouver celui que l'on cherche. Avec un ordinateur quantique cependant, il existe un algorithme [3.01] pour effectuer une telle recherche en seulement $\sqrt{n}$ opération, qui est plus petit que $n/2$. Imaginons un ensemble de $n = 1000000$ d'éléments, l'ordinateur classique aurait besoin de $n/2 = 500000$ opérations, si seulement $\sqrt{n} = 1000$ est nécessaire pour un ordinateur quantique.

Il est important de quantifier la différence en termes de *complexité en temps* (*time complexity* en anglais) des algorithmes classiques et quantiques, d'habitude dénotée par $O()$. Dans notre exemple précédent, l'algorithme classique s'exécute en un temps $O(n)$, si leur équivalent quantiques ont besoin de $O(\sqrt{n})$. La différence étant exprimée en termes de carré (ou de racine carrée, c'est selon), on parle d'*accélération quadratique* (*quadratic speedup* en anglais). Mais ayons en tête qu'il y a encore d'autres accélérations quantiques plus puissantes.

b. *Accélération quadratique et problème de Simon (Simon's problem en anglais)*

L'*accélération exponentielle* est le saint graal des ordinateurs quantiques. Elle se produit lorsqu'un problème que les ordinateurs classiques mettent un temps exponentiel à résoudre, de l'ordre de $O(2^n)$, si de l'autre côté les ordinateurs quantiques n'en demande qu'un temps polynomial, genre $O(n^k)$ pour un k donné. Cette accélération exponentielle peut rendre un problème pratiquement impossible à résoudre, en un problème qui peut être résolu dans la pratique. Notons que la littérature associe au temps de résolution exponentielle à l'impossible, et à celui polynomial à pratique.

Le *problème de Simon* est l'un des plus connus pour illustrer l'accélération exponentielle. Il (le problème de Simon) consiste en une fonction f qui transforme un text de n-bits en un autre de n-bits. La sortie parait parfaitement aléatoire, sauf que pour tout y tel que $f(x) = f(y)$, il existe une valeur m qui justifie $y = x \oplus m$. Pour résoudre ce problème, il faut trouver m.

Avec un algorithme classique, résoudre le problème de Simon revient à retrouver la paire d'entrée (x, y) qui renvoie à une collision à la sortie de la fonction f. Cette méthode consiste à appliquer en moyenne à la fonction f, $2^{n/2}$ requêtes avant de trouver le couple (x, y). Avec un ordinateur quantique cependant, l'algorithme illustré par la **Figure 3.02**, peut trouver la solution en ne soumettant à f que n requêtes seulement.

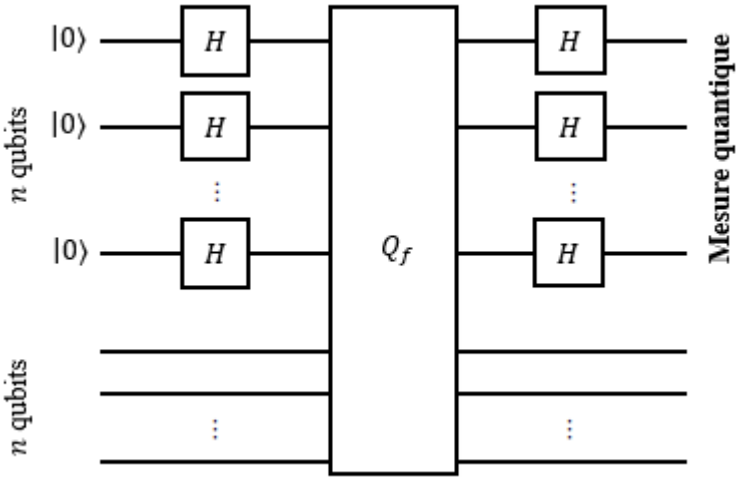


Figure 3.02 : *Circuit d'un algorithme quantique pouvant résoudre efficacement le problème de Simon*

On initialise $2n$ qubits et applique aux n qubits les moins significatifs, la porte d'Hadamard pour leur donner un état complètement aléatoire. On applique par la suite aux $2n$ qubits ainsi obtenus la porte Q_f. Cette dernière (la porte Q_f), est une porte construite à base de la fonction f. Etant donné les 2 groupes de n qubits, x et y, la porte Q_f transforme, par définition, l'état quantique $|x\rangle|y\rangle$ en $|x\rangle|f(x) \oplus y\rangle$. Notons que la porte Q_f justifie la propriété de réversibilité que nous avons énoncée dans notre paragraphe 3.2.1.2 c. En effet, on pourra revenir à notre $2n$ qubits en entrée $|x\rangle|y\rangle$ en réappliquant la porte Q_f à la sortie obtenue, $|x\rangle|f(x) \oplus y\rangle$: $|x\rangle|f(x) \oplus f(x) \oplus y\rangle = |x\rangle|0^n \oplus y\rangle = |x\rangle|y\rangle$. Où $|0^n\rangle$ représente la succession de n qubits égaux à $|0\rangle$: $|0^2\rangle = |00\rangle$ pour $n = 2$.

Pour plus de détails sur comment l'algorithme de Simon aide à la détection de la fonction f, nous renvoyons nos lecteurs à nos bibliographies [3.02]. [3.03]. L'objectif du présent chapitre étant majoritairement de s'étaler sur les théories autour des objets mathématiques lattices. Mais bien sûr de donner aussi dans un premier temps une initiation aux ordinateurs quantiques, et sur le thème comment de tels types d'ordinateurs peuvent venir à bout des cryptosystèmes actuellement existants.

L'accélération exponentielle que nous apporte l'algorithme de Simon peut être utilisé comme outil de cryptanalyse à l'endroit de certains chiffres de substitution bien spécifiques. Dans les paragraphes qui vont suivre, nous allons voir des algorithmes quantiques qui sont plus destructeurs à l'endroit des cryptosystèmes de renoms actuels.

c. *Menace qu'apporte l'algorithme de Shor*

L'algorithme de Shor est un algorithme quantique qui apporte une accélération quantique exponentielle lors de la résolution de problèmes mathématiques tels la factorisation, le logarithme discret (*DLP* ou *Discrete Logarithm Problem* en anglais) et le logarithme discret courbe elliptique (*ECDLP* ou l'*Elliptic Curve DLP*). Ces problèmes ne peuvent être résolus par un ordinateur classique, s'ils peuvent l'être par un ordinateur quantique. Ceci dit, on peut donc avec un ordinateur quantique casser tout cryptage basé sur le RSA, Diffie-Hellman, cryptosystème à base de courbes elliptiques, ou encore tout cryptosystème à clé publique/privée.

L'algorithme de Shor résout un problème plus général que ceux mentionnés dans le paragraphe précédent. En effet, si une fonction f est périodique, en d'autres termes, s'il y a un ω de telle sorte que $f(x+\omega) = f(x)$ pour tout x donné, l'algorithme de Shor permet de trouver facilement ω. On pourrait voir une similitude dans la formulation de la propriété de la fonction f pour l'algorithme de Shor et celui de Simon. En effet Shor se serait basé sur le travail de Simon pour l'élaboration de son algorithme.

d. *Algorithme de Shor et problème de factorisation*

Supposons que nous voulons factoriser un grand nombre $N = pq$. C'est facile de factoriser le nombre N si nous pouvons trouver la période de $a^x \bmod N$, une tâche qui est facile à faire avec un ordinateur quantique, si elle est non faisable dans la pratique pour un ordinateur classique. On prend un nombre aléatoire $a < N$ et par la suite utiliser l'algorithme de Shor pour trouver la période de la fonction $f = a^x \bmod N$. Une fois que cette période est trouvée, nous aurons $a^x \bmod N =$

$a^{x+\omega} \bmod N = a^x a^\omega \bmod N$ soit encore $a^\omega \bmod N = 1$ ou $a^\omega - 1 \bmod N = 0$. En d'autres $a^\omega - 1$ est multiple de N soit $a^\omega - 1 = kN$ pour un k donné.

$a^\omega - 1$ peut par la suite être formulé en tant que produit de deux termes. $a^\omega - 1 = \left(a^{\omega/2} - 1\right)\left(a^{\omega/2} + 1\right)$. On peut par la suite calculer le PGCD de $a^{\omega/2} - 1$ et N et voir si l'on a obtenu une valeur non trivial (ni 1 ni N) des facteurs de N. Sinon, on peut tout simplement changer la valeur de a et recommencer l'algorithme. Après quelques essais, on est en mesure d'obtenir la clé privée RSA à partir de la clé publique, celle qui permet de décrypter le message.

Notons que les meilleurs algorithmes classiques pour factoriser le nombre N s'exécute en temps exponentiel de n, où n étant la longueur en nombre de bits de N : $n = \log_2 N$. L'algorithme de Shor cependant, s'exécute en temps polynomial en n, soit $O\left(n^2 (\log n)(\log\log n)\right)$.

e. *Algorithme de Shor et logarithme discret*

Le défi derrière le logarithme discret est de trouver y tel que $y = g^x \bmod p$, pour g et p connus. Trouver la solution à ce problème, encore une fois, nécessite un temps colossal sur un ordinateur classique, s'il se résout en temps convenable sur un ordinateur quantique.

Soit par exemple la fonction $f(a, b) = g^a y^b$. Supposons que nous aimerions trouver les périodes de cette fonction, les nombres ω et ω', de telle sorte que $f(a, b) = f(a + \omega, b + \omega')$ pour tout nombre a et b. La solution que nous recherchons est par la suite $x = -\omega/\omega' \bmod q$, l'exposant de g qui est un paramètre connu. L'égalité $f(a, b) = f(a + \omega, b + \omega')$ implique $g^\omega y^{\omega'} \bmod p = 1$. En substituant y avec g^x, nous avons $g^{\omega + x\omega'} \bmod p = 1$, ce qui équivaut à dire $\omega + x\omega' \bmod q = 0$, d'où $x = -\omega/\omega'$.

Encore une fois, la complexité de cette algorithme est de $O\left(n^2 (\log n)(\log\log n)\right)$ avec n le nombre de bits de p. Notons que cet algorithme se généralise pour tout problème de logarithme discret dans n'importe quel groupe commutatif, pas seulement le groupe des nombres modulo un nombre premier.

f. *Algorithme de Grover*

L'algorithme de Grover est l'algorithme quantique derrière l'accélération quadratique que nous avons introduit dans notre paragraphe plus haut 3.2.1.3 a. Le présent paragraphe détaillera l'impact

qu'aura cet algorithme sur les cryptosystèmes actuels. Il montrera également le fait qu'il est possible de sauver les cryptosystèmes symétriques, ce en doublant la longueur de la clé. Les cryptosystèmes asymétriques sont par contre détruits pour de bon. [3.01]

Imaginons l'algorithme de Grover comme étant un moyen de trouver la valeur x parmi n valeurs possibles, tels que $f(x) = 1$. La fonction f prenant la valeur $f(x) = 0$ la plupart du temps. Si m valeurs de x justifient $f(x) = 1$ alors l'algorithme de Grover peut trouver une solution en $O\left(\sqrt{n/m}\right)$. A titre de comparaison, les algorithmes classiques requièrent $O(n/m)$.

Maintenant supposons que la fonction f se définit par : « $f(x) = 1$ si et seulement si x est égal à la clé secrète K tel que $E(K, P) = C$ ». Où E représente un chiffre (une méthode cryptographique) donné et P et C sont respectivement les textes en clair et chiffrés. Dans la pratique, cela signifie que si nous sommes à la recherche d'une clé AES 128 bits, un ordinateur quantique sera en mesure de la trouver en un temps proportionnel à 2^{64}. Si 2^{128} est nécessaire pour un ordinateur classique. La solution pour retrouver la sécurité de l'ordre des 128 bits, il suffit de doubler la longueur de la clé : à 256 bits donc. Cela retournera le temps nécessaire aux ordinateurs quantiques de trouver la clé à : $2^{256/2} = 2^{128}$.

3.2.1.4 Construire un ordinateur quantique, une tâche difficile

Il est difficile de construire un ordinateur quantique du fait que l'on ait besoin d'éléments extrêmement petits, de l'ordre des électrons et photons, pour jouer le rôle de qubits. Et le fait d'être si petits les (les qubits) rend fragiles.

En addition, il est nécessaire de tenir les qubits à une température proche du 0 absolu, pour les rendre stables. Or, même à de telle température, les qubits se détériorent et deviennent inutilisables.

Un autre défi à mentionner est que l'environnement (chaleurs et champs magnétiques entre autres) dans lequel opèrent les qubits peut avoir des effets néfastes sur eux. On peut citer la génération de bruits qui peuvent conduire à des erreurs de calcul. Pour corriger ces erreurs-là, on fait appel à la notion de *codes quantiques* (ou *quantum code corretion* en anglais), l'équivalent des codes correcteurs du monde des ordinateurs classiques. On a besoin de qubits supplémentaires pour supporter cette tâche de correction.

Il existe actuellement deux pistes en cours pour la construction d'ordinateur quantique :

- Celle à base de *circuits superconducteurs* (*superconducting circuits* en anglais), recherche menée majoritairement par les laboratoires de *Google* et *IBM*. Les qubits sont formés par de petits circuits électriques qui utilisent les phénomènes de la physique quantique des matériaux superconducteurs. Les conducteurs de charge étant un pair d'électrons.

- *Les pièges à ions* (ou *ions trap* en anglais), sont faits à base d'ions (atomes chargés). Ces derniers (les ions) sont manipulés à l'aide de lasers pour préparer les qubits dans un état de départ désiré. Les pièges à ions semblent plus stables que leurs équivalents en circuits superconducteurs, mais ils sont difficiles à produire à grande échelle et sont moins rapides à manipuler.

3.2.2 Algorithme cryptographique post-quantique

Le domaine de la cryptographie post-quantique se réfère aux études des algorithmes cryptographiques résistants aux ordinateurs quantiques. Des candidats aptes à remplacer les cryptosystèmes tels RSA ou encore ceux à base de logarithme discret.

3.2.2.1 Groupes de cryptographies post-quantiques

On discerne principalement 2 groupes de cryptographies post-quantiques :

- Ceux relatifs aux *PKE/KEM* ou *Public Key Encryption/Key Encapsulation Mechanism* dans un premier temps, pour leur nom en version anglaise. En français, on parlera de (1) *Cryptographie à Clé Publique* pour PKE. Notons en passant que l'on a omis le terme « *privée (clé)* ». En effet, on parle, pour être succinct, de cryptographie à clé publique/privée. Le chiffrement s'effectue avec la clé publique, si le déchiffrement s'effectue en clé privée. (2) Le KEM quant à lui peut être traduit par *Mécanisme d'Encapsulation de Clé*. Cryptographie à clé publique/privée toujours, la clé publique est utilisée pour créer un texte chiffré pour encapsuler une clé symétrique privée. Le déchiffrement se fait par clé privée.

- Ceux qui se concentrent autour du *Digital Signature* pour son nom en anglais (*Signature Numérique* en français). La signature numérique permet au destinataire de s'assurer que le message provient bien d'un envoyeur connu (qu'il est authentique), et que le message n'ait pas été modifié en cours de route (qu'il est intègre).

Dans les paragraphes qui vont suivre, nous allons énumérer le top 3 des objets mathématiques sur lesquels se basent la majorité des candidats à la cryptographie post-quantiques. On distingue notamment les lattices, les codes correcteurs d'erreurs et les systèmes d'équations multivariées.

3.2.2.2 Cryptographie à base de treillis ou lattice-based cryptography en anglais

Les lattices sont des objets mathématiques, géométriques, qui peuvent être décrits comme étant les intersections d'une grille (non nécessairement orthogonales) de dimension n, régulière et infinie. Malgré leur simplicité apparente, elles (les lattices) renferment de riches structures combinatoires, qui ont attiré l'attention de nombreux mathématiciens de renoms. Aussi, il s'ensuit que les lattices ont trouvé de nombreuses applications en mathématiques et en informatique, pour ne citer que le présent sujet sur lequel on base notre recherche : la cryptographie.

L'étude sur les lattices, en terme de complexité de calcul, a été marquée par deux avancées majeures : (1) le développement de l'algorithme de réduction *LLL* du nom de leur créateur *Lenstra–Lenstra–Lovász* (de leur nom complet Arjen Lenstra, Hendrik Lenstra et László Lovász) dans le début des années 1980, et (2) la découverte d'*Ajtai* sur l'existence de relation entre les *pires des cas* (*worst case* en anglais) et les *cas moyens* (*average case* en anglais), en termes de complexité, ce dans la fin des années 1990.

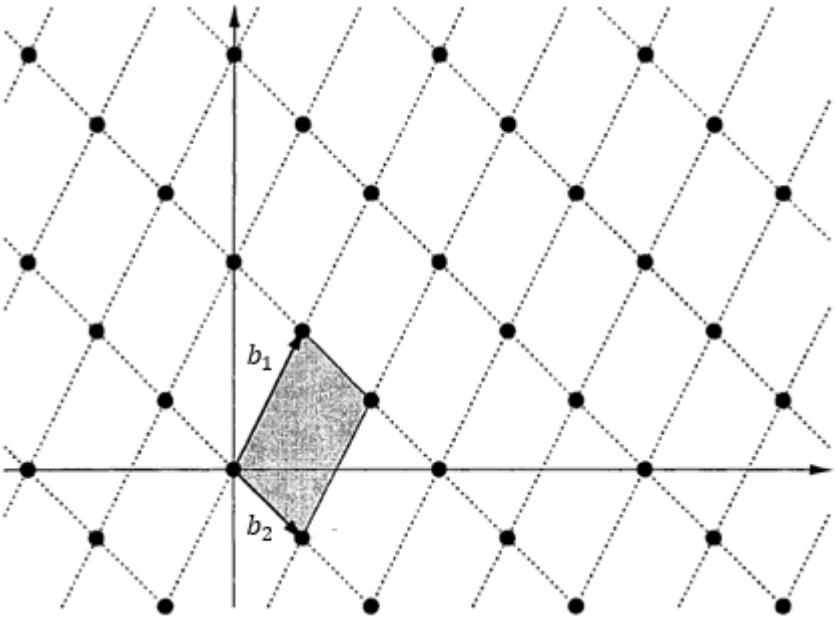


Figure 3.03 : *Une Lattice dans* $\mathbb{R}^2$ *où* $\boldsymbol{b_1}$ *et* $\boldsymbol{b_2}$ *en sont les vecteurs de bases.*

Dans la dimension $n = 2$ par exemple, une lattice se présente comme le montre la **Figure 3.03**. Dans cette exemple, on a pris $\boldsymbol{b_1} = \begin{bmatrix}1\\2\end{bmatrix}$ et $\boldsymbol{b_1} = \begin{bmatrix}1\\-1\end{bmatrix}$. Les lattices faisant le sujet principal de notre présent chapitre, nous allons nous en réserver les détails aux paragraphes vus plus loin, paragraphes qui lui seront dédiés.

3.2.2.3 Cryptographie à base de codes ou code-based cryptography en anglais

a. *Généralités sur les cryptosystèmes à base de codes*

Les algorithmes cryptographiques post-quantiques à base de codes se fondent sur des *codes correcteurs d'erreurs* (*error correcting codes* en anglais). Ces derniers (codes correcteurs d'erreur) ont été conçus pour transmettre des données sur un canal sujet à du bruit. Les théories sur les codes correcteurs d'erreurs ont débuté vers 1950. Le premier algorithme de chiffrement basé sur le code est celui de *McEliece*, développé en 1978, et s'avère jusqu'à aujourd'hui assez résistant, moyennant un accroissement raisonnable de la longueur de la clé de chiffrement.

Les cryptosystèmes à base de codes peuvent être utilisés pour le chiffrement proprement dit, et également pour la signature. En effet, des algorithmes cryptographiques ont été soumis à standardisation (projet de standardisation des algorithmes post-quantiques par *NIST* ou *National Institute of Standards and Technology*) dans les deux catégories (PKE/KEM pour le chiffrement proprement dit, et Signature pour la signature comme l'indique son nom). Cependant, seule une partie de ceux relatifs à la première catégorie a été retenue dans la seconde étape de sélection. [3.01], [3.06]

b. *Cryptosystème de McEliece comme exemple*

Le cryptosystème de McEliece consiste en trois algorithmes : un algorithme probabiliste de génération de clé publique et privée, un algorithme probabiliste de chiffrement, et vers la fin un algorithme déterministe de déchiffrement.

Tous les utilisateurs du système McEliece sont en possession de trois principaux paramètres : n, k et t. [3.07]

i. Génération de clé

- Alice sélectionne un code linéaire (n, k) binaire C ayant la capacité de corriger t erreurs. Ce code doit avoir un algorithme de déchiffrement efficace. Une matrice génératrice G de dimension $k \times n$, relative au code C est disponible en d'autres termes. Dans sa version d'origine, l'algorithme utilise des *codes Goppa binaires* ou *binary Goppa code* en anglais).

- Alice choisit une matrice aléatoire non singulière (inversible) S de dimension $k \times k$.

- Alice choisit une matrice aléatoire de permutation P, la dimension en est $n \times n$.

- Alice calcule la matrice $\hat{G} = SGP$;

- La clé publique d'Alice est de $(\hat{G}, t)$, si sa clé privée est (S, G, P) ;

ii. Chiffrement

Supposons que Bob aimerait envoyer un message à Alice dont la clé publique est $(\hat{G}, t)$:

- Bob encode son message m en tant que chaîne binaire de longueur k ;

- Bob calcule le vecteur $c' = m\hat{G}$;

- Bob génère un vecteur aléatoire z de longueur n contenant exactement t bits 1 (un vecteur de longueur n et de poids t) ;

- Bob calcule le texte chiffré $c = c' + z$;

iii. Déchiffrement du message

Après réception du message c, Alice effectue le déchiffrement comme suit :

- Alice calcule l'inverse P^{-1} de P ;

- Alice calcule $\hat{c} = cP^{-1}$;

- Alice utilise l'algorithme de décodage du code C pour passer de $\hat{c}$ à $\hat{m}$;

- Alice calcule $m = \hat{m}S^{-1}$;

3.2.2.4 Cryptographie multivariée ou multivariate cryptography en anglais

a. *Problème NP-difficile ou NP-hard en anglais*

Il est intéressant de détailler la notion de *problème NP-difficile* (ou *NP-hard* en anglais) étant donné que le présent groupe de cryptographie post-quantique (la cryptographie multivariée) soit classifié comme tel (être un problème NP-difficile en d'autres termes). [3.01], [3.08]

Au plus haut de la classification des théories de complexité se trouve la classe NP-difficile (se référer à la **Figure 3.04**). Cette dernière (la classe NP-difficile) se réfère aux problèmes qui sont tout au moins aussi difficile que les plus difficiles des problèmes dans NP. NP-difficile pourrait donc contenir d'autres classes de problèmes qui n'appartiennent pas à la classe NP.

Vient par la suite la classe dite *NP-complet* (*NP-complete* en anglais). Un problème est dit NP-complet, lorsqu'il est à la fois *NP,* pour *Non déterministe Polynomial* (ou *Non-deterministic Polynomial* en anglais) et NP-difficile.

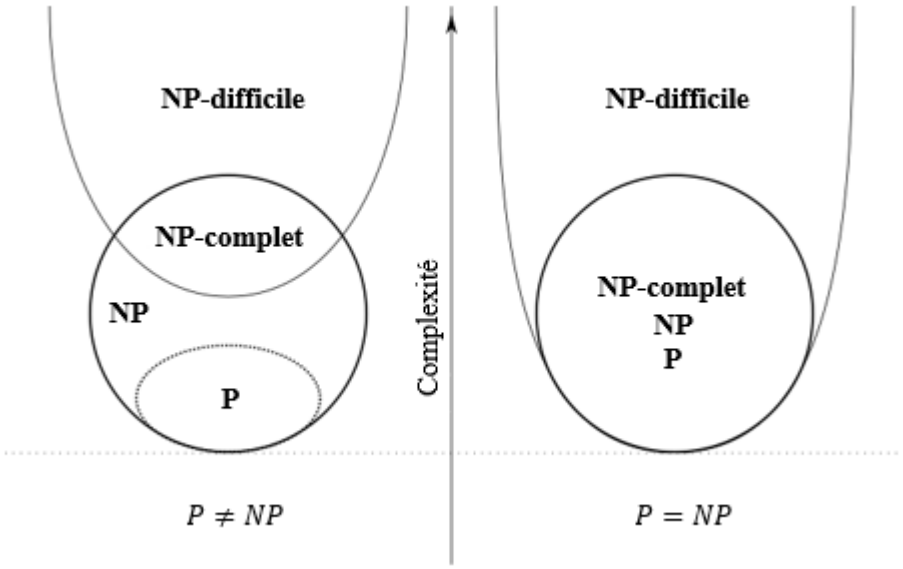


Figure 3.04 : *Diagramme d'Euler pour les classes de problèmes P, NP, NP-complet et NP-*difficile.

NP quant à lui, signifie que l'on peut trouver une solution en un temps acceptable (ayons en tête qu'acceptable/pratique se réfère au temps de résolution polynomial, si impossible revient à un temps de résolution exponentiel) au problème, en utilisant une *machine de Turing non déterministe* (*non deterministic Turing machine* en anglais).

La classe de problème *P* vers la fin, P pour *Polynomial*, représente l'ensemble des problèmes pouvant être résolus en un temps polynomial par une *machine de Turing déterministe*. Une machine de Turing déterministe suggère une seule sortie au plus pour tout état/statut donné. Si son homologue non déterministe offre plus d'une sortie. Se référer à la **Figure 3.05** pour le flux de travail d'une machine de Turing déterministe ou non déterministe.

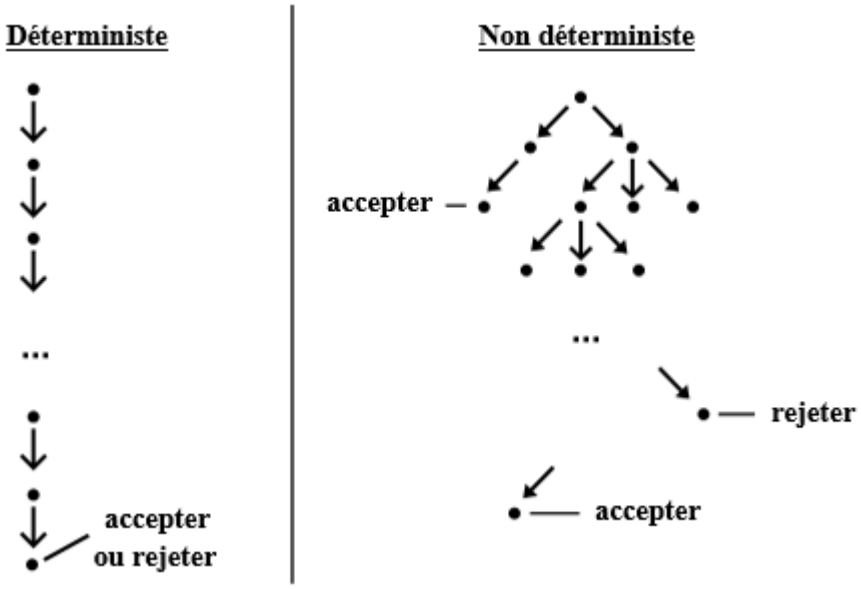


Figure 3.05 : *Machine de Turing déterministe et non déterministe.*

Une propriété importante à citer dans la théorie des complexités est que s'il existe un algorithme polynomial pour résoudre un quelconque des problèmes NP-complets, alors tous les problèmes de la classe NP peuvent être résolus en temps polynomial. Trouver un algorithme polynomial pour un problème NP-complet ou prouver qu'il n'en existe pas permettrait de savoir si $P = NP$ ou $P \neq NP$, une question ouverte qui fait partie des problèmes non résolus en mathématiques les plus importants à ce jour. [3.09]

b. *Généralités sur les cryptographies multivariées*

Les cryptographies multivariées sont basées sur la difficulté de résoudre les systèmes d'équations multivariées, ou des systèmes d'équation faisant entrer la notion d'équations à plusieurs variables. Ces équations consistent en la somme de termes constituées d'une seule inconnue (de degré un), x_4 dans le cas de l'exemple ci-dessous. Ou de la multiplication de deux (on parle de termes de degré deux, ou quadratiques) variables, x_2x_3 dans notre exemple.

Exemple : Soit le système d'équations multivariées ci-dessous :

$$\begin{cases} x_1x_2 + x_3x_4 + x_2 = 1 \\ x_1x_3 + x_1x_4 + x_2x_3 = 12 \\ x_1^2 + x_3^2 + x_4 = 4 \\ x_2x_3 + x_2x_4 + x_1 + x_4 = 0 \end{cases};$$

Pour résoudre ce système d'équation, nous avons besoin de rechercher les valeurs de x_1, x_2, x_3 et x_4 qui vérifient les 4 équation données. La solution pourrait appartenir à l'ensemble des nombres réels, des nombres entiers ou encore sur un ensemble fini de nombres. En cryptographie cependant, le système d'équations est souvent défini sur l'ensemble des entiers modulo un nombre premier donné, ou encore sur les valeurs binaires (constituées de 0 et d 1).

Le *système d'équations quadratiques multivariées* (*Multivariate Quadratic equation*, pour les intitiales *MQ*, en anglais) est classifié comme étant des problèmes NP-difficiles. Est, par conséquent, candidat à la cryptographie post-quantiques. De telles classes de problèmes (classe NP-difficile) sont difficilement résolues par les ordinateurs quantiques. [3.06]

La robustesse des cryptosystèmes basés sur les MQ dépend des paramètres choisis, tels que le nombre d'équations utilisées, le cardinal de l'ensemble de nombres sur lequel il est défini (nous faisons allusion au nombre premier utilisé pour l'opération de réduction en modulo), entre autres.

Mais le choix des paramètres sécurisés reste difficile (plus d'un cryptosystème multivarié considéré comme sûr ont été rompus). Ainsi, l'usage de la cryptographie multivariée se fait encore rare au moment où nous parlons. Un avantage pratique, cependant, est qu'ils produisent des signatures courtes. C'est dans ce domaine (cryptosystème dans le groupe signature), que ces cryptosystèmes excellent (quatre algorithmes basés sur les MQ sont passés à la deuxième étape de normalisation des cryptographies post-quantiques, organisée par le NIST). [3.06]

c. *Algorithme de signature UOV (pour Unbalanced Oil and Vinegar) comme exemple*

L'algorithme de signature *UOV* (initiales pour *Unbalanced Oil and Vinegar*) a été proposé par *Kipnis* et *Patarin* en 1999. Il (l'algorithme) peut être décrit comme suit.

Soit $\mathbb{F}$ un corps fini et o et v des entiers. Nous définissons $n = o + v$, $V = \{1, \dots, v\}$, et $O = \{v + 1, \dots, n\}$. Nous appelons $x_1, \dots, x_v$ les variables Vinegar et $x_{v+1}, \dots, x_n$ les variables Oil. Nous

sommes en présence d'Unbalanced Oil and Vinegar (UOV) lorsque $v > o$ et de (Balanced) Oil and Vinegar (B)OV dans le cas où $o = v$.

i. <u>Génération de la clé</u>

La transformation centrale $\mathcal{F}: \mathbb{F}^n \longrightarrow \mathbb{F}^o$ de l'algorithme UOV est constituée de o polynomes quadratiques $f^{(1)}, \dots, f^{(o)}$ de la forme :

$$f^{(k)} = \sum_{i,j \in V} \alpha_{ij}^{(k)} x_i x_j + \sum_{i \in V, j \in O} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in O \cup V} \gamma_i^{(k)} x_i + \delta^{(k)} \qquad \textbf{(}3.05\textbf{)}$$

$$k = \{1, \dots, o\}$$

Il est à noter que les polynômes $f^{(1)}, \dots, f^{(o)}$ ne contiennent pas de termes quadratiques $x_i x_j$ avec les indices $i, j \in O$ (les deux appartiennent toutes à O). Ce fait sera utilisé plus tard pour inverser $\mathcal{F}$.

Pour cacher la structure de $\mathcal{F}$ dans la clé publique, on combinera $\mathcal{F}$ avec une transformation affine inversable $\mathcal{T}: \mathbb{F}^n \longrightarrow \mathbb{F}^n$. Ainsi, la clé publique de l'algorithme sera de la forme : $\mathcal{P} = \mathcal{F} \circ \mathcal{T}: \mathbb{F}^n \longrightarrow \mathbb{F}^o$. La clé privée est constituée par $\mathcal{F}$ et $\mathcal{T}$.

Exemple : Soit $\mathbb{F} = \mathrm{GF}(7)$ et $o = v = 2$. Soit la transformation centrale $\mathcal{F} = \left(f^{(1)}, f^{(2)}\right)$ de notre instance de (Balanced) OV par :

$f^{(1)}(x_1, \dots, x_4) = 2x_1^2 + 3x_1x_2 + 6x_1x_3 + x_1x_4 + 4x_2^2 + 5x_2x_4 + 3x_1 + 2x_2 + 5x_3 + x_4 + 6$;

$f^{(2)}(x_1, \dots, x_4) = 3x_1^2 + 6x_1x_2 + 5x_1x_4 + 3x_2^2 + 5x_2x_3 + x_2x_4 + 2x_1 + 5x_2 + 4x_3 + 2x_4 + 1$;

Pour trouver la pré-image $x = (x_1, \dots, x_4)$ de $y = (3, 4)$ sous la transformation centrale $\mathcal{F}$, nous choisissons des valeurs aléatoires de x_1 et x_2, exemple $(x_1, x_2) = (1, 4)$ et remplaçons ces valeurs dans $f^{(1)}$ et $f^{(2)}$. On obtient :

$\tilde{f}^{(1)}(x_3, x_4) = 4x_3 + x_4 + 4$;

$\tilde{f}^{(2)}(x_3, x_4) = 3x_3 + 4x_4$;

En résolvant le système d'équations linéaires $\tilde{f}^{(1)} = y_1 = 3$, $\tilde{f}^{(2)} = y_2 = 4$, nous obtenons

$(x_3, x_4) = (1, 2)$. Ainsi, la pré-image requise est $x = (1, 4, 1, 2) \in \mathbb{F}^4$.

ii. Inversion de la transformation centrale

Pour retrouver la pré-image de $x \in \mathbb{F}^n$ de $y \in \mathbb{F}^o$ sous la transformation centrale $\mathcal{F}$, on choisit d'une manière aléatoire la valeur des variables Vinegar $x_1, \dots, x_v$ et les substitue dans les polynômes $f^{(1)}, \dots, f^{(o)}$. Du fait des structures spéciales des polynômes centraux, nous obtenons par cette stratégie o polynômes linéaires $\tilde{f}^{(1)}, \dots, \tilde{f}^{(o)}$ dans les o variables Oil $x_{v+1}, \dots, x_n$. Nous pouvons résoudre le système linéaire qui en résulte $\tilde{f}^{(k)}(x_{v+1}, \dots, x_n) = y_k,\ k = 1, \dots, o$ par la méthode d'*élimination Gaussienne*. Si le système n'a pas de solution, on choisira de nouvelles valeurs pour les variables Vinegar et essayer à nouveau.

iii. Génération de la signature

Pour générer une signature $z \in \mathbb{F}^n$ d'un message d, on peut utiliser la fonction de hachage $\mathcal{H}: \{0,1\} \longrightarrow \mathbb{F}^o$ pour calculer $w = \mathcal{H}(d) \in \mathbb{F}^o$ et suivre les étapes ci-dessous :

- Calculer la pré-image de $x \in \mathbb{F}^n$ de w par la transformation centrale $\mathcal{F}$. Ceci peut se faire suivant notre paragraphe précédent ;

- Calculer la signature $z \in \mathbb{F}^n$ du document d par $z = \mathcal{T}^{-1}(x)$;

iv. Vérification de la signature

Pour vérifier l'authenticité d'une signature $z \in \mathbb{F}^n$, nous pouvons utiliser la fonction de hachage $\mathcal{H}$ pour calculer $w = \mathcal{H}(d) \in \mathbb{F}^o$ et calculer $w' = \mathcal{P}(z)$. Si nous trouvons $w' = w$, la signature est authentique, sinon c'est rejetée.

v. Sécurité de l'UOV

Pour la sécurité de l'UOV, nous avons besoin de $v \gtrsim 2o$ (d'où le nom Unbalanced Oil and Vinegar). A côté de cela, l'algorithme de signature UOV a résisté (pour des paramètres appropriés cependant) à la cryptanalyse voilà maintenant 20 années depuis sa création. Et de ce fait estimé être sécurisé.

3.3 Généralités sur les lattices

3.3.1 Lattices

Soit $\mathbb{R}^m$ l'espace Euclidien de dimension m. Une lattice dans $\mathbb{R}^m$ est l'ensemble :

$$\mathcal{L}(b_1, \dots, b_n) = \left\{ \sum_{i=1}^{n} x_i b_i \, , x_i \in \mathbb{Z} \right\} \quad \textbf{(}3.06\textbf{)}$$

de tous les combinaisons intégrales (entières) de n vecteurs $b_1, \dots, b_n$, linéairement indépendants dans $\mathbb{R}^m$ ($m \geq n$). Les entiers m et n sont appelés respectivement les *rang* et *dimension* de la lattice (respectivement *rank* et *dimension* en anglais). La séquence de vecteurs $b_1, \dots, b_n$est appelée *base de la lattice*, et est représentée d'une manière plus convenable en tant que matrice :

$$B = [b_1, \dots, b_n] \in \mathbb{R}^{m \times n} \quad \textbf{(}3.07\textbf{)}$$

Les vecteurs de base étant représenté en tant que colonnes. En utilisant la notation matricielle, la formule **(**3.06**)** peut se réécrire comme suit :

$$\mathcal{L}(B) = \{Bx, x \in \mathbb{Z}^n\} \quad \textbf{(}3.08\textbf{)}$$

où Bx étant la multiplication matrice vecteur usuelle.

Une même lattice peut avoir différentes bases. Par exemple, la lattice que nous avons vue dans la **Figure 3.03** peut être générée par les bases $b'_1 = b_1 + b_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ et $b'_2 = 2b_1 + b_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$. Ce fait est illustré par **Figure 3.06**. Notons que même si les grilles générées par b'_1 et b'_2 sont différentes, l'ensemble des points d'intersection qui en résulte reste le même. En d'autres termes, $\{b_1, b_2\}$ et $\{b'_1, b'_2\}$ sont deux bases différentes de la même lattice $\mathcal{L}(b_1, b_2) = \mathcal{L}(b'_1, b'_2)$. Tout au long du présent ouvrage, nous allons adopter la notation qui suit. Les points de la lattice seront dénotés en vecteurs colonnes. Mais les vecteurs étant plus aisément représentables en ligne, nous utiliserons l'annotation de transposée A^T, qui signifie donc transposée de la matrice A. Nos vecteurs b_1 et b_2 peuvent ainsi être écrits : $b_1 = [1, 2]^T$ et $b_1 = [1, -1]^T$.

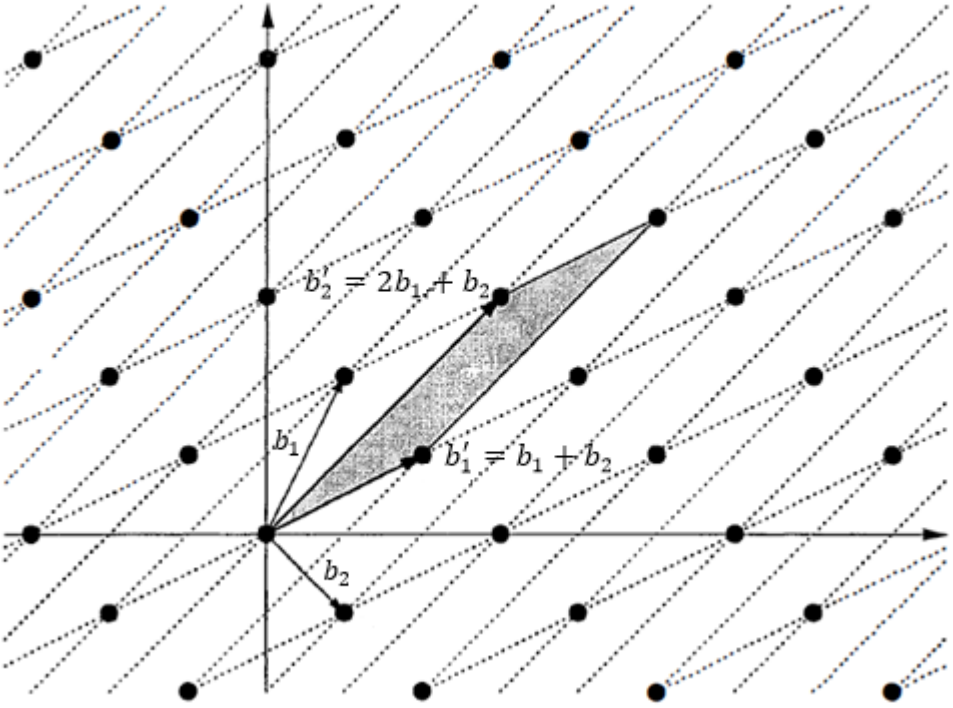


Figure 3.06 : *La même lattice que la* **Figure 3.03**, *mais cette fois ci, générée par de différents vecteurs de base* $\boldsymbol{b_1'}$ *et* $\boldsymbol{b_2'}$.

Un simple exemple de lattice de dimension n est donné par l'ensemble $\mathbb{Z}^n$ représenté par des vecteurs de coordonnées entières. Une possible base est donnée par les vecteurs unitaires standards :

$$e_i = \left[\underbrace{0, \dots, \overbrace{1}^{i}, \dots, 0}^{n}\right]^T \qquad \textbf{(}3.09\textbf{)}$$

En notation matricielle, $\mathbb{Z}^n = \mathcal{L}(I)$, ou $I \in \mathbb{Z}^{n \times n}$ étant la matrice identité de dimension n (ayant 1 comme diagonale et 0 ailleurs).

Quand $n = m$, c'est-à-dire le nombre de vecteurs de base est égal au nombre de coordonnées, nous sommes en présence de lattice de *plein rang* ou de *pleine dimension* (*full rank* ou *full dimension*). D'une manière équivalente, une lattice $\mathcal{L}(B) \subseteq \mathbb{R}^m$ est de rang plein si et seulement si l'espace vectoriel engendré (*linear span* en anglais) par les vecteurs de base :

$$\text{span}(B) = \{Bx, x \in \mathbb{R}^n\} \qquad \textbf{(}3.10\textbf{)}$$

équivaut en entier l'espace $\mathbb{R}^m$. La différence entre **(**3.08**)** et **(**3.10**)** est que dans **(**3.10**)**, on peut utiliser des coefficients réels pour combiner les vecteurs de base, si dans **(**3.08**)**, seuls des coefficients en entiers sont autorisés.

On pourra remarquer que $\text{span}(B)$ ne dépend pas de la base particulière B. C'est-à-dire si B et B' génèrent la même lattice, alors $\text{span}(B) = \text{span}(B')$. Ainsi, pour une lattice $\Lambda = \mathcal{L}(B)$, nous pouvons définir l'espace vectoriel engendré $\text{span}(\Lambda)$ sans mentionner la base B. Il est à noter que B est la base de $\text{span}(B)$ en tant qu'espace vectoriel. En particulier, le rang de la lattice $\mathcal{L}(B)$ est égal à la dimension de $\text{span}(B)$ en tant qu'espace vectoriel dans $\mathbb{R}$ et il $(\text{span}(B)$, l'espace vectoriel engendré) ne dépend pas de la lattice, des vecteurs de base en d'autres termes.

D'un autre côté, du point de vue de la lattice, tout ensemble de n vecteurs linéairement indépendants d'une lattice, $B' \in \mathcal{L}(B)$ est une base de $\text{span}(B)$ en tant qu'espace vectoriel. Cependant, B' n'est nécessairement pas une base de $\mathcal{L}(B)$. La **Figure 3.07** illustre à la fois notre lattice originale (celle à base de b_1 et b_2), et la seconde lattice, différente de la première, à base de $b_1' = b_1 - b_2 = [2, 1]^T$ et $b_2' = b_1 + b_2 = [0, 3]^T$.

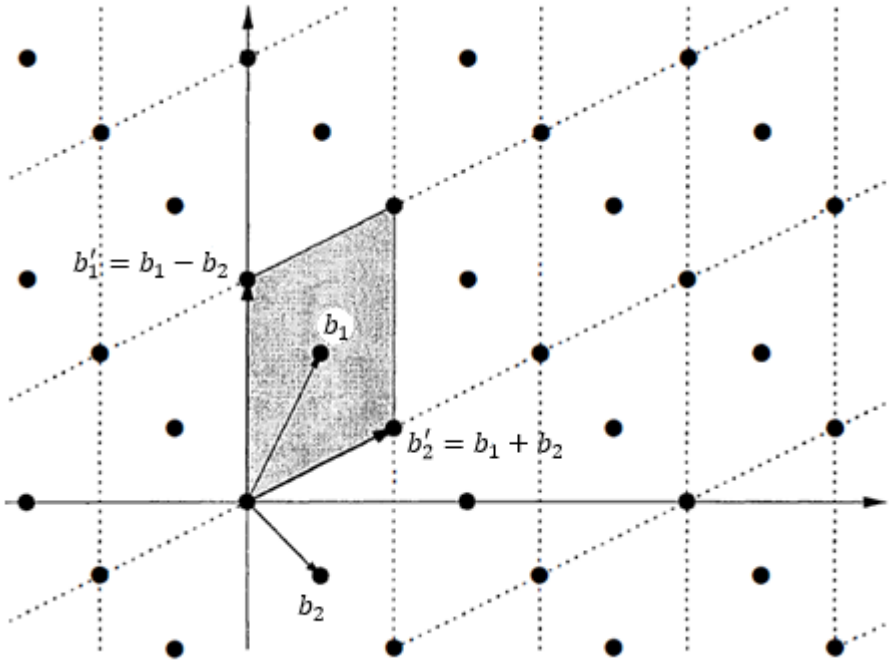


Figure 3.07 : *Les bases* $\boldsymbol{b_1' = b_1 - b_2}$ *et* $\boldsymbol{b_2' = b_1 + b_2}$ *engendrant une lattice différente que celle générée par les bases originales* $\boldsymbol{b_1}$ *et* $\boldsymbol{b_2}$.

Les vecteurs b_1' et b_2' sont linéairement indépendants, sont de ce fait une base du plan $\mathbb{R}^2 =$

$\text{span}(b_1, b_2)$ en tant qu'espace vectoriel. Cependant, ils (les vecteurs b_1' et b_2') ne sont pas une base de la lattice $\mathcal{L}(b_1, b_2)$ dû au fait que le point b_1 ne peut être exprimé en tant que combinaison linéaire, entière, de b_1' et b_2'. Il y a une caractéristique géométrique que doit respecter des vecteurs linéairement indépendants d'une lattice pour que ceux-ci (les vecteurs) puissent générer la même lattice en totalité. Pour tout n vecteurs lattices linéairement indépendants $B' = [b_1', \dots, b_n']$ avec $b_i' \in \mathcal{L}(B) \subset \mathbb{R}^m$ pour tout $i = 1, \dots, n$, nous définissons le parallélépipède semi-ouvert :

$$\mathcal{P}(B') = \{B'x, 0 \leq x_i < 1\} \quad \textbf{(}3.11\textbf{)}$$

B' est une base de la lattice $\mathcal{L}(B)$ si et seulement si $\mathcal{P}(B')$ ne contient aucun vecteur lattice autre que l'origine. Les figures Figure 3.03, Figure 3.06 et **Figure 3.07** illustrent cette égalité et/ou différence entre les lattices. Les lattices dans les figures Figure 3.03 et **Figure 3.06** sont les mêmes étant donné que le parallélépipède semi-ouvert $\mathcal{P}(B')$ (d'une couleur un peu sombre dans la seconde figure, **Figure 3.06**) ne contient aucun autre point de lattice que l'origine. On a $\mathcal{L}(B) = \mathcal{L}(B')$. Dans la **Figure 3.07** cependant, le parallélépipède semi-ouvert $\mathcal{P}(B')$ contient le point b_1. Par conséquent, $\mathcal{L}(B) \neq \mathcal{L}(B')$, donc B' ne peut être considérée comme une autre base de $\mathcal{L}(B)$.

Il est à noter que B' est un ensemble de vecteurs linéairement indépendants. $\mathcal{L}(B')$ est donc une lattice et B' en est la base. De la **Figure 3.07**, nous voyons que tous les points de $\mathcal{L}(B')$ appartiennent à $\mathcal{L}(B)$, c'est-à-dire $\mathcal{L}(B') \subseteq \mathcal{L}(B)$. Quand $\mathcal{L}(B') \subseteq \mathcal{L}(B)$, nous disons que $\mathcal{L}(B')$ est une *sous-lattice* (*sublattice* en anglais) de $\mathcal{L}(B)$. Quand $\mathcal{L}(B') = \mathcal{L}(B)$, nous disons que les deux lattices sont *équivalentes*. Quand $\mathcal{L}(B') \subseteq \mathcal{L}(B)$, mais que $\mathcal{L}(B') \neq \mathcal{L}(B)$, c'est-à-dire $\mathcal{L}(B') \subset \mathcal{L}(B)$, nous disons $\mathcal{L}(B')$ est une *sous-lattice stricte* (*proper sublattice* en anglais) de $\mathcal{L}(B)$.

Deux bases équivalentes peuvent être algébriquement décrites comme suit : deux bases B et B' sont dites équivalentes, en termes de lattice, s'il existe une matrice uni-modulaire $U \in \mathbb{Z}^{m \times n}$ (c'est-à-dire une matrice dont les éléments sont des nombres entiers, et ayant un déterminant $\det(U) = \pm 1$) tel que $B' = BU$.

Démonstration :

♣

En premier lieu, supposons que $B' = BU$ pour une matrice uni-modulaire U à élément entier donné.

Si U est uni-modulaire, alors U^{-1} l'est aussi. En particulier, U et U^{-1} sont des matrices à élément entiers et justifiant $B' = BU$ donc $B = B'U^{-1}$. Il s'ensuit que $\mathcal{L}(B') \subseteq \mathcal{L}(B)$ et que $\mathcal{L}(B) \subseteq \mathcal{L}(B')$. Donc forcément $\mathcal{L}(B') = \mathcal{L}(B)$, soit en d'autres termes, B' et B génèrent les mêmes lattices.

Dans un second temps, supposons maintenant que B et B' sont les bases d'une même lattice, soit $\mathcal{L}(B) = \mathcal{L}(B')$. Alors, d'après la propriété de deux bases générant la même lattice (la seconde base s'obtient par une combinaison linéaire de la première et vice versa), il existe deux matrices carrées, entières, V et W de telle sorte que $B = CW$ et que $C = BV$. En combinant ces deux équations, on aura $B = BVW$, ou encore $B(I - VW) = 0$. Etant donné que B est linéairement indépendant, nous aurons $I - VW = 0$, soit $VW = I$. En particulier, $\det(VW) = \det(V)\det(W) = \det(I) = 1$. Etant donné que V et W sont des matrices carrées à éléments entiers, $\det(V)$ et $\det(W)$ appartiennent à $\mathbb{Z}$. Donc finalement $\det(V) = \det(W) = \pm 1$. [3.12], [3.13]

♦

Un dernier aspect intéressant à mentionner est qu'une lattice peut être caractérisée sans pour autant faire référence à une quelconque base. Une lattice peut être définie comme un sous ensemble Λ discret de $\mathbb{R}^m$ qui est *stable* ou *clos* (*closed* en anglais) pour l'opération soustraction. C'est-à-dire si $x \in \Lambda$ et $y \in \Lambda$, alors $x - y \in \Lambda$. Dans le cas présent, discret signifie qu'il existe un nombre réel positif $\lambda > 0$ tel que la distance entre tout couple de vecteurs lattices est au moins λ. Un exemple typique en est l'ensemble $\Lambda = \{x \in \mathbb{Z}^n, Ax = 0\}$ de solutions entières d'un système d'équations linéaires homogènes (homogène, pour rappel, signifie que la matrice A est singulière ou non inversible, en addition, le terme constant – celui après le signe d'égalité – est égal à 0).

3.3.1.1 Déterminant

Le déterminant d'une lattice $\Lambda = \mathcal{L}(B)$, noté $\det(\Lambda)$, est le volume de dimension n du parallélépipède $\mathcal{P}(B)$ sur lequel s'étendent les vecteurs de bases (se référer à la zone ombrée des figures **Figure 3.03** et **Figure 3.06**). Le déterminant d'une lattice est invariant, c'est-à-dire il ne dépend pas de la base utilisée pour le (le déterminant) calculer. Cela découle du fait que, si deux bases B et B' sont équivalentes alors $B = B'U$, avec U une matrice uni-modulaire.

En termes géométriques, le déterminant d'une lattice correspond à l'inverse de la densité de points de la lattice dans $\mathrm{span}(B)$. Comme exemple, considérons encore une fois les figures **Figure 3.03** et

Figure 3.06. La surface du parallélépipède $\mathcal{P}(B)$ est exactement la même parce que les deux bases génèrent la même lattice. Par contre, dans la **Figure 3.07**, la surface du parallélépipède est deux fois plus grande, dû au fait que la lattice $\mathcal{L}(B')$ est une sous-lattice de $\mathcal{L}(B)$. Plus exactement, la lattice $\mathcal{L}(B')$ contient deux fois moins de points que celle (la lattice) originale $\mathcal{L}(B)$.

Une méthode pour calculer le déterminant d'une lattice passe par l'*algorithme d'orthogonalisation de Gram-Schmidt*. Pour toute séquence de vecteurs $b_1, \ldots, b_n$, on définit les vecteurs orthogonalisés $b_1^*, \ldots, b_n^*$ par :

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$$
$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \qquad \textbf{(}3.12\textbf{)}$$

Où $\langle x, y \rangle = \sum_{i=1}^{m} x_i y_i$ est le produit scalaire dans $\mathbb{R}^m$. Pour tout i, b_i^* est la projection de b_i orthogonale à $b_1, \ldots, b_{i-1}$. En particulier, $\text{span}(b_1, \ldots, b_i) = \text{span}(b_1^*, \ldots, b_i^*)$, et que les vecteurs b_i^* sont deux à deux orthogonaux, c'est-à-dire $\langle b_i^*, b_j^* \rangle = 0$ pour $i \neq j$. Le déterminant de la lattice équivaut au produit de la longueur des vecteurs orthogonaux :

$$\det(\Lambda) = \prod_{i=1}^{n} \|b_i^*\| \qquad \textbf{(}3.13\textbf{)}$$

où $\|x\| = \sqrt{\sum_i x_i^2}$ est la longueur Euclidienne usuelle. Nous remarquons que la définition des vecteurs orthogonalisés b_i^* dépend de l'ordre des vecteurs de bases originaux. Etant donné la matrice $B = [b_1^*, \ldots, b_n^*]$, nous définissons $B^* = [b_1^*, \ldots, b_n^*]$. B^* est une base de $\text{span}(B)$, en tant qu'espace vectoriel. Cependant, B^* n'est pas forcément une base de $\Lambda = \mathcal{L}(B)$. En particulier, ce ne sont pas toutes les lattices qui ont une base constituée de vecteurs deux à deux orthogonaux.

Un autre moyen pour calculer le déterminant d'une lattice, est de passer par la *matrice de Gram*

$B^T B$. On a :

$$\det(\Lambda) = \sqrt{\det(B^T B)} \quad \textbf{(}3.14\textbf{)}$$

En particulier si la lattice $\mathcal{L}(B)$ est de pleine dimension (c'est-à-dire $m = n$), alors B est une matrice carrée non singulière. Dans ce cas, le déterminant de la lattice $\det\big(\mathcal{L}(B)\big)$ est égal à la valeur absolue du déterminant de la matrice de base $\det(B)$:

$$\det(\Lambda) = |\det(B)| \quad \textbf{(}3.15\textbf{)}$$

Démonstration :

♣

Rappelons-nous l'algorithme d'orthogonalisation de Gram-Schmidt. La définition montre que les vecteurs orthogonalisés B^* justifient, $B = B^* T$ où T est une matrice triangulaire supérieure, avec des 1 dans la diagonale, et les coefficients $\mu_{i,j}$ à la position (j, i) pour tout $j < i$. Ainsi, notre formule pour le déterminant d'une lattice peut être écrite comme : [3.12], [3.13]

$$\sqrt{\det(B^T B)} = \sqrt{\det(T^T B^{*T} B^* T)} = \sqrt{\det(T^T)\det(B^{*T} B^*)\det(T)}$$

Les matrices T^T et T sont triangulaires. Leurs déterminants sont donc égaux au produit de ses éléments diagonaux, qui est égal à 1. Maintenant, pour le terme restant $B^{*T} B^*$, cette matrice est diagonale, parce que les éléments de B^* sont orthogonaux. Ainsi, son (celui de la matrice $B^{*T} B^*$) déterminant peut être calculé par le produit de ses éléments diagonaux :

$$\det(B^{*T} B^*) = \prod_i \langle b_i^*, b_i^* \rangle = \left(\prod_i \|b_i^*\| \right)^2 = (\det(\mathcal{L}(B)))^2$$

En prenant la racine carrée, nous obtenons :

$$\prod_i \|b_i^*\| = \det\big(\mathcal{L}(B)\big)$$

♦

3.3.1.2 Minimums successifs ou Successive minima en anglais

Soit $\mathcal{B}_m(0,r) = \{x \in \mathbb{R}^m, \|x\| < r\}$, la sphère ouverte de dimension m et de rayon r, centrée en 0. La dimension m étant mentionnée clairement dans le contexte, nous pouvons l'omettre et écrire plus simplement $\mathcal{B}(0,r)$. Des constants fondamentaux liés à une lattice Λ donnée sont les constants $\lambda_1, \ldots, \lambda_n$, nommés *minimums successifs* ou *successive minima* en anglais. Le i-ième minimum $\lambda_i(\Lambda)$ est le rayon du plus petit sphère centré sur l'origine, et contenant i vecteurs lattices linéairement indépendants :

$$\lambda_i = \inf\Big\{r, \dim\Big(\mathrm{span}\big(\Lambda \cap \mathcal{B}(0,r)\big)\Big) \geq i\Big\} \qquad \textbf{(}3.16\textbf{)}$$

Où inf tient pour *infimum* et dim pour *dimension*. L'infimum se définissant comme étant la borne inférieure, ou encore le plus grand des minorants d'un ensemble. Une différence entre l'infimum et le minimum étant que l'infimum n'est jamais atteint (ayons en tête par exemple l'infimum 0 de la fonction $f(x) = 1/x$ quand $x \longrightarrow \infty$), si le minimum l'est (le minimum 0, toujours, de la fonction $f'(x) = x^2$).

Les minimums successifs peuvent être définis suivant une norme donnée. Une norme, est une fonction $\|.\|: \mathbb{R}^n \longrightarrow \mathbb{R}$, positive, homogène, qui satisfait l'inégalité triangulaire :

- $\|x\| \geq 0$, avec l'égalité qui tient uniquement lorsque $x = 0$;

- $\|\alpha x\| = |\alpha|\|x\|$;

- $\|x + y\| \leq \|x\| + \|y\|$;

pour $x, y \in \mathbb{R}^n$ et $\alpha \in \mathbb{R}$. Une importante famille de normes est donnée par la norme ℓ_p. Pour tout

$p \geq 1$, la norme ℓ_p d'un vecteur $x \in \mathbb{R}^n$ est donnée par :

$$\|x\|_p = \left(\sum_{i=1}^{n} x_i^p \right)^{1/p} \quad \textbf{(}3.17\textbf{)}$$

d'où la norme ℓ_1:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

la norme ℓ_2 :

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

la norme ℓ_∞ :

$$\|x\|_\infty = \lim_{p \to \infty} \|x\|_p = \max_{i=1,\ldots,n} |x_i|$$

Nous tenons à faire remarquer que quand $p < 1$, la fonction **(**3.17**)** n'est pas une norme dû au fait que l'inégalité triangulaire n'est pas respectée. Une autre remarque est que la valeur des minimums successifs dépend de la norme utilisée.

Exemple : Considérons la lattice générée par la formulation ci-dessous. Cette dernière (la formulation) est :

$$\Lambda = \{v \in \mathbb{Z}^2, v_1 + v_2 = 0 \bmod 2\}$$

Cette lattice possède comme base l'ensemble de vecteurs b_1 et b_2 donnés par $b_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ et $b_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

En prenant en compte la norme ℓ_1, le vecteur b_1 est le vecteur le plus court de la lattice Λ. On a donc $\lambda_1 = \|b_1\|_1 = 2$. Avec les normes ℓ_2 et ℓ_∞ par contre, cette propriété (vecteur le plus court) est détenue par b_2. On a donc, $\lambda_1 = \|b_2\|_2 = \sqrt{2}$ et $\lambda_1 = \|b_2\|_\infty = 1$.

La norme qui nous intéressent dans la suite de cette ouvrage est la norme ℓ_2, sauf bien sûr si explicitement mentionnée :

$$\mathrm{dist}(x-y) = \|x-y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad \textbf{(}3.18\textbf{)}$$

Il s'ensuit de par notre exemple précédent que la lattice considérée contient un vecteur b de telle sorte que $\lambda_1 = \|b\|$. Cela est en effet vrai pour toute lattice donnée, et non pas pour celle mentionnée en exemple seulement. Il s'ensuit d'après la caractéristique discrète d'une lattice qu'il existe une succession de vecteurs, linéairement indépendants $x_1, \dots, x_n \in \Lambda$ telle que $\|x_i\| = \lambda_i$ pour tout $i = 1, \dots, n$. L'infimum dans notre formule **(**3.16**)** devient un minimum (il est donc atteint, en d'autres termes pour rappel) si le sphère ouvert $\mathcal{B}(0,r)$ est remplacé par un sphère fermé $\overline{\mathcal{B}}(0,r) = \{x \in \mathbb{R}^m, \|x\| \leq r\}$. En particulier, $\lambda_1(\Lambda)$ est la longueur du plus court, non égal à 0, vecteur lattice. Et est également équivalent à la plus courte distance entre deux points distincts de la lattice :

$$\lambda_1(\Lambda) = \min_{x \neq y \in \Lambda} \|x-y\| = \min_{x \in \Lambda \setminus \{0\}} \|x\| \qquad \textbf{(}3.19\textbf{)}$$

Nous allons, dans les paragraphes qui suivent, démontrer en partie (en effet la démonstration complète est assez longue et nous renvoyons nos lecteurs à notre bibliographie [3.12] pour la version complète) que toute lattice contient un vecteur non nul de longueur minimale. La version complète de la démonstration est par la suite généralisable au fait qu'il existe n vecteurs linéairement indépendants, $v_1, \dots, v_n$, qui peuvent être associés aux n minimums successifs $\lambda_1, \dots, \lambda_n$ tel que $\lambda_i = \|v_i\|$ pour $i = 1, \dots, n$.

Soit une lattice $\mathcal{L}(B)$, et considérons le premier minimum :

$$\lambda_1 = \inf\{\|v\|, v \in \mathcal{L}(B) \setminus \{0\}\}$$

La partie de la démonstration que nous allons détailler est celle mettant en exergue le fait que λ_1 est strictement positif : $\lambda_1 > 0$. Cette affirmation conduit au théorème qui va suivre.

Théorème 3.01 :

Soit B une base d'une lattice donnée, et B^* l'orthogonalisation de Gram-Schmidt correspondant. Le premier minimum de la lattice, par respect à la norme ℓ_2, satisfait :

$$\lambda_1 \geq \min_j \left\| b_j^* \right\| > 0$$

Démonstration :

♣

Soit un vecteur lattice générique non égal à 0, Bx (où $x \in \mathbb{Z}^n$ et $x \neq 0$), et i le plus grand indexe de telle sorte que $x_i \neq 0$. De par cette définition de i, nous avons $Bx = \sum_{j=1}^{i} b_j x_j$. En utilisant la définition des vecteurs orthogonalisés **(**3.12**)** nous avons :

$$\begin{aligned} \langle Bx, b_i^* \rangle &= \sum_{i=1}^{i} \langle b_j, b_i^* \rangle x_j = \langle b_i, b_i^* \rangle x_i \\ &= \langle b_i^* + \sum_{j<i} \mu_{ij} b_j^* , b_i^* \rangle \, x_i = \langle b_i^*, b_i^* \rangle x_i + \sum_{j<i} \mu_{ij} \langle b_j^*, b_i^* \rangle \, x_i \\ &= \|b_i^*\|^2 x_i \end{aligned}$$

Etant donné que x_i est un entier non égal à 0,

$$|\langle Bx, b_i^* \rangle| = \|b_i^*\|^2 |x_i| \geq \|b_i^*\|^2$$

Une connaissance de base en algèbre linéaire nous dit que $|\langle x, y \rangle| \leq \|x\| \|y\|$, pour tout vecteur x et y donnés. On a donc : $\|Bx\| \|b_i^*\| \geq |\langle Bx, b_i^* \rangle| \geq \|b_i^*\|^2$. Soit en prenant uniquement les bornes des deux inégalités $\|Bx\| \|b_i^*\| \geq \|b_i^*\|^2$ et comme $\|b_i^*\| \neq 0$, on a donc $\|Bx\| \geq \|b_i^*\|$ (1).

Comme $\lambda_1 = \inf \|Bx\|$ (2) par définition, et que $\|b_i^*\| \geq \min_j \left\| b_j^* \right\|$ (3). Ces trois dernières égalités/inégalités - (1), (2) et (3) - nous donnent :

$$\lambda_1 = \inf \|Bx\| \geq \|b_i^*\| \geq \min_j \left\| b_j^* \right\| > 0$$

En enlevant les termes non présents dans notre *Théorème 3.01*, on arrive à :

$$\lambda_1 \geq \min_j \left\| b_j^* \right\| > 0$$

♦

Un second théorème, que nous allons outrepasser la démonstration comme énoncé plus haut est comme suit.

Théorème 3.02 :

Soit Λ une lattice de rang n, ayant comme minimums successifs $\lambda_1, \dots, \lambda_n$. Alors, il existe des vecteurs lattices linéairement indépendants $v_1, \dots, v_n \in \Lambda$ tels que $\|v_i\| = \lambda_i$ pour $i = 1, \dots, n$.

3.3.1.3 Théorème de Minkowski

Avant d'étaler les théorèmes de Minkowski, nous allons passer par celui (le théorème) qui va suivre plus bas. Pour les démonstrations, nous tenons à informer nos lecteurs qu'elles (les démonstrations) sont majoritairement disponibles dans notre bibliographie ([3.12] notamment pour le cas du présent chapitre autour de la lattice). Nous n'en énumèrerons que quelques-unes, dû à notre manque d'espace (le présent chapitre est limité à 50 pages, si le nombre de théories à étaler est assez nombreux).

Théorème 3.03 :

(*Théorème de Blichfeldt*, du nom de son inventeur *Hans Blichfeldt*, qui l'a démontré vers 1914). Pour toute lattice Λ et pour un ensemble mesurable $S \subseteq \text{span}(\Lambda)$, si S a un volume $\text{vol}(S) > \det(\Lambda)$, alors il existe deux points distincts $z_1, z_2 \in S$ tels que $z_1 - z_2 \in \Lambda$.

De ce théorème de Blichfeldt se découle le premier théorème de Minkowski.

Théorème 3.04 :

(*Théorème du corps convexe de Minkowski* ou encore *Minkowski's convex body theorem en anglais*). Pour toute lattice Λ de rang n, et pour tout ensemble convexe $S \subset \text{span}(\Lambda)$ symétrique par rapport à l'origine, si $\text{vol}(S) > 2^n \det(\Lambda)$, alors S contient un point lattice $v \in S \cap \Lambda \setminus \{0\}$.

Démonstration :

♣

Soit l'ensemble $S' = \{x, 2x \in S\}$. Le volume S' satisfait :

$$\text{vol}(S') = 2^{-n}\text{vol}(S) > \det(\Lambda)$$

Ainsi d'après le théorème de Blichfeldt, il existe deux points distincts $z_1, z_2 \in S'$ tel que $z_1 - z_2 \in \Lambda$. D'après la définition de S' nous avons $2z_1, 2z_2 \in S$. Et sachant que S est symétrique par rapport à l'origine, nous avons $-2z_2 \in S$. Finalement, de par le fait que S ait été supposé être convexe, le point médian du segment $[2z_1, -2z_2]$ appartient aussi à S :

$$\frac{2z_1 + (-2z_2)}{2} = z_1 - z_2 \in S$$

Ceci prouve que $v = z_1 - z_2$ est un point lattice non nul de S.

♦

Le théorème du corps convexe de Minkowski peut être utilisé pour majorer la longueur du vecteur non nul, le plus court, d'une lattice de rang n. Soit $S = \mathcal{B}\left(0, \sqrt{n}\det(\Lambda)^{\frac{1}{n}}\right) \cap \text{span}(\Lambda)$ la boule ouverte de rayon $\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$ dans $\text{span}(\Lambda)$. Notons que S a un volume strictement supérieur à $2^n\det(\Lambda)$ parce qu'il contient un hypercube de dimension n et de côté égal à $2\det(\Lambda)^{\frac{1}{n}}$. D'après le théorème de Minkowski, il existe un vecteur lattice, non nul, $v \in \mathcal{L}(B) \setminus \{0\}$ tel que $v \in S$, c'est-à-dire, $\|v\| < \sqrt{n}\det(\Lambda)^{\frac{1}{n}}$. Ceci prouve que pour toute lattice Λ de rang n, la longueur du vecteur le plus court (dans la norme ℓ_2) est de :

$$\lambda_1 < \sqrt{n}\det(\Lambda)^{\frac{1}{n}} \qquad \textbf{(}3.20\textbf{)}$$

Ce résultat est connu sous le nom de *premier théorème de Minkowski*. Mais ce dernier (Minkowski) est encore allé plus loin en faisant intervenir tous les minimums successifs, connus cette fois ci sous le nom de *second théorème de Minkowski*. Il se trouve que $\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$ est non seulement le majorant du premier minimum λ_1, mais aussi celui (le majorant) de la moyenne géométrique de tous les

minimum successifs.

Théorème 3.05 :

(Second théorème de Minkowski) Pour tout lattice Λ de rang n, les minimums successifs (dans la norme ℓ_2, toujours) $\lambda_1, \dots, \lambda_n$ satisfont :

$$\left(\prod_{i=1}^{n} \lambda_1\right)^{\frac{1}{n}} < \sqrt{n}\det(\Lambda)^{\frac{1}{n}} \qquad \textbf{(}3.21\textbf{)}$$

Démonstration :

♣

Pour la démonstration, nous allons le faire dans la norme ℓ_2, la démonstration pour une norme donnée, en général, s'avère relativement complexe.

Soient $x_1, \dots, x_n$ les vecteurs lattices de longueur égale à chacun des minimums successifs $\|x_i\| = \lambda_i$ pour $i = 1, \dots, n$. Et assumons l'affirmation contradictoire $\prod_{i=1}^{n} \lambda_i \geq \sqrt{n}^n \det(\Lambda)$. Soit les vecteurs othogonalisés de Gram-Schmidt x_i^* et définissons la transformation :

$$T\left(\sum c_i x_i^*\right) = \sum \lambda_i c_i x_i^*$$

Cette transformation étend chacun des coordonnés x_i^* par un facteur λ_i. Soit $S = \mathcal{B}(0,1) \cap \text{span}(B)$ la boule ouverte de dimension n et de rayon unitaire. Si nous appliquons la transformation T à S nous obtenons un corps convexe symétrique $T(S)$ de volume :

$$\begin{aligned} \text{vol}\big(\text{T(S)}\big) &= \left(\prod_i \lambda_i\right) \text{vol}(S) \\ &\geq \sqrt{n}^n \det(\Lambda)\text{vol}(S) \\ &= \text{vol}\big(\sqrt{n}S\big)\det(\Lambda) \end{aligned}$$

où $\sqrt{n}S$ se réfère à la boule de rayon $\sqrt{n}$. Le volume de $\sqrt{n}S$ est plus grand que 2^n parce que $\sqrt{n}S$ contient un hypercube de côté égal à 2. Ainsi $\mathrm{vol}\big(\mathrm{T(S)}\big) > 2^n \det(\Lambda)$. Ainsi, d'après le théorème des corps convexes de Minkowski, $T(S)$ contient un point lattice y différent de l'origine. Etant donné que $y \in T(S)$, il doit être une transformation d'un point $x \in S$ donné. Soit en d'autre termes $y = T(x)$. D'après la définition de S, nous avons $\|x\| < 1$. Maintenant, en exprimant x et y en termes de bases othogonalisées :

$$x = \sum_i c_i x_i^*$$

$$y = \sum_i \lambda_i c_i x_i^*$$

Sachant que y est non nul, au moins un c_i est non nul pour un i donné. Soit k le plus grand index parmi ces i. Et $k' \leq k$ le plus petit index tel que $\lambda_{k'} = \lambda_k$. Notons que y est linéairement indépendant de $x_1, \dots, x_{k'-1}$ parce que $\langle x_k^*, y \rangle = \lambda_k c_k \|x_k^*\|^2 \neq 0$ et que x_k^* est orthogonal à $x_1, \dots, x_{k'-1}$. Montrons maintenant que $\|y\| < \lambda_k$:

$$\begin{aligned}
\|\mathrm{y}\|^2 &= \left\| \sum_{i \leq k} \lambda_i c_i x_i^* \right\|^2 \\
&= \sum_{i \leq k} \lambda_i^2 c_i^2 \|x_i^*\|^2 \\
&\leq \sum_{i \leq k} \lambda_k^2 c_i^2 \|x_i^*\|^2 \\
&= \lambda_k^2 \left\| \sum_{i \leq k} c_i x_i^* \right\|^2 \\
&= \lambda_k^2 \sum_{i \leq k} \|x\|^2 < \lambda_k^2
\end{aligned}$$

Ceci prouve que $x_1, \dots, x_{k'-1}, y$ sont k' vecteurs lattices linéairement indépendants de longueur strictement inférieure à $\lambda_k = \lambda_{k'}$. Ce qui est contradictoire à la définition du k'-ième minimum

successif $\lambda_{k'}$.

♦

3.3.2 Problèmes mathématiques autour des lattices

Le théorème de Minkowski n'est pas constructif. Nous savons de par le théorème que le vecteur non nul, le plus court existe. Mais la démonstration correspondante ne donne aucun algorithme permettant de calculer ledit vecteur de longueur majorée par $\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$. Et encore moins celui de longueur égale au premier minimum λ_1. Le problème consistant à la recherche du vecteur non nul, le plus court, de longueur λ_1 est connu sous le nom de *problème de vecteur le plus court* ou *Shortest Vector Problem* pour les initiales *SVP* en anglais.

3.3.2.1 Shortest Vector Problem ou SVP

Etant donné une base de lattice $B \in \mathbb{Z}^{m\times n}$ le Shortest Vector Problem ou SVP revient à trouver un vecteur lattice non nul Bx avec $x \in \mathbb{Z}^n \setminus \{0\}$ de telle sorte que $\|Bx\| < \|By\|$ pour tout $y \in \mathbb{Z}^n \setminus \{0\}$.

Nos bibliographies s'accordent sur le fait qu'aucun algorithme s'exécutant en un temps polynomial n'existe pour résoudre le SVP. Et ceci même si le problème est rendu plus facile, en recherchant uniquement un vecteur majoré par la borne supérieure mentionnée dans le premier théorème de Minkowski, $\sqrt{n}\det(\Lambda)^{\frac{1}{n}}$. Un autre problème relatif au SVP, pour lequel (pour cet autre problème) aucun algorithme à temps polynomial n'existe également, est le *problème du vecteur le plus proche*, ou *Closest Vector Problem*, pour les initiales *CVP* en anglais.

3.3.2.2 Closest Vector Problem ou CVP

Soit toujours une base de lattice $B \in \mathbb{Z}^{m\times n}$, en addition de laquelle on mentionne un vecteur cible t, le Closest Vector Problem ou CVP consiste à trouver un vecteur lattice Bx qui se rapproche le plus du vecteur cible $t \in \mathbb{Z}^m$. C'est-à-dire, trouver un vecteur à éléments entiers ($x \in \mathbb{Z}^n$) de telle sorte que $\|Bx - t\| \leq \|By - t\|$ pour tout $y \in \mathbb{Z}^n$.

Pour le SVP et CVP, ci-dessous différentes approches algorithmiques, classées suivant leur

complexité en ordre dégressif :

- *Le problème de recherche :* Rechercher un vecteur $x \in \Lambda$ tel que $\|x\|$ (respectivement, $\|x - t\|$) soit minimisé ;

- *Le problème d'optimisation :* Trouver le minimum de $\|x\|$ (respectivement, $\|x - t\|$), parmi $x \in \Lambda \setminus \{0\}$ (respectivement parmi $x \in \Lambda$) ;

- *Le problème de décision :* Pour un nombre rationnel $r > 0$ donné, décider si oui ou non il y a un vecteur lattice, non nul x tel que $\|x\| \leq r$ (respectivement $\|x - t\| \leq r$) ;

Selon nos bibliographies toujours, nous savons dans un premier temps que tous les algorithmes SVP et CVP connus (algorithmes s'exécutant en un temps exponentiel pour rappel) permettent la résolution du problème de type recherche, mentionné un peu plus haut. Et résout donc dans ce cas, les deux problèmes à complexité moindre susmentionnés (le problème d'optimisation et de décision). En second lieu, les affirmations en termes de difficultés tiennent pour la version la moins complexe, le problème de décision. Ce qui induit au moins les mêmes niveaux de complexité pour les deux versions plus ardues (le problème de recherche et d'optimisation).

Pour dire tout ça d'une manière plus simple, en faisant intervenir les notions de complexité dans notre paragraphe 3.2.2.4 a, le problème de décision lié aux SVP et CVP est classifié comme NP-complet. Ainsi, aucun algorithme déterministe, s'exécutant en un temps polynomial ne peut résoudre les SVP et CVP. Sauf bien sûr s'il s'avère que $P = NP$ (une question qui reste toujours ouverte, au moment où nous écrivons, pour rappel).

Cette difficulté pour résoudre les SVP et CVP a poussé les fervents du domaine à considérer une version approximative de ces problèmes. Les algorithmes d'approximation retournent des solutions qui sont garanties aux environs d'un facteur de marge γ, partant de l'optimale. Ces versions approximatives des SVP et CVP sont abordées dans nos paragraphes qui vont suivre.

3.3.2.3 SVP approximatif ou Approximative Shortest Vector Problem pour son nom en anglais

Etant donné une base de lattice $B \in \mathbb{Z}^{m \times n}$, le SVP approximatif ou Approximative Shortest Vector Problem pour son nom en anglais, consiste à trouver un vecteur lattice non nul Bx avec $x \in \mathbb{Z}^n \setminus \{0\}$ tel que $\|Bx\| < \gamma \|By\|$ pour tout $y \in \mathbb{Z}^n \setminus \{0\}$.

Avec la version optimisation du SVP approximatif, il est question de trouver uniquement le module de vecteur lattice $\|Bx\|$, c'est-à-dire la valeur d de telle sorte que $\lambda_1(B) \leq d < \gamma\lambda_1(B)$.

3.3.2.4 CVP approximatif ou Approximative Closest Vector Problem pour son nom en anglais

Etant donné une base de lattice $B \in \mathbb{Z}^{m\times n}$, et un vecteur cible $t \in \mathbb{Z}^m$, le CVP approximatif ou Approximative Closest Vector Problem pour son nom en anglais, consiste à trouver un vecteur lattice Bx avec $x \in \mathbb{Z}^n$ tel que $\|Bx - t\| < \gamma\|By - t\|$ pour tout $y \in \mathbb{Z}^n$.

Avec la version optimisation du CVP approximatif, il est question de trouver uniquement le module de vecteur lattice $\|Bx - t\|$, c'est-à-dire la valeur d de telle sorte que $\text{dist}\big(t, \mathcal{L}(B)\big) \leq d < \gamma\text{dist}\big(t, \mathcal{L}(B)\big)$.

Dans les deux cas du SVP et CVP, le facteur d'approximation γ est généralement dépendant des paramètres de la lattice. Typiquement, il (le facteur d'approximation γ) est dépendant du rang n de la lattice, ceci expose le fait que le problème est de plus en plus difficile au fur et à mesure que la valeur de n augmente. Nos bibliographies avancent que les plus avancés des algorithmes d'approximation pour les SVP et CVP, atteignent leurs performances les plus mauvaises se caractérisant par un facteur d'approximation $\gamma(n)$ qui augmente d'une manière exponentielle avec le rang n. Trouver des algorithmes qui se caractérisent par un facteur d'approximation polynomial, $\gamma(n) = n^c$ (pour une constante c non dépendant du rang n) est un défi encore d'actualité pour le présent contexte (SVP et CVP approximatifs pour les lattices).

3.4 Algorithme approximatif

3.4.1 Résolution approximative du SVP dans la dimension n

Dans cette section, nous allons décrire un algorithme de dimension n, qui s'exécute en un temps polynomial, pour trouver un vecteur lattice v, garantit à avoir une norme dans l'ordre des $\gamma(n) = \big(2/\sqrt{3}\big)^n$ comparé à λ_1, c'est-à-dire $\|v\| \leq \gamma(n)\lambda_1$. Le présent algorithme ne trouve forcément donc pas le plus court vecteur lattice. Il (l'algorithme) peut être adapté à tout type de norme (parmi ceux que nous avons évoqué dans notre paragraphe 3.3.1.2), mais pour faire simple, nous allons baser nos écrits sur la norme ℓ_2.

3.4.1.1 Bases réduites

Juste pour rappel, le processus d'orthogonalisation de Gram-Schmidt nous donnait :

$$b_i^* = b_i - \sum_{j<i} \mu_{i,j} b_j^*$$

avec :

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$$

Nous définissons l'opération de projection π_i depuis $\mathbb{R}^m$ sur $\mathrm{span}(b_i^*, b_{i+1}^*, \ldots, b_n^*)$ par :

$$\pi_i(x) = \sum_{j=i}^{n} \frac{\langle x, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} b_j^* \qquad \textbf{(}3.22\textbf{)}$$

Pour tout vecteur $x \in \mathrm{span}(B)$, $\pi_i(x)$ est la composante de x orthogonal aux vecteurs $b_1, \ldots, b_{i-1}$. En particulier, les vecteurs orthogonalisés de Gram-Schmidt peuvent être exprimés comme $b_i^* = \pi_i(b_i)$.

Nous pouvons maintenant définir la base réduite d'une lattice de dimension n. Pour une raison qui sera clarifiée ultérieurement (dans la section dédiée à l'analyse du temps d'exécution du présent algorithme), nous introduisons un paramètre réel $1/4 < \delta < 1$ et paramétrer la définition de base réduite comme ce qui suit.

Une base $B = [b_1, \ldots, b_n] \in \mathbb{R}^{m \times n}$ est *réduite en LLL* (*LLL-reduced* en anglais, et *LLL* pour les initiales *Lenstra–Lenstra–Lovász* pour rappel) selon le paramètre δ (réduite en δLLL en quelque sorte) si :

(1) $|\mu_{i,j}| \leq \frac{1}{2}$ pour tout $i > j$, où $\mu_{i,j}$ sont les coefficients de Gram-Schmidt ;

(2) pour tout pair de vecteurs consécutifs b_i, b_{i+1} :

$$\delta\|\pi_i(b_i)\|^2 \leq \|\pi_i(b_{i+1})\|^2 \qquad \textbf{(}3.23\textbf{)}$$

Pour $\delta = 1$, la définition précédente dit que, pour tout $i = 1, \dots, n-1$, la base à deux dimensions $[\pi_i(b_i), \pi_i(b_{i+1})]$ est réduite. Dans les parties restantes du présent sous-titre, nous allons prouver que la norme du premier vecteur dans une base réduite en δLLL n'est pas trop longue que λ_1.

Lemme 3.01 :

Si $B = [b_1, \dots, b_n] \in \mathbb{R}^{m \times n}$ est une base réduite en δLLL avec $\delta \in]1/4\,,1[$ alors, $\|b_1\| \leq \left(2/\sqrt{4\delta - 1}\right)^{n-1}\lambda_1$. En particulier si $\delta = 1/4 + 3/4^{(n/n-1)}$ alors $\|b_1\| \leq \left(2/\sqrt{3}\right)^n \lambda_1$.

Démonstration :

♣

Nous pouvons noter que si la base est réduite en LLL alors, pour tout i :

$$\begin{aligned}
\delta\|b_i^*\|^2 &= \delta\|\pi_i(b_i)\|^2 \leq \|\pi_i(b_{i+1})\|^2 \\
&= \left\|b_{i+1}^* + \mu_{i+1,i}b_i^*\right\|^2 \\
&= \|b_{i+1}^*\|^2 + \mu_{i+1,i}^2\|b_i^*\|^2 \\
&\leq \|b_{i+1}^*\|^2 + \frac{1}{4}\|b_i^*\|^2
\end{aligned}$$

En réarrangeant un peu les termes, on aurra :

$$\left(\delta - \frac{1}{4}\right)\|b_i^*\|^2 \leq \|b_{i+1}^*\|^2$$

Ainsi, le vecteur orthogonalisé b_i^* aura une norme de plus en plus petite au fur et à mesure que i augmente. Par exemple si $\delta = 3/4$ alors, tout b_{i+1}^* aura une longueur tout au plus $\sqrt{2}$ fois plus

courte que b_i^*. En itérant notre équation précédente j fois, on aura :

$$\left(\delta - \frac{1}{4}\right)^{i-j} \left\|b_j^*\right\|^2 \leq \|b_i^*\|^2 \qquad \textbf{(3.24)}$$

En particulier :

$$\|b_i^*\| \geq \left(\delta - \frac{1}{4}\right)^{\frac{i-1}{2}} \|b_1^*\| \geq \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}} \|b_1\|$$

En utilisant maintenant la borne inférieure de λ_1 énoncée dans le *Théorème 3.01* nous aurons :

$$\lambda_1 \geq \min_i \|b_i^*\| \geq \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}} \|b_1\|$$

Ceci prouve que $\|b_1\|$ est tout au plus $(\delta - 1/4)^{(1-n)/2}$ fois plus long que λ_1. En posant $\delta = 1/4 + 3/4^{(n/n-1)}$, on aura $\|b_1\| \leq \left(2/\sqrt{3}\right)^n \lambda_1$.

♦

3.4.1.2 Algorithme de réduction de base LLL

Nous savons que le premier vecteur de toute base réduite en δLLL (avec $\delta = 1/4 + 3/4^{(n/n-1)}$) est dans le facteur $\left(2/\sqrt{3}\right)^n$ de l'optimal λ_1. Dans cette section nous allons décrire un algorithme pour calculer les bases réduites en LLL. Pour une lattice de dimension 2, l'algorithme de réduction de Gauss (spécifique à la norme ℓ_2) est comme suit :

(1) Etape de réduction : $b_2 \coloneqq b_2 - cb_1$, où :

$$c = \left\lfloor \frac{\langle b_2, b_1 \rangle}{\langle b_1, b_1 \rangle} \right\rceil$$

avec $\lfloor x \rceil = \left\lfloor x + \frac{1}{2} \right\rfloor$ équivaut à l'arrondissement de la valeur $x + \frac{1}{2}$ à l'entier directement inférieur ou égal à elle-même (*round half up* en anglais).

(2) Etape d'échange : si $\|b_1\| > \|b_2\|$ échanger $b_1 \Leftrightarrow b_2$;

(3) Si b_1 et b_2 ne sont pas réduites, il faut répéter ;

Il est à noter qu'après l'étape (1), nous obtenons toujours $|\mu_{2,1}| \leq 1/2$. L'algorithme LLL suit la même logique, en alternant l'étape de réduction, à la suite de laquelle $|\mu_{i,j}| \leq 1/2$ pour tout $i > j$, et l'étape d'échange où les pairs de vecteurs adjacents sont échangés. On doit se référer à l'*Algorithme 3.01* pour ce faire. Dans la suite du présent sous-titre, nous allons expliquer les étapes de l'algorithme précédemment annoncé, et justifier dans la foulée que s'il se termine, la sortie en est correcte.

Algorithme 3.01 : Algorithme de réduction de base en LLL.

Entrée : Base de lattice $B = [b_1, \dots, b_n] \in \mathbb{Z}^{m \times n}$;
Sortie : Une base réduite en LLL pour $\mathcal{L}(\mathrm{B})$;

(boucle) : **pour** $i = 1, \dots, n$
pour $j = i - 1, \dots 1$
$b_i \coloneqq b_i - c_{i,j} b_j$ où $c_{i,j} = \lfloor \langle b_2, b_1 \rangle / \langle b_1, b_1 \rangle \rceil$
si $\delta \|\pi_i(b_i)\|^2 > \|\pi_i(b_{i+1})\|^2$ pour un quelconque i **alors**
échanger b_i et b_{i+1} et revenir à (boucle)
sinon retourner B

Dans l'étape de réduction, nous nous assurons que $|\mu_{i,j}| \leq 1/2$ pour tout $i > j$. Ceci est obtenu suite à une légère modification au processus d'orthogonalisation de Gram-Schmidt. Nous définissons itérativement la séquence de vecteurs $b'_1, \dots b'_n$ où $b'_1 = b_1$ et tout b'_i (pour $i > 1$ donc) est obtenu en soustrayant un multiple entier approprié des b'_j (avec $j < i$) à b_i. Etant donné que B' s'obtient de $B = [b_1, \dots, b_n]$ par une séquence d'opérations élémentaires en colonne à coefficients entiers, les bases B et B' sont équivalentes. La seule opération effectuée sur B' après l'étape de réduction est le réarrangement de l'ordre des colonnes durant l'étape d'échange. Ainsi, à la fin de chaque itération, B est une base de la lattice en entrée.

Il est important de noter que la base orthogonalisée B^* associée à B avant et après l'étape de réduction reste la même, c'est-à-dire la transformation $B \longrightarrow B'$ ne change pas les vecteurs orthogonalisés b_i^*. Cependant, il peut être facilement vérifié qu'après la transformation $B \longrightarrow B'$,

tous les coefficients de Gram-Schmidt $\mu_{i,j}$ pour tout $i > j$ de la nouvelle base B' satisfont $|\mu_{i,j}| \leq 1/2$.

Après l'étape de réduction, et que la condition $|\mu_{i,j}| \leq 1/2$ soit vérifiée donc, nous testons si la seconde propriété des bases réduites en LLL est remplie pour tout couple de vecteurs adjacents b_i et b_{i+1}. Si pour un quelconque i donné nous avons :

$$\delta \|\pi_i(b_i)\|^2 > \|\pi_i(b_{i+1})\|^2 \qquad \textbf{(}3.25\textbf{)}$$

nous échangeons b_i et b_{i+1}. Plusieurs couples de vecteurs adjacents peuvent présenter le même problème, le choix du couple à échanger n'a pas d'importance. Echanger plus d'un couple de vecteurs est même autorisé.

Si un couple de vecteurs a été échangé, la base n'est nécessairement plus réduite en termes de longueur (les coefficients de Gram-Schmidt $|\mu_{i,j}| > 1/2$. Ainsi, il faut revenir à l'étape de réduction et tout recommencer.

Si en un point donné de l'algorithme, après l'étape de réduction, aucun couple de vecteurs adjacents n'a besoin d'être échangé, alors la base B obtenue à la sortie est réduite en LLL.

De plus, comme déjà mentionné plus haut, la base B obtenue à la sortie est équivalente à celle de départ. En effet, la base à la sortie de l'algorithme a été obtenue par une séquence d'opérations élémentaires en colonne.

Ainsi si l'algorithme LLL se termine, la sortie en est une base réduite en LLL. Le fait que l'algorithme se termine toujours est démontré dans le sous-titre qui va suivre. Nous y verrons également que le temps d'exécution est majoré par une valeur polynomiale en n.

3.4.1.3 Analyse du temps d'exécution

Afin de montrer que l'algorithme s'exécute en un temps polynomial, nous devons prouver que le nombre d'itérations est polynomial dépendant de la dimension en entrée, et que chaque itération requiert un temps polynomial. Dans un premier temps, majorons le nombre d'itérations nécessaires.

a. *Majoration du nombre d'itérations*

Le nombre d'itérations effectuées par l'algorithme est égal au nombre de fois que deux vecteurs adjacents sont échangés durant l'étape d'échange. Dans le but de majorer ce nombre, nous associons un nombre entier positif à la base B et prouver qu'à chaque fois que deux vecteurs sont échangés, ce nombre décroit par un facteur δ. Il s'ensuit que le nombre d'itération est logarithmique suivant l'entier associé à la base initiale. Pour $k = 1, \dots, n$, considérons la sous-lattice $\Lambda_k = [b_1, \dots, b_k]$ générée par les premiers k vecteurs de base. Etant donné que Λ_k est une lattice à coordonnées entière, $\det(\Lambda_k)^2$ est un entier positif. L'entier associé à la base B est :

$$d = \prod_{i=1}^{n} \det(\Lambda_k)^2 \qquad \textbf{(}3.26\textbf{)}$$

Notons que l'étape de réduction ne modifie pas la valeur de d parce que les vecteurs orthogonalisés b_i^* ne sont pas modifiés durant cette étape et que chacun des $\det(\Lambda_k)$ peut être exprimé en fonction de $\|b_1^*\|, \dots, \|b_k^*\|$. Nous voulons prouver que quand deux vecteurs b_i et b_{i+1} sont échangés, d diminue d'un facteur δ. Soient d et d' la valeur de d avant et après l'échange. De la même manière, soient Λ_k et Λ'_k les sous-lattices générées par les bases avant et après l'échange. Pour rappel, l'échange a lieu si et seulement si **(**3.25**)** tient. Quand les vecteurs b_i et b_{i+1} sont échangés, $\det(\Lambda_k)$ reste le même pour tout $k \neq i$. Ceci parce que, pour $k < i$, les bases $[b_1, \dots, b_k]$ ne sont pas modifiées, si pour $k > i$, seul l'ordre des deux vecteurs est échangé. Dans les deux cas, les sous-lattices Λ_k et Λ'_k restent inchangées. $\Lambda_k = \Lambda'_k$ et $\det(\Lambda_k) = \det(\Lambda'_k)$. Nous avons :

$$\frac{d}{d'} = \frac{\det(\Lambda'_i)^2}{\det(\Lambda_i)^2} = \frac{\det([b_1, \dots, b_{i-1}, b_{i+1}])^2}{\det([b_1, \dots, b_i])^2}$$

$$= \frac{\left(\prod_{j=1}^{i-1} \|b_j^*\|^2\right) \|\pi_i(b_{i+1})\|^2}{\prod_{j=1}^{i} \|b_j^*\|^2}$$

$$= \frac{\|\pi_i(b_{i+1})\|^2}{\|\pi_i(b_i)\|^2}$$

qui est inférieure à δ d'après **(**3.25**)**. Ceci prouve que d décroit par un facteur δ après chaque

itération. Soit d_0 l'entier associé à la matrice d'entrée, et d_k l'entier associé à B après k itérations. Par récurrence, nous avons :

$$d_k \leq \delta^k d_0$$

Etant donné que d_k est un entier positif, $\delta^k d_0 \geq d_k \geq 1$. Et pour tout $\delta < 1$:

$$k \leq \frac{\ln d_0}{\ln(1/\delta)}$$

Et comme d_0 peut être évalué en un temps polynomial à partir de B, $\ln d_0$ est également polynomial selon la dimension en son entrée. Si δ est fixé à une valeur inférieure à 1, alors $\ln(1/\delta)^{-1}$ augmente le nombre d'itérations par un facteur constant. La valeur de δ peut même être fixée en tant que dépendante de n, augmentant avec celui-ci (n), avec $\lim_{n\to\infty} \delta = 1$. Le nombre d'itérations polynomial sera toujours maintenu.

Lemme 3.02 :

Si $\delta = (1/4) + (3/4)^{n/(n-1)}$, alors pour tout $c > 1$ et tout n suffisamment grand, $\ln(1/\delta)^{-1} \leq n^c$.

Démonstration :

♣

Soit $\delta = (1/4) + (3/4)^{n/(n-1)}$, nous voulons prouver que $\ln(1/\delta)^{-1}$ est tout au plus égal à n^c, ou encore d'une manière équivalente $1 - e^{-(1/n)^c} \leq 1 - \delta$, pour tout n suffisamment grand. Notons que $1 - \delta = (3/4)\left(1 - (3/4)^{1/(n-1)}\right)$. Pour $n \longrightarrow \infty$, on a la formule (1) ci-dessous :

$$\frac{1 - e^{-(1/n)^c}}{(3/4)(1 - (3/4)^{1/(n-1)})}$$

que nous allons prouver être strictement inférieur à 1, pour n suffisamment grand. En posant $x = 1/n + 1$, soit en substituant n par $n = 1 + 1/x$. La formule (1) est équivalente la limite pour $x \longrightarrow 0$, de la formule (2) ci-dessous :

$$\frac{1 - e^{-(x/x+1)^c}}{(3/4)(1 - (3/4)^x)}$$

dont la limite du numérateur et du dénominateur tend vers 0 lorsque x tend vers 0. On peut faire entrer la règle de l'hôpital :

$$\lim_{x\to 0}\frac{1-e^{-(x/x+1)^c}}{(3/4)(1-(3/4)^x)}=\lim_{x\to 0}\frac{\frac{c}{(x+1)^2}\left(\frac{x}{x+1}\right)^{c-1}e^{-\left(\frac{x}{x+1}\right)^c}}{-\frac{3}{4}(3/4)^x\ln\frac{3}{4}}=0$$

♦

b. *Majoration du temps d'exécution de chaque itération*

Nous avons vu dans le paragraphe précédent que le nombre d'opérations nécessaires pour chaque itération était polynomial. Ainsi, Pour démontrer une majoration polynomiale, en termes de temps d'exécution, nous avons besoin de montrer que la taille des nombres impliqués dans les opérations nécessaires dans son ensemble est majorée, d'une manière polynomiale, dépendant de la dimension en entrée. L'algorithme LLL utilise des nombres rationnels, aussi nous avons besoin de majorer à la fois la partie entière et la précision en décimale.

D'après la formule dans le processus usuel d'orthogonalisation de Gram-Schmidt **(**3.12**)**, nous savons que $b_i - b_i^*$ appartient à $\mathrm{span}(b_1, \dots, b_{i-1})$, c'est-à-dire :

$$b_i - b_i^* = \sum_{j=1}^{i-1} v_{i,j} b_j$$

pour certains nombres réels donnés $v_{i,j}$. Notons en passant que ces nombres réels sont différents des coefficients de Gram-Schmidt $\mu_{i,j}$. En particulier $|v_{i,j}|$ peut-être plus grand que $1/2$. Soit $t < i$ et évaluons le produit scalaire de $b_i - b_i^*$ par b_t. Etant donné que b_i^* est orthogonal à b_t, nous avons :

$$\langle b_i, b_t \rangle = \sum_{j=1}^{i-1} v_{i,j} \langle b_j, b_t \rangle$$

Soit $B_t = [b_1, \dots, b_t]$ et $v_i = \left[v_{i,1}, \dots, v_{i,i-1}\right]^T$. En combinant la précédente équation pour tout $t = 1, \dots, i-1$, nous avons :

$$b_i^T B_{i-1} = v_i^T B_{i-1}^T B_{i-1}$$

Ainsi, v_i est la solution d'un système d'équations linéaires avec des coefficients la matrice $B_{i-1}^T B_{i-1}$. Soit $d_{i-1} = \det(B_{i-1}^T B_{i-1}) = \det(\Lambda_{i-1})^2$, où Λ_{i-1} n'est autre que la même sous-lattice définie dans notre paragraphe précédent sur la majoration du nombre d'itérations. D'après la règle de Cramer, $d_{i-1} v_i$ est un vecteur à éléments entiers. Nous utiliserons cette propriété pour majorer les dénominateurs qui peuvent intervenir dans les coefficients $\mu_{i,j}$ et les vecteurs orthogonalisés b_i^*. Notons que :

$$d_{i-1} b_i^* = d_{i-1} b_i + \sum_{j=1}^{i-1} \left(d_{i-1} v_{i,j}\right) b_j$$

est une combinaison entière de vecteurs à coordonnées entières. Ainsi, tous les dénominateurs qui interviennent dans le vecteur b_i^* sont facteur de d_{i-1}. En revenant aux coefficients de Gram-Schmidt :

$$\begin{aligned} \mu_{i,j} &= \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \\ &= \frac{d_{j-1} \langle b_i, b_j^* \rangle}{d_{j-1} \left\| b_j^* \right\|^2} \\ &= \frac{\langle b_i, d_{j-1} b_j^* \rangle}{d_j} \end{aligned}$$

parce que $d_j = \prod_{k=1}^{j} \|b_k^*\|^2$. Ainsi le dénominateur de $\mu_{i,j}$ divise d_j. Ceci démontre que les dénominateurs de tous les nombres rationnels qui interviennent dans l'évaluation divisent $d = \prod_{i=1}^{n} d_i$. Notons que la valeur d mentionnée dans le cas présent n'est autre que celle vue dans la formule **(**3.26**)**. Mais nous savons, de par notre analyse sur le nombre d'itérations dans notre paragraphe précédent que $\log d$ est initialement majoré d'une manière polynomiale dépendant de la dimension en entrée, et que cette valeur décroit tout au long de l'exécution de l'algorithme. Ainsi, les dénominateurs de tous les nombres rationnels qui interviennent durant l'évaluation sont polynomiaux en termes de taille.

Il nous reste à montrer que la partie entière des nombres est également polynomiale. Nous savons déjà que $\mu_{i,j}$ est tout au plus égal à $1/2$ en valeur absolue. Majorons maintenant la longueur des

vecteurs. Notons que pour tout $i > 1$, $\|b_i^*\| \geq 1/d_{i-1}$, parce que $d_{i-1}b_i^*$ est un vecteur non nul à coordonnées entières, et que $\|b_1^*\| = \|b_1\| \geq 1$. En plus, $d_i = \prod_{j=1}^{i}\|b_i^*\|^2$, ainsi :

$$\|b_i^*\|^2 = \frac{d_i}{\prod_{j=1}^{i-1}\left\|b_j^*\right\|^2} \leq d_i \prod_{j=1}^{i-2} d_j^2 \leq d^2$$

Et enfin :

$$\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \left\|b_j^*\right\|^2 \leq d^2 + \frac{n}{4}d^2 \leq nd^2$$

Ceci prouve que toutes les grandeurs intervenant durant l'exécution de l'algorithme LLL peuvent être représentées par des bits dont le nombre est polynomial. Notre preuve que l'algorithme LLL avec $\delta = (1/4) + (3/4)^{n/(n-1)}$ s'exécute en un temps polynomial est désormais complète.

Théorème 3.06 :

Il existe un algorithme à temps polynomial, qui pour une entrée représentée par la base $B \in \mathbb{Z}^{m\times n}$, donne en sortie une base réduite en LLL pour $\mathcal{L}(B)$ avec comme paramètre $\delta = (1/4) + (3/4)^{n/(n-1)}$.

Ensemble avec le *Lemme 3.01*, le théorème précédent nous emmène à un autre, énoncé ci-dessous, sur l'existence d'un algorithme d'approximation à temps polynomial, pour le SVP.

Théorème 3.07 :

Il existe un algorithme à temps polynomial, qui suivant une entrée représentée par une base B, donne à la sortie un vecteur lattice non nul $x \in \mathcal{L}(B) \setminus \{0\}$ de longueur $\|x\| \leq \left(2/\sqrt{3}\right)^n \lambda_1$. Où n est la dimension de la lattice.

3.4.1 Résolution approximative du CVP dans la dimension n

Dans cette section, nous allons montrer comment résoudre approximativement le CVP en utilisant l'algorithme LLL de réduction de base, et ce avec une marge d'erreur égale à $2\left(2/\sqrt{3}\right)^n$. En effet, l'algorithme pour résoudre le CVP est déjà contenu dans la procédure de réduction LLL. L'idée est

comme ce qui suit. Etant donnée une base réduite en LLL $B = [b_1, \dots, b_n]$, et un vecteur cible t, procéder à l'algorithme LLL de réduction de base avec en entrée $[B, t]$ comme si nous voulions additionner t aux vecteurs de base de la lattice. Ainsi, nous trouvons un vecteur lattice $x \in \mathcal{L}(B)$ de telle sorte que $t - x$ puisse être exprimé comme $t^* + \sum_{i=1}^{n} c_i b_i^*$ où t^* est la composante de t orthogonal au $\text{span}(b_1, \dots, b_n)$ et $|c_i| \leq 1/2$ pour tout $i = 1, \dots, n$. Nous pouvons montrer que la distance entre x et t est dans le facteur $2\left(2/\sqrt{3}\right)^n$ de l'optimal.

Algorithme 3.02 : Algorithme du plan le plus proche (*nearest plan algorithm* en anglais).

Entrée : Base de lattice $B \in \mathbb{Z}^{m \times n}$ et un vecteur cible $t \in \mathbb{Z}^m$;
Sortie : Un vecteur lattice $x \in \mathcal{L}(B) \setminus \{0\}$ telle que $\|t - x\| \leq 2\left(2/\sqrt{3}\right)^n \text{dist}(t, \mathcal{L}(B))$;

Exécuter l'algorithme LLL sur B ;
Poser $b \coloneqq t$
pour $j = n, \dots, 1$
    $c_j = \lfloor \langle b, b_j \rangle / \langle b_j, b_j \rangle \rceil$
    $b \coloneqq b - c_j b_j$
retourner $t - b$

L'algorithme qui résout d'une manière approximative le CVP est détaillé dans l'algorithme *Algorithme 3.02*. Le temps d'exécution de cet algorithme est polynomial. Dans le lemme qui va suivre, nous allons prouver que ledit algorithme s'achève avec un facteur d'approximation de $2\left(2/\sqrt{3}\right)^n$. Pour une raison que nous allons voir dans la section de démonstration du lemme, cet algorithme est appelé *algorithme du plan le plus proche* (*nearest plan algorithm* en anglais).

Lemme 3.03 :

Quand $\delta = (1/4) + (3/4)^{n/(n-1)}$, l'algorithme du plan le plus proche résout le CVP dans un facteur $\gamma(n) = 2\left(2/\sqrt{3}\right)^n$.

Démonstration :

♣

Assumons que B soit une base déjà réduite en LLL. Assumons également que le vecteur cible t appartienne au $\text{span}(B)$. Sinon, projeter t sur $\text{span}(B)$ et travailler sur cette composante projetée. La preuve s'obtient de par la propriété de la base B réduite en LLL de la lattice de dimension n. En Effet, si $B = [b_1, \dots, b_n]$ est une base réduite en LLL, alors $[b_1, \dots, b_k]$ l'est également pour $k = 1, \dots, n$. Soit $B^* = [b_1^*, \dots, b_n^*]$ la base B obtenue après le processus d'orthogonalisation de Gram Schmidt.

Alors l'algorithme permettant de trouver le plan le plus proche, *Algorithme 3.02*, ou encore **Figure 3.08** si représenté sous forme graphique, peut être décrit comme suit :

- Trouver un entier c de telle sorte que l'hyperplan $cb_n^* + \text{span}(b_1, \dots, b_{n-1})$ soit le plus proche possible de t ;

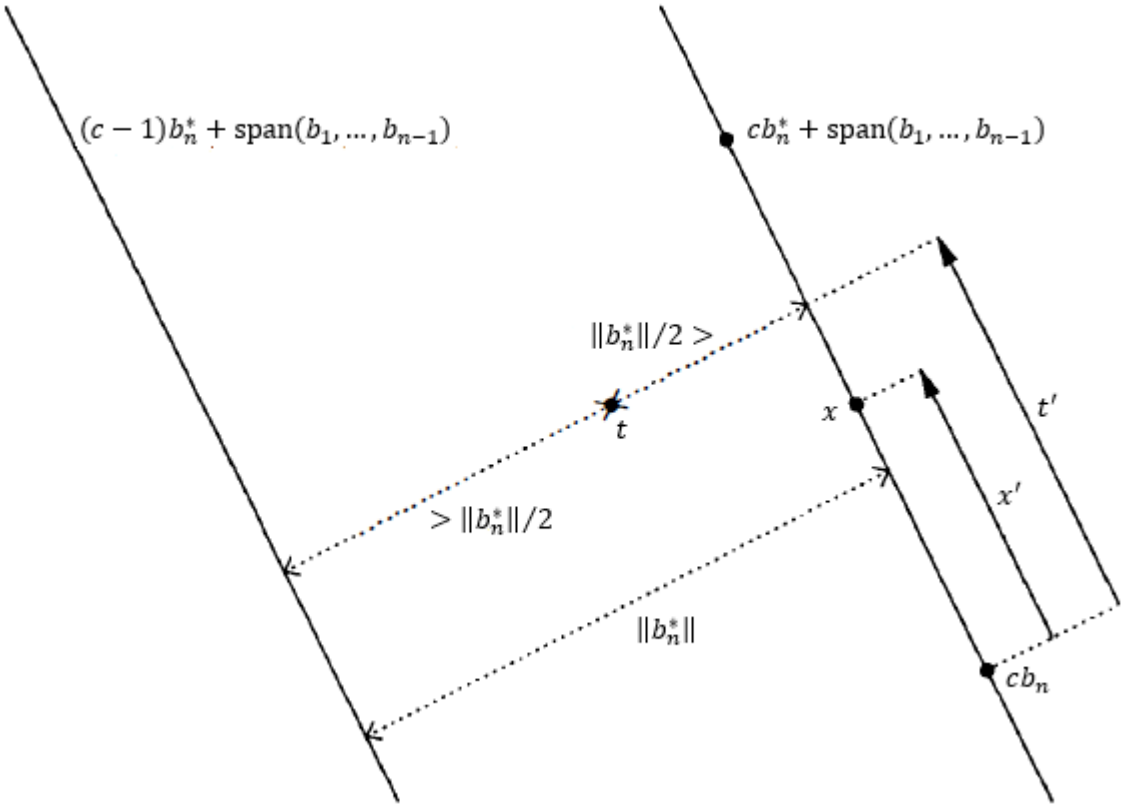


Figure 3.08 : *Algorithme du plan le plus proche.*

- Trouver d'une manière récursive un point (ou vecteur) $x' \in \mathcal{L}(b_1, \dots, b_{n-1})$ approximativement proche de la projection t' de $t - cb_n$ sur $\text{span}(b_1, \dots, b_{n-1})$;

- Retourner $x = x' + cb_n$;

Soit $y \in \mathcal{L}(B)$ le point lattice le plus proche de t. Nous voulons prouver que $\|t - x\|$ est tout au plus $2\left(2/\sqrt{3}\right)^n \|t - y\|$. Il y a 2 cas.

<u>*Cas 1 :*</u> Si $\|t - y\| < \|b_n^*\|/2$, alors y appartient nécessairement à l'hyperplan $cb_n^* + \text{span}(b_1, \dots, b_{n-1})$ parce que chaque hyperplan est $\|b_n^*\|$ plus loin les uns des autres. Ainsi, tout autre hyperplan est distancié de t d'une valeur supérieure à $\|b_n^*\|$. Par conséquent, $y' = y - cb_n$ est le point lattice dans $\mathcal{L}([b_1, \dots, b_{n-1}])$ le plus proche de t' et d'une manière récursive, un point lattice x' à distance dans la distance $2\left(2/\sqrt{3}\right)^{n-1}\|t' - y'\| = 2\left(2/\sqrt{3}\right)^{n-1}\|t - y\|$ from $t - cb_n$. Il s'ensuit que $x' + cb_n$ est dans la distance $2\left(2/\sqrt{3}\right)^{n-1}\|t - y\|$ de t.

<u>*Cas 2 :*</u> Cette fois-ci, nous assumons que $\|t - y\| \geq \|b_n^*\|/2$. Nous utiliserons les propriétés de base réduite en LLL pour montrer que le vecteur lattice trouvé par l'algorithme du plan le plus proche satisfait $\|t - x\| \leq (4/3)^n \|b_n^*\|^2$ et ainsi :

$$\|t - x\| \leq 2\left(2/\sqrt{3}\right)^n \|t - y\|$$

Nous savons que, $\|t - x'\| = \sum_{i=1}^{n} \mu_i b_i^*$ pour un nombre réel donné $\mu_i \leq 1/2$. Rappelons que d'après l'inégalité de notre formule **(**3.24**)**, les vecteurs orthogonalisés dans une base réduite en LLL satisfont $\|b_i^*\| \leq \alpha \|b_{i+1}^*\|$ où $\alpha = 2/\sqrt{4\delta - 1}$ et par itération en $n - i$, $\|b_i^*\| \leq \alpha^{n-i}\|b_n^*\|$. Ainsi :

$$\begin{aligned} \|t - x'\|^2 &= \sum_{i=1}^{n} \mu_i^2 \|b_i^*\|^2 \\ &\leq \frac{1}{4} \sum_{i=1}^{n} \alpha^{2(n-i)} \|b_n^*\|^2 \\ &= \frac{1}{4} \frac{\alpha^{2n} - 1}{\alpha^2 - 1} \|b_n^*\|^2 \end{aligned}$$

$$= \frac{\alpha^{2(n-1)}}{4}\left(1 + \frac{1-\alpha^{2(1-n)}}{\alpha^2 - 1}\right)\|b_n^*\|^2$$

De par le fait que $\alpha = 2/\sqrt{4\delta - 1}$ et que $\delta = (1/4) + (3/4)^{n/(n-1)}$, nous obtenons $\alpha^{n-1} = \left(2/\sqrt{3}\right)^n$, $\alpha^{2(1-n)} = (3/4)^n$ et que $\alpha^2 = (4/3)^{1+1/(n-1)}$. Qui, quand substituée dans la dernière équation, donne :

$$\|t - x\|^2 \leq \frac{1}{4}\left(\frac{4}{3}\right)^n \left(1 + \frac{1-\left(\frac{3}{4}\right)^n}{\left(\frac{4}{3}\right)^{1+\frac{1}{n-1}} - 1}\right)\|b_n^*\|^2 \leq \left(\frac{4}{3}\right)^n \|b_n^*\|^2$$

♦

3.5 Conclusion

Ce chapitre nous a permis de s'étaler sur deux sujets importants. D'une part, la menace qu'apporte les ordinateurs quantiques aux cryptosystèmes largement utilisés en ce moment. D'autre part, nous avons développé les basiques de la lattice, outil mathématique le plus promettant sur lequel se base bon nombre de candidats aux cryptosystèmes post-quantiques, ainsi que deux problèmes mathématiques fondamentaux qui lui sont relatifs.

Pour la première partie, nous avons vu que l'algorithme de Shor apporte une accélération exponentielle pour casser tout cryptosystème à clé publique/privée. L'algorithme de Grover, pour sa part, apporte une accélération quadratique pour résoudre le problème mathématique des chiffres symétriques (la permutation suivant une clé donnée par exemple). Doubler la longueur de la clé permet d'outrepasser cette accélération quadratique.

Dans la seconde partie, nous avons vu qu'une lattice n'est autre que l'ensemble discret constitué par les points de formes géométriques du genre treillis (ceci étant son non en français, malheureusement un peu méconnu). Les deux problèmes les plus connus de la lattice sont le SVP pour Short Vector Problem, et le CVP pour Closest Vector Problem. A ce jour, aucun algorithme déterministe s'exécutant en un temps polynomial n'existe pour résoudre les SVP et CVP. L'algorithme LLL, de leur créateur Lenstra–Lenstra–Lovász, permet d'en résoudre des versions approximatives.

CHAPITRE 4
MLWE OU LEARNING-WITH-ERRORS IN MODULE LATTICES – OUTIL MATHEMATIQUE DU CRYPTOSYSTEME CRYSTALS KYBER/DILITHIUM

4.1 Introduction

Tout au long de l'avancement de notre présente étude, nous avons suivi de près l'évolution du projet *Post-Quantum Cryptography Standardization* ou *Standardisation des cryptographies post-quantiques*, pour son nom en français, organisé par le *NIST* (National Institute of Standards and Technology).

Ledit projet consiste en une soumission des cryptosystèmes (signatures et cryptosystème à proprement parler, encore dénommé KEM ou Key Encapsulation Mechanism pour rappel, comme vu dans notre précédent chapitre) en provenance de tous les chercheurs du domaine. La soumission initiale a pris fin en fin 2017, si les plus prometteurs d'entre eux (les cryptosystèmes) ont dernièrement (fin juillet 2020) avancé en phase 03 du concours. Ces cryptosystèmes arrivés en phase 03 entamerons donc l'étape de standardisation, but final du projet. [4.05]

Pour notre part, nous nous intéresserons de près au cyptosystème *CRYSTALS* ou *Cryptographic Suite for Algebraic Lattices*. Nous nous sommes permis de prendre cette décision du fait que ledit cryptosystèmes s'est trouvé être en mesure d'avoir réussi à faire avancer en phase 03 du concours NIST, mentionné plus haut, un cryptosystème de type KEM et un autre de type signature, respectivement dénommés *CRYSTALS Kyber* et *CRYSTALS Dilithium*.

Les algorithmes composant ces deux cryptosystèmes (CRYSTALS Kyber et CRYSTALS Dilithium) sont basés sur des problèmes difficiles autour des *module lattices* (notons que ces deux termes sont leurs versions anglaises). Ces objets mathématiques (problèmes difficiles autour des module lattices) ont été conçus pour résister aux éventuelles attaques en provenance des ordinateurs quantiques.

Les module lattices peuvent être considérés comme des lattices situées entre ceux utilisés dans les définitions du problème *LWE* pour *Learning With Error* et ceux utilisés pour le problème *ring-LWE*. Pour avant-goût dans l'explication du choix de module lattices par les créateurs du cryptosystème, ces outils mathématiques (les module lattices) ont l'avantage de préserver la difficulté offerte par les ring-LWE, mais présentant la facilité d'implémentation liée au LWE.

Dans nos paragraphes qui vont suivre, nous allons développer ce que représente les outils mathématiques précédemment mentionnés : le LWE, le ring-LWE pour finir avec le MLWE ou encore Module LWE.

4.2 LWE ou Learning With Error

4.2.1 Préliminaires

4.2.1.1 Notations en Big O

Une définition formelle de la notation en Big O en est que celle-ci (notation Big O) est une notation mathématique qui décrit le comportement aux limites d'une fonction quand l'argument tend vers infini ou vers une valeur particulière. Big O fait partie de la famille de notations inventées par *Paul Bachmann* et *Edmund Landau* entre autres.

Nous énumèrerons dans le tableau qui va suivre (**Tableau 4.01**) les notations en Big O qui interviennent dans le présent chapitre en particulier.

Tableau 4.01 : *Quelques notations en Big* O.

Notation	Nom	Description	Description formelle
$f(n) = O\big(g(n)\big)$;	Big O ;	$\lvert f\rvert$ est majorée, asymptotiquement, suivant un facteur constant donné, par g ;	$\exists k > 0, \exists n_0, \forall n > n_0 : \lvert f(n)\rvert \leq k g(n)$;
$f(n) = \Theta\big(g(n)\big)$;	Big Theta ;	f est à la fois majorée et minorée, asymptotiquement par g ;	$\exists k_1 > 0, \exists k_2 > 0, \exists n_0, \forall n > n_0 : k_1 g(n) \leq f(n) < k_2 g(n)$;
$f(n) = O(n^k) =$ poly(n) ;	Polynomial en n ;	$\lvert f\rvert$ est majorée, asymptotiquement, par une fonction polynomiale en n ;	$\exists k > 0, \exists n_0, \forall n > n_0 : \lvert f(n)\rvert \leq n^k$;
$f(n) = \tilde{O}(n)$;	Big O tilde ;	$\lvert f\rvert$ est majorée par n, avec un facteur logarithmique ignoré dans la notation ;	Equivalent à $O(n \log n)$, ou encore $O(n \log n \log\log n)$, ou encore $O(n \log^k n)$;

$f(n) = \omega(g(n))$	Small Omega ;	f domine g, asymptotiquement parlant ;	$\forall k > 0, \exists n_0, \forall n > n_0 : \lvert f(n)\rvert > kg(n)$;
$f(n) = \Omega(g(n))$;	Big Omega dans la théorie des nombres (Hardy–Littlewood)	$\lvert f\rvert$ n'est pas dominée par g, asymptotiquement parlant	$\exists k > 0, \exists n_0, \forall n > n_0 : \lvert f(n)\rvert \geq kg(n)$;

4.2.1.2 Dual lattice

La *dual lattice* (ou *treillis double* pour son nom en français), dénoté L^*, d'une lattice L dans $\mathbb{R}^n$, est une lattice formée par l'ensemble de tous le vecteurs $y \in \mathbb{R}^n$ de telle sorte que $\langle x, y\rangle \in \mathbb{Z}$ pour tout vecteur $x \in L$.

D'une manière similaire, étant donné une base $(v_1, \ldots, v_n)$ d'une lattice, nous définissons par *dual basis* l'ensemble de vecteurs $(v_1^*, \ldots, v_n^*)$ de telle sorte que $\langle v_i, v_j^*\rangle = \delta_{ij}$ pour tout $i, j \in [n]$: Où δ_{ij} dénote le *Kronecker delta* (*symbole de Kronecker* en français), c'est-à-dire 1 si $i = j$ et 0 sinon.

Par abus de notation, la lattice L est souvent représentée par une matrice $n \times n$ dont les colonnes sont constituées par les vecteurs de base $(v_1, \ldots, v_n)$. Avec cette notation, il est à noter que $L^* = (L^T)^{-1}$. De par cela, il s'ensuit que $\det(L^*) = 1/\det(L)$.

4.2.1.3 Transformation de Fourier d'une probabilité de distribution D sur une lattice L

La discrete Gaussian distribution $D_{L,r}$ (un nommage plus succinct, comparé au contenu de notre présent sous-titre « probabilité de distribution D ») est détaillée dans notre paragraphe plus loin 4.2.4.1 . Nous invitons nos lecteurs à faire un saut là-bas pour une meilleure compréhension.

Considérons donc une probabilité de distribution D sur une lattice L et sa transformée de Fourier $f: \mathbb{R}^n \longrightarrow \mathbb{C}$ définie par :

$$f(x) = \sum_{y \in L} D(y) \exp(2\pi i \langle x, y\rangle) = \operatorname*{Exp}_{y \sim D}[\exp(2\pi i \langle x, y\rangle)] \qquad \textbf{(}4.01\textbf{)}$$

Où dans la seconde égalité, nous avons simplement écrit la somme en tant qu'espérance (moyenne

ou encore expectation en anglais. D'où le diminutif Exp[.] d'ailleurs).

Par définition, f est L^*-périodic. C'est-à-dire, $f(x) = f(x + y)$ pour tout $x \in \mathbb{R}^n$ et $y \in L^*$. Il a été montré [4.03] qu'étant donné un nombre polynomial d'echantillons à partir de D, on peut calculer une approximation de f avec une précision de $\pm 1/\mathrm{poly}(n)$. Pour voir cela, notons que de par l'*inégalité de Chernoff-Hoeffding* (*Chernoff-Hoeffding bound* en anglais), si $y_1, \ldots, y_N$ sont $N = \mathrm{poly}(n)$ échantillons indépendants, alors :

$$f(x) \approx \frac{1}{N}\sum_{j=1}^{N} \exp\big(2\pi i\langle x, y_j\rangle\big) \qquad \textbf{(}4.02\textbf{)}$$

où l'approximation est de l'ordre de $\pm 1/\mathrm{poly}(n)$ et tient avec une probabilité exponentiellement proche de 1, assumant que N est de valeur assez grande, polynomialement parlant.

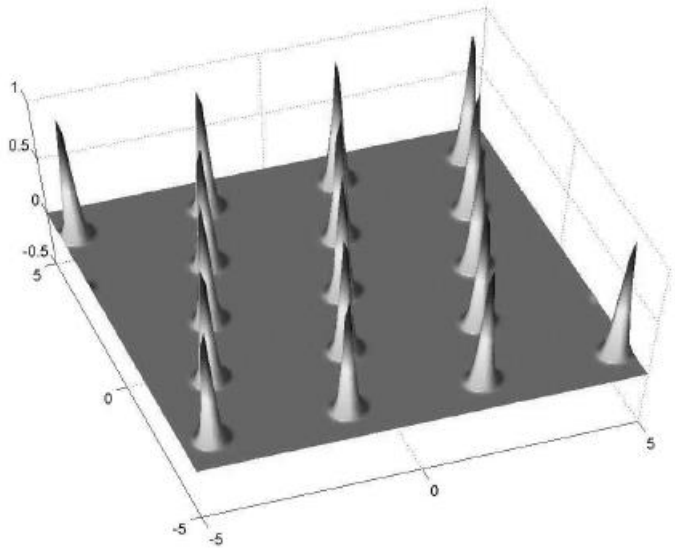


Figure 4.01 : *$f_{1/r}$ pour une lattice de dimension égale à* 2.

En appliquant cette idée aux échantillons à partir de $D_{L,r}$, échantillons qui nous ont été fournis en tant que variables en entrée, nous obtenons une bonne approximation de la transformation de Fourier de $D_{L,r}$, que nous allons dénoter par $f_{1/r}$. Il peut être montré qu'étant donné que $1/r \ll \lambda_1(L^*)$, on

arrive l'approximation :

$$f_{1/r}(x) \approx \exp\left(-\pi\left(r.\,\mathrm{dist}(L^*, x)\right)^2\right) \quad \textbf{(4.03)}$$

Ainsi, $f_{1/r}(x) \approx 1$ pour tout $x \in L^*$, et quand on s'éloigne de L^*, sa valeur diminue (se référer à la figure **Figure 4.01** pour la représentation visuelle). Pour tout point à une distance égale à $1/r$ de la lattice, sa valeur est toujours un constant positif (à peu près $\exp(-\pi)$) . Et quand la distance par rapport à L^* croît, la valeur de la fonction devient rapidement négligeable. Etant donné que la distance entre deux vecteurs dans L^* est au moins $\lambda_1(L^*) \gg 1/r$, le Gaussien autour de chaque point de L^* est bien séparé.

4.2.2 Problème learning from parity with error

Pour un entier $n \geq 1$ et un nombre réel $\varepsilon \geq 0$, considérons le problème *learning from parity with error* dont la description est comme ce qui va suivre. Le but est de trouver un inconnu $s \in \mathbb{Z}_2^n$ étant donné une liste d'équations avec erreurs :

$$\begin{aligned} \langle s, a_1 \rangle &\approx_\varepsilon b_1 (\mathrm{mod}\ 2) \\ \langle s, a_2 \rangle &\approx_\varepsilon b_2 (\mathrm{mod}\ 2) \\ &\vdots \end{aligned} \quad \textbf{(4.04)}$$

Où les a_i sont choisis indépendamment d'après une distribution uniforme sur $\mathbb{Z}_2^n$, $\langle s, a_i \rangle = \sum_j s_j (a_i)_j$ est le produit scalaire modulo 2 de s et a_i, et que toutes les équations sont correctes indépendamment avec une probabilité $1 - \varepsilon$.

Plus précisément, l'entrée du problème consiste en de pairs (a_i, b_i) où chaque a_i est choisi indépendamment et uniformément dans $\mathbb{Z}_2^n$ et chaque b_i est également choisi indépendamment pour être égal au produit scalaire $\langle s, a_i \rangle$ avec une probabilité $1 - \varepsilon$. Le but est de trouver s. Notons que, avec $\varepsilon = 0$, le système d'équations peut être résolu par élimination Gaussienne. Cela nécessite $O(n)$ équations et un temps de résolution de l'ordre de $\mathrm{poly}(n)$.

Le problème commence à être significativement plus difficile lorsque nous prenons $\varepsilon > 0$. Par exemple, considérons encore une fois l'élimination Gaussienne mentionnée plus haut, et assumons

que nous aimerions juste récupérer le premier bit de s. En utilisant l'élimination Gaussienne, nous avons besoin de trouver un ensemble S de $O(n)$ équations de telle sorte que $\sum_S a_i = (1,0,\dots,0)$. En sommant les valeurs correspondantes de b_i, nous obtenons une valeur probable du premier bit de s. Cependant, un calcul standard [4.03] nous montre que cette valeur est correcte avec une probabilité de $\frac{1}{2} + 2^{-\Theta(n)}$. Ainsi, dans le but d'obtenir le premier bit avec une bonne certitude, nous avons besoin de répéter toutes ces étapes $2^{\Theta(n)}$ fois. Ceci nous conduit à un algorithme faisant intervenir $2^{O(n)}$ équations et $2^{O(n)}$ temps. En effet, notre bibliographie [4.03] toujours, nous montre qu'étant donné uniquement $O(n)$ équations, le $s' \in \mathbb{Z}_2^n$ qui maximise le nombre d'équations satisfaites n'est, avec une forte probabilité, autre que s. Ceci nous emmène à un algorithme simple, de maximum vraisemblance (*maximum likelihood algorithm*, pour son nom en anglais) qui requiert uniquement $O(n)$ équations, et s'exécute en un temps $2^{O(n)}$.

Une question ouverte est d'expliquer la difficulté apparente à trouver des algorithmes efficaces pour ce problème d'apprentissage (learning problème, pour sa version anglaise). Un théorème qui sera énoncé dans le paragraphe qui suit, expliquera cette difficulté, pour une extension du problème à de moduli plus élevés.

4.2.3 Problème Learning With Error

Soit $p = p(n) \leq \text{poly}(n)$ un nombre entier premier, et considérons la liste d'équations ci-dessous :

$$\begin{aligned} \langle s, a_1 \rangle &\approx_\chi b_1 (\text{mod } p) \\ \langle s, a_2 \rangle &\approx_\chi b_2 (\text{mod } p) \\ &\vdots \end{aligned} \qquad \textbf{(4.05)}$$

Où cette fois-ci, $s \in \mathbb{Z}_p^n$, a_i sont indépendamment et uniformément choisis dans $\mathbb{Z}_p^n$, et $b_i \in \mathbb{Z}_p$. L'erreur dans l'équation est maintenant spécifiée par une fonction de distribution de probabilité $\chi: \mathbb{Z}_p \longrightarrow \mathbb{R}^+$ dans $\mathbb{Z}_p$. Ainsi, pour toute équation i, on a $b_i = \langle s, a_1 \rangle + e_i$ où les $e_i \in \mathbb{Z}_p$ sont choisis indépendamment suivant χ. Ce problème consistant en la recherche de s à partir des équations précédemment mentionnées, est dénommé par $LWE_{p,\chi}$ (Learning With Error). Par exemple, le problème de learning from parity with error détaillé dans notre paragraphe 4.2.1, faisant intervenir l'erreur ε, est un cas spécial du $LWE_{p,\chi}$ avec $p = 2$, $\chi(0) = 1 - \varepsilon$ et $\chi(1) = \varepsilon$.

Sous une hypothèse raisonnable concernant la fonction χ (notamment que celle-ci justifie la condition $\chi(0) > 1/p + 1/\text{poly}(n)$), l'algorithme de maximum vraisemblance, décrit dans notre paragraphe précédent, résout également le $LWE_{p,\chi}$ pour $p \leq \text{poly}(n)$ en utilisant $\text{poly}(n)$ équations et $2^{O(n \log n)}$ temps [4.03]. Encore une fois, sous une même hypothèse que celle énoncée dans notre paragraphe précédent, toujours, un algorithme ressemblant à celui de Blum et Al. requiert seulement $2^{O(n)}$ équations/temps.

Le théorème qui va suivre montre que pour certains choix sur p et χ, une solution au $LWE_{p,\chi}$ implique une solution quantique pour les problèmes de lattice, dans leurs plus bas performance (niveau de difficulté le plus élevé).

Théorème 4.01 :

Soient n et p deux entiers, et $\alpha \in (0,1)$ (équivalent à $\alpha \in]0,1[$ pour rappel) tels que $\alpha p > 2\sqrt{n}$. S'il y a un algorithme efficace qui peut résoudre le $LWE_{p,\bar{\Psi}_\alpha}$, alors il existe un algorithme quantum efficace pour approcher la version décision du SVP (dénoté également GapSVP) et le Shortest Independent Vector Prolem (SIVP) en $\tilde{O}(n/\alpha)$ dans le cas le plus défavorable. Notons le SIVP équivaut à la recherche de, vecteurs cette fois-ci, indépendants dont les longueurs sont respectivement les successives minimas que nous avons énoncés dans notre précédent chapitre (chapitre 3).

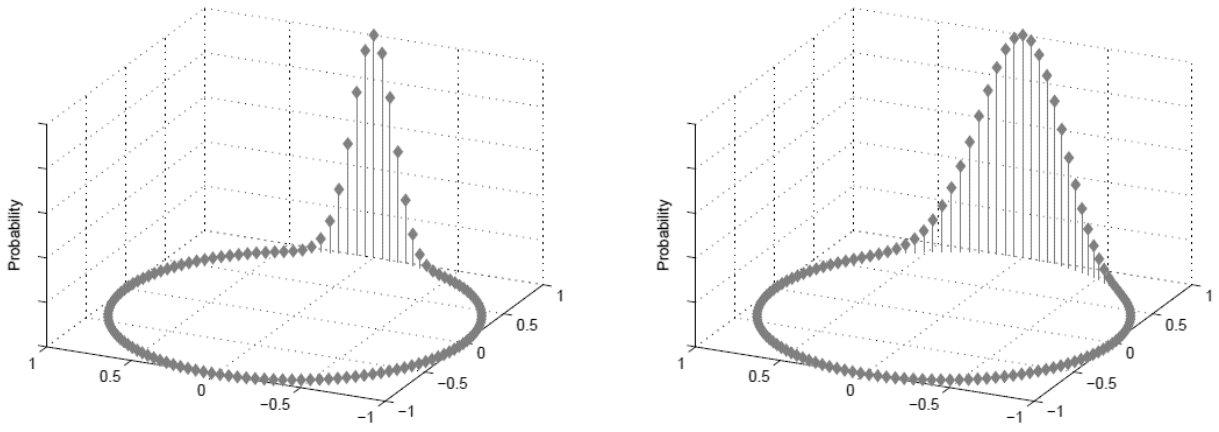


Figure 4.02 : $\bar{\Psi}_\alpha$ *pour* $p = 127$ *et* $\alpha = 0{,}05$ *(à gauche) et* $\alpha = 0{,}1$ *(à droite). Les éléments de* $\mathbb{Z}_p$ *sont arrangés sur un cercle.*

Pour la définition de $\overline{\Psi}_\alpha$, il est pour le moment suffisant de savoir que c'est une distribution sur $\mathbb{Z}_p$ qui a la forme d'un Gaussien discret centré sur 0 avec un écart type αp, comme montré en **Figure 4.02**. Une possible valeur des paramètres est $p = O(n^2)$ et $\alpha = 1/\sqrt{n}\log^2 n$.

GapSVP et SIVP sont deux des problèmes de calcul fondamentaux autour des lattices. Pour le GapSVP, pour rappel, l'entrée est une lattice, et la sortie attendue est de se rapprocher de la longueur du plus court vecteur lattice non nul. Le plus performant des algorithmes polynomiaux qui résout le problème se caractérise par des facteurs d'approximation sub-exponentiels. Il est conjoncturé qu'il n'y a pas d'algorithme classique (c'est-à-dire non-quantique) qui s'exécute en temps polynomial, et effectue l'approximation avec un facteur polynomial.

On pourrait en addition conjoncturer qu'il n'y a pas d'algorithme quantique à temps polynomial qui approche le GapSVP (ou le SIVP) en respect à un facteur polynomial. On pourrait ainsi interpréter le *Théorème 4.01* plus haut pour dire que le LWE est difficile. La seule évidence qui supporte cette conjoncture est qu'il n'y a pas d'algorithme quantique connu pour les problèmes relatifs aux lattices qui peuvent mieux faire que les algorithmes classiques.

En fait, on pourrait aussi interpréter le théorème *Théorème 4.01* plus haut comme un moyen de réfuter cette conjecture : si on trouve un algorithme efficace pour LWE, alors on obtient aussi un algorithme quantique pour approximer les problèmes de lattice dans le cas le plus défavorable.

4.2.4 Aperçu sur la démonstration du Théorème 4.01

Dans le présent sous-titre, nous allons étaler un aperçu de la démonstration du *Théorème 4.01*. Pour la démonstration complète, nous renvoyons nos lecteurs dans notre bibliographie [4.03].

Pour ce faire, on aura, dans un premier temps, besoin de quelques notions autour de la lattice. Nos lecteurs peuvent revisiter notre chapitre précédent (chapitre 3) au cas où des éclaircissements sont nécessaires. Dans un second lieu, une nouvelle notion, la *distribution Gaussienne discrète* (*discrete Gaussian distribution* pour sa version anglaise), sera utilisée à maintes reprises.

4.2.4.1 Discrete Gaussian distribution $D_{L,r}$ et smoothing parameter $\eta_\varepsilon(L)$

Le *discrete Gaussian distribution* sur L et de rayon r, dénoté $D_{L,r}$, est une distribution dont le support est L (qui est dans la majorité des cas une lattice), et dans laquelle, la probabilité de chaque $x \in L$

est proportionnelle à $\exp(-\pi\|x/r\|^2)$. Se référer à la **Figure 4.03**.

Il est nécessaire également de faire la mention de $\eta_\varepsilon(L)$ qui est un *paramètre de lissage* (*smoothing parameter* en anglais). C'est un nombre positif réel associé à toute lattice L (le paramètre de précision ε peut-être ignorer pour le moment). Ce paramètre donne le rayon r à partir duquel $D_{L,r}$ se comporte comme une *distribution Gaussienne continue* (*continuous Gaussian distribution* pour son nom en anglais). Par exemple, pour $r \geq \eta_\varepsilon(L)$, les vecteurs de $D_{L,r}$ ont approximativement une norme égale à $r\sqrt{n}$ avec une très forte probabilité. Et au contraire, pour un r suffisamment petit, $D_{L,r}$ concentre la majorité de sa masse au point origine 0.

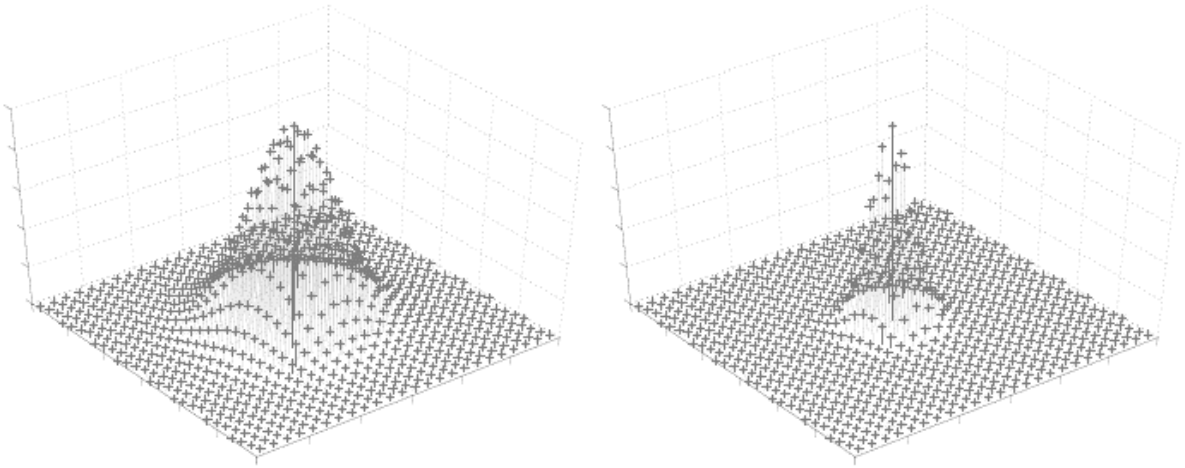


Figure 4.03 : *$D_{L,2}$ (à gauche) et $D_{L,1}$ (à droite) pour une lattice de dimension 2. L'axe des z représente la probabilité.*

4.2.4.2 Problème discret Gaussian sampling (DGS)

Soient α, p, n tels que $\alpha p > 2\sqrt{n}$ comme requis dans le *Théorème 4.01*. Et assumons que nous avons un oracle qui résout le LWE_{p,Ψ_α}. Nous pouvons penser à une valeur de $p = n^2$ et $\alpha = 1/n$. Notre but est de montrer comment résoudre les deux problèmes relatifs aux lattices, mentionnés dans notre *Théorème 4.01*. Dans notre bibliographie [4.03], sous-section 3.3, faisant intervenir des réductions standards, cela suffit à résoudre le problème *discret Gaussian sampling* (DGS), se définissant comme suit : étant donné une lattice L de dimension n, et un nombre $r \geq \sqrt{2n}.\eta_\varepsilon(L)/\alpha$, générer des échantillons à partir de $D_{L,r}$. D'une manière intuitive, la connexion au GapSVP et SIVP vient au fait de prendre r près de sa limite inférieure $\sqrt{2n}.\eta_\varepsilon(L)/\alpha$, nous pouvons prendre un court

vecteur lattice (de longueur de l'ordre de $\sqrt{n}r$).

Dans les sous-titres qui vont suivre, nous allons décrire l'algorithme d'échantillonnage à partir de $D_{L,r}$. Il est important de mentionner que l'exacte borne inférieure de r n'est pas importante dans le cadre de cet aperçu. En effet, cela affecte seulement le facteur d'approximation que nous obtenons pour le GapSVP et le SIVP. Il suffit de garder en tête que notre but est d'échantillonner à partir de $D_{L,r}$ pour r assez petit, disons un facteur polynomial de $\eta_\varepsilon(L)$.

4.2.4.3 Algorithme d'échantillonnage à partir de $D_{L,r}$

Le cœur de l'algorithme s'effectue par étapes itératives. L'entrée est constituée d'un nombre r (qui est garanti à ne pas être trop petit, en particulier supérieur à $\sqrt{2p}.\eta_\varepsilon(L)$), et n^c échantillons à partir de $D_{L,r}$ où c est une constante donnée. La sortie est un échantillon à partir de la distribution $D_{L,r'}$ pour $r' = r\sqrt{n}/(\alpha p)$. Etant donné que $\alpha p > 2\sqrt{n}$, $r' < r/2$. Dans le but de pouvoir faire cette magie de conversion de vecteur de longueur $\sqrt{n}r$ en un autre vecteur de longueur plus courte $\sqrt{n}r'$, le présent algorithme a besoin de l'oracle qui peut résoudre le LWE.

Etant donnée l'étape itérative, l'algorithme pour résoudre le DGS fonctionne comme ce qui va suivre. Soit $r_i = r\left(\alpha p/\sqrt{n}\right)^i$. L'algorithme commence par produire n^c échantillons à partir de $D_{L,r_{3n}}$. Comme r_{3n} est assez longue en termes de normes, il existe (nos lecteurs peuvent se référer au Lemme 3.2 dans notre bibliographie [4.03] toujours) une procédure simple pour le faire. Le corps de l'algorithme se poursuit ainsi : pour $i = 3n, 3n-1, \ldots, 1$, l'algorithme utilise ces n^c échantillons à partir de D_{L,r_i} pour produire n^c échantillons à partir de $D_{L,r_{i-1}}$ en appelant l'étape itérative n^c fois. Eventuellement, nous arrivons à n^c échantillons à partir de $D_{L,r_0} = D_{L,r}$ et l'algorithme se termine par l'impression de l'un des deux.

L'étape itérative quant à elle, peut être décrite comme suit. Rappelons qu'en entrée, nous avons n^c échantillons à partir de $D_{L,r}$, et nous sommes supposés générer des échantillons à partir de $D_{L,r'}$ où $r_i = r\sqrt{n}/(\alpha p)$. En plus, r est garanti à être tout au moins $\sqrt{2}p\eta_\varepsilon(L)$, qui engendre $p/r < \lambda_1(L^*)/2$.

L'étape itérative est obtenue par combinaison de deux parties (Se référer à la **Figure 4.04**). Dans la première partie, nous construisons un algorithme classique qui utilise les échantillons donnés et

l'oracle LWE pour résoudre le closest vector problem que nous allons dénoter par $CVP_{L^*,\alpha p/r}$: étant donné un point $x \in \mathbb{R}^n$ dans la distance $\alpha p/r$ de la *dual lattice* (notons que ces deux dernières termes sont anglais) L^*, générer le vecteur le plus proche de x dans L^*. De par notre hypothèse sur r, la distance entre tout couple de points dans L^* est plus grande que $2\alpha p/r$, ainsi le vecteur le plus proche est unique. Dans la seconde partie, nous utilisons cet algorithme pour générer des échantillons à partir de $D_{L,r'}$. Cette partie est quantique. L'idée est d'utiliser l'algorithme $CVP_{L^*,\alpha p/r}$ pour générer une certaine superposition quantique qui, après avoir appliqué la *transformation de Fourier quantique* (*quantum Fourier transform* pour sa version anglaise) et effectué une mesure quantique, génère des échantillons à partir de $D_{L,r\sqrt{n}/(\alpha p)}$.

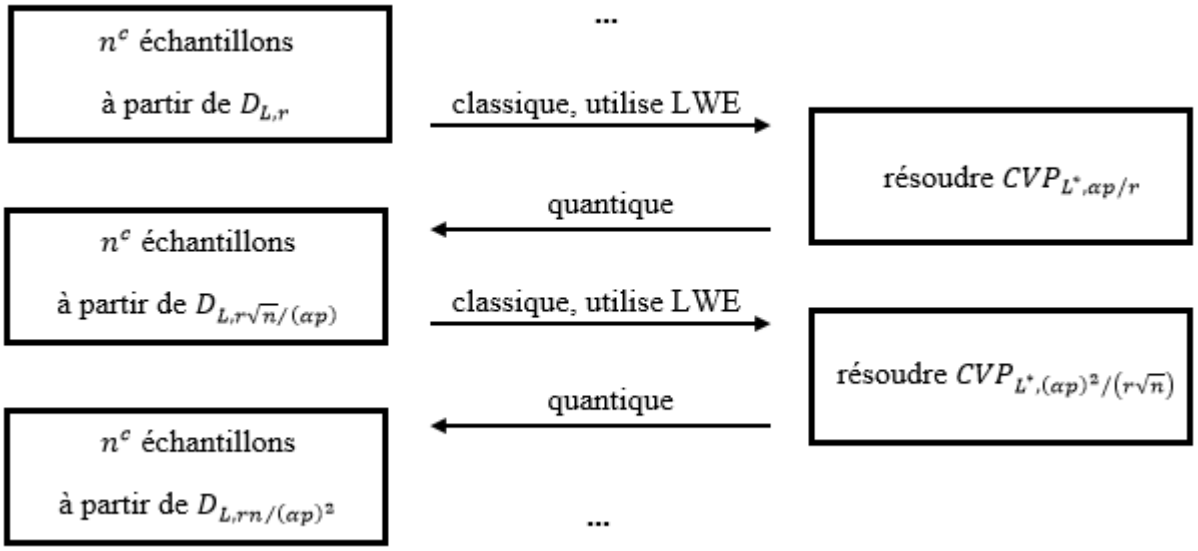


Figure 4.04 : *Deux itérations de l'algorithme.*

Dans ce qui va suivre, nous allons aborder en détails chacune des 2 parties de cette étape itérative.

Partie 1 : par soucis de manque de place, nous renvoyons nos lecteurs dans notre bibliographie [4.03] (paragraphe 1.1, sous-section Part 1) pour cette partie.

Partie 2 : Dans cette partie, nous allons décrire un algorithme quantique qui, en utilisant l'oracle $CVP_{L^*,\alpha p/r}$, génère un échantillon à partir de $D_{L,r\sqrt{n}/(\alpha p)}$. D'une manière équivalente, nous montrons comment produire un échantillon à partir de $D_{L,r}$ étant donné un oracle $CVP_{L^*,\sqrt{n}/r}$. La procédure est essentiellement comme ce qui suit : Dans un premier temps, en utilisant l'oracle CVP, créer un

état quantique correspondant à $f_{1/r}$. Ensuite, appliquer la transformation de Fourier quantique pour obtenir un état quantique correspondant à $D_{L,r}$. En mesurant cet état, nous obtenons les échantillons à partir de $D_{L,r}$.

Dans ce qui va suivre, nous allons décrire la procédure plus en détails. Notre premier but est de créer un état quantique correspondant à $f_{1/r}$. Cela peut être écrit comme :

$$\sum_{x\in\mathbb{R}^n} f_{1/r}|x\rangle \qquad \textbf{(}4.06\textbf{)}$$

Dans le but de préparer l'état dans notre formule **(**4.06**)**, nous créons d'abord la superposition uniforme sur L^* :

$$\sum_{x\in L^*} |x\rangle \qquad \textbf{(}4.07\textbf{)}$$

Dans un registre séparé, nous créons un état Gaussien, de longueur $1/r$:

$$\sum_{z\in\mathbb{R}^n} \exp(-\pi\|rz\|^2)\,|z\rangle \qquad \textbf{(}4.08\textbf{)}$$

L'état combiné du système peut être écrit comme :

$$\sum_{x\in L^*, z\in\mathbb{R}^n} \exp(-\pi\|rz\|^2)\,|x,z\rangle \qquad \textbf{(}4.09\textbf{)}$$

Et par la suite, rajouter le premier registre au second pour obtenir :

$$\sum_{x\in L^*, z\in\mathbb{R}^n} \exp(-\pi\|rz\|^2)\,|x,x+z\rangle \qquad \textbf{(}4.10\textbf{)}$$

Finalement, nous aimerions supprimer le premier registre pour obtenir :

$$\sum_{x\in L^*, z\in\mathbb{R}^n} \exp(-\pi\|rz\|^2)\,|x+z\rangle \approx \sum_{z\in\mathbb{R}^n} f_{1/r}(z)|z\rangle \qquad \textbf{(4.11)}$$

Toutefois, supprimer un registre est une opération non réversible. Pour qu'elle (l'opération) le soit, nous avons besoin de calculer x à partir du registre $x+z$ restant. Ceci justifie pourquoi nous avons besoin de l'oracle $CVP_{L^*,\sqrt{n}/r}$. Il peut être montré que tout le poids de $\exp(-\pi\|rz\|^2)$ est sur z avec $\|z\| \leq \sqrt{n}/r$. Ainsi $x+z$ est dans une distance $\sqrt{n}/r$ de la lattice donc l'oracle peut trouver le point lattice le plus proche, qui n'est autre que x. Cela nous permet de supprimer le premier registre d'une manière réversible.

Dans la dernière partie de la procédure, nous appliquons la transformation de Fourier quantique. Cela nous conduit à l'état quantique correspondant à $D_{L,r}$:

$$\sum_{y\in L} D_{L,r}\,(y)|y\rangle \qquad \textbf{(4.12)}$$

En mesurant cet état, nous obtenons l'échantillon à partir de la distribution $D_{L,r}$.

4.3 Le ring-LWE ou ring-Learning With Error

Pour petit rappel, le problème LWE consiste à distinguer les équations linéaires aléatoires, qui ont été perturbées par une petite quantité de bruits, des équations qui sont uniformes. Le problème a été montré être au même niveau de difficulté que les problèmes de lattices dans le cas le plus défavorable (worst-case lattice problem en anglais, pour rappel, toujours), a donc été la fondation de nombreux cryptosystèmes. Malheureusement, ces applications sont plutôt inefficaces en raison d'une surcharge quadratique (taille élevée de la clé, qui consiste généralement en une matrice carrée donnée) inhérente à l'utilisation de LWE. Une question qui s'est ouverte était de savoir si LWE et ses applications pouvaient être rendus plus efficaces en exploitant une structure algébrique supplémentaire.

Cette question a été résolue en introduisant une variante algébrique de LWE appelée ring-LWE, et en prouvant qu'elle bénéficie également de très fortes garanties en termes de difficultés. Plus

précisément, il a été montré que la distribution ring-LWE est pseudo-aléatoire, en assumant que les problèmes du cas le plus défavorable sur des *ideal lattices* (notons que ces deux termes sont anglais) sont difficiles pour les algorithmes quantiques à temps polynomial.

Comme avant-goût, ayons en tête que les ideal lattices peuvent être définies comme une généralisation des cyclic lattices. Ces dernières (les cyclic lattices) quant à elles se disent des lattices ayant une propriété telle que si $x \in L$ (L étant la lattice dont on parle, x un vecteur donné), alors x', un vecteur s'obtenant en tournant les coordonnées de x, appartient aussi à L.

4.3.1 Définition

Soit $f(x) = x^n + 1 \in \mathbb{Z}[x]$, où le paramètre de sécurité n est une puissance de 2, de telle sorte que $f(x)$ soit irréductible sur les nombres rationnels. Soit $R = \mathbb{Z}[x]/\langle f(x)\rangle$ l'*anneau* (the *ring*, en anglais) de polynômes entier modulo $f(x)$. Les éléments de R (c'est-à-dire, les résidus modulo $f(x)$) peuvent être présentés par un polynôme entier de degré plus petit que n. Soit $q = 1 \bmod 2n$ un module (*modulus* pour sa version anglaise) premier, public, suffisamment large (majoré d'une manière polynomial par n). Et soit $R_q = R/\langle q\rangle = \mathbb{Z}_q[x]/\langle f(x)\rangle$ l'anneau de polynômes entiers, modulo $f(x)$ et q à la fois. Les éléments q^n de R_q peuvent être représentés par de polynômes de degré inférieur à n dont les coefficients proviennent d'un ensemble de représentants canoniques de $\mathbb{Z}_q$, par exemple $\{0, \dots, q-1\}$.

Le problème ring-LWE dans R, dénoté R-LWE, peut-être informellement défini comme ce qui va suivre (une définition plus formelle sera donnée plus tard) : fixer une certaine distribution d'erreurs sur R, qui est concentré sur un nombre réduit d'éléments – ceux qui ont des coefficients entiers petits – et soit $s = s(x) \in R_q$ uniformément aléatoire. D'une manière analogique au LWE, le but est de distinguer arbitrairement plusieurs « équations sur l'anneau, aléatoirement bruitées » (random noisy ring equations, pour sa version anglaise) de ses pairs uniformes. D'une manière plus spécifique, les équations bruitées sont de la forme $(a, b) \approx a.s \in R_q \times R_q$, où chaque a est uniformément aléatoire, et le produit $a.s$ est perturbé par un terme tiré aléatoirement de la distribution d'erreur sur R.

Théorème 4.02 (informel) :

Supposons qu'il est difficile pour un algorithme quantique à temps polynomial d'approximer la

version recherche du SVP dans le cas le plus défavorable sur les ideal lattices dans R dans le respect d'un facteur $\mathrm{poly}(n)$ donné. Alors, tout nombre d'échantillons égal à $\mathrm{poly}(n)$ tiré d'une distribution R-LWE est pseudo-aléatoire pout un attaqueur (pouvant être quantique) à temps polynomial.

Le *Théorème 4.02* découle des deux affirmations qui vont suivre : la première (prouvé dans la section 4 de notre bibliographie [4.04]) est une réduction quantique du SVP approximative du cas le plus défavorable sur des ideal lattices vers la version de recherche du ring-LWE. La seconde (prouvée dans la section 5 de notre bibliographie [4.04], toujours) montre que la distribution R-LWE est en fait pseudo-aléatoire en supposant que le problème de recherche (du ring-LWE) est difficile.

4.3.2 Efficacité

Pour les applications cryptographiques, le problème R-LWE présente de nombreuses fonctionnalités intéressantes. Notez tout d'abord la force cryptographique du R-LWE par rapport au LWE standard : chaque produit bruité $b \approx as$ donne simultanément n valeurs pseudo-aléatoires sur $\mathbb{Z}_q$, plutôt qu'un seul scalaire, cependant le coût de sa génération reste assez faible : la multiplication polynomiale peut être effectuée en $O(n \log n)$ opérations scalaires, et en profondeur parallèle en $O(\log n)$, en utilisant la transformée de Fourier rapide (FFT) ou ses variantes, avec des implémentations optimisées en pratique. Enfin, dans la plupart des applications, chaque échantillon $(a, b) \in \mathbb{R}_q \times \mathbb{R}_q$ de la distribution R-LWE peut remplacer n échantillons $(a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ de la distribution LWE standard, réduisant ainsi la taille de la clé publique (et souvent de la clé secrète également) d'un facteur n. Ceci est particulièrement avantageux car la taille de la clé a été le principal obstacle des instanciations pratiques des cryptosystèmes à base de lattices ayant bénéficié d'une analyse de sécurité rigoureuse.

4.3.3 Sécurité

Etant donné l'efficacité du problème ring-LWE, une question se pose naturellement : dans quelle mesure l'hypothèse sous-jacente est-elle plausible ? Tous les outils algébriques et algorithmiques (y compris le calcul quantique) employés dans les réductions de difficulté, mentionnées dans le présent chapitre, peuvent également être utilisés contre (réduire leur difficulté) le SVP et d'autres problèmes sur des ideal lattices. Pourtant, malgré des efforts considérables, aucun progrès significatif n'a été réalisé pour s'attaquer à ces problèmes. Les algorithmes les plus connus pour les ideal lattices ne

fonctionnent pas mieux que ceux de leurs homologues génériques. En particulier, les algorithmes les plus rapides, asymptotiquement parlant, connus pour obtenir une approximation du SVP sur des ideal lattices, à des facteurs polynomiaux, nécessitent un temps $2^{\Omega(n)}$, tout comme dans le cas des lattices en général.

Une certaine confiance dans la difficulté des ideal lattices est également gagnée du fait qu'elles surviennent naturellement dans la théorie algébrique des nombres, une branche des mathématiques qui a été étudiée de manière rigoureuse, d'un point de vue informatique. En raison de leur application récente dans la conception de schémas cryptographiques, il est encore tôt pour dire quoi que ce soit sur leur sécurité avec une grande confiance. Une étude plus approfondie est certainement une direction de recherche importante.

4.4 Le MLWE ou Learning With Error over Module lattices

La plupart des schémas cryptographiques basés sur les lattices sont construits sur la difficulté supposée des Short Integer Solution (SIS) et du Leaning With Error (LWE). Leur efficacité peut être améliorée en déplaçant les hypothèses de difficulté vers les problèmes plus compacts que constituent les Ring-SIS et Ring-LWE. Cependant, ce changement d'hypothèses de difficulté s'accompagne d'un affaiblissement de la sécurité : SIS et LWE sont connus pour être au moins aussi difficiles que les problèmes standards (dans le cas le plus défavorable) sur les lattices euclidiennes (version classique des lattices), alors que Ring-SIS et Ring-LWE ont le même niveau de difficulté uniquement pour leurs restrictions à des classes spéciales d'ideal lattices, correspondant aux ideal de certains rings polynomiaux.

Le présent sous-titre de notre chapitre consiste en la définition des problèmes Module-SIS et, surtout, Module-LWE, qui relient respectivement SIS et Ring-SIS, et LWE avec Ring-LWE. Ces problèmes de cas moyen (average case, en anglais pour rappel) sont au moins aussi difficiles que les problèmes de lattices standards limités aux module lattices (ces derniers, les module lattices, à leur tour, servent de juste milieu entre les lattices arbitraires et les ideal lattices). Il est important de noter que les réductions allant du cas le plus défavorable au cas moyen pour les problèmes de module sont fortes, en ce sens qu'il existe des réductions inverses. Cette propriété n'est pas connue pour tenir dans le contexte de Ring-SIS / Ring-LWE : des problèmes d'ideal lattices pourraient se révéler faciles sans affecter la difficulté de Ring-SIS / Ring-LWE.

4.4.1 Preliminaires

4.4.1.1 Quelques théories algébriques des nombres

a. *Corps des nombres et corps cyclotomique*

Toute racine complexe d'un polynôme $g(X) \in \mathbb{Q}(X)$ est, par définition, un *nombre algébrique*. Le *polynôme minimal* d'un nombre algébrique ξ est l'unique *polynôme unitaire* (unitaire signifie que le coefficient du plus haut degré soit égal à un) f de degré minimal de telle sorte que ξ soit une de ses racines. Un *entier algébrique* est un nombre algébrique dont le polynôme minimal appartient à $\mathbb{Z}[X]$. Soit ξ un nombre algébrique, le corps de nombres $K = \mathbb{Q}(\xi)$ est une extension finie du corps des nombres rationnels $\mathbb{Q}$. Il (le corps de nombres K) est également un espace vectoriel de dimension n sur $\mathbb{Q}$, avec comme base $\{1, \xi, \dots, \xi^{n-1}\}$, où n étant le degré de f. Nous appelons également n, le degré de K. Soit R l'ensemble des entiers algébriques appartenant à K. Celui-ci (l'ensemble R) est un anneau, appelé anneau des entiers (ou ordre maximal) de K. Si ξ est un entier algébrique, alors $\mathbb{Z}[\xi] \subseteq R$.

Un *corps cyclotomique* est un corps $K = \mathbb{Q}(\xi)$ où ξ étant la racine de l'unité. Si ξ est la v-ième racine de l'unité alors il est le racine du v-ième polynôme cyclotomique Φ_v. Le degré $n = \phi(v)$ de Φ_v est le degré de K ($\phi(.)$ dénote la fonction indicatrice d'Euler, Euler totient function en anglais pour rappel). Dans le cas des corps cyclotomiques, nous avons $R = \mathbb{Z}[\xi]$.

b. *Complexe embeddings*

Ce sous-titre traitera les *complexe embeddings* (*plongements complexes* pour sa version française, mais nous nous tiendrons à la version anglaise dans le but d'éviter des ambigüités. Comme mentionné au début de la présente recherche également, les bibliographies en anglais sont largement plus répandues).

Les *embenddings canoniques* (on entendra par canonique, ce qui vient naturellement) sont les n anneaux d'homomorphismes : $\sigma_j : K \longrightarrow \mathbb{C}$ qui déterminent chacun des éléments de $\mathbb{Q}$. Dans notre cas particulier de corps cyclotomiques, les n embeddings sont complexes : ils sont définis par $\sigma_j : \xi \longrightarrow \xi^j$ pour tout $j \in \mathbb{Z}_v^\times$. Il est à noter que si j est inversible modulo v, alors il en est de même pour $v - j$, et que $\sigma_{v-j} = \overline{\sigma_j}$. Pour faciliter l'annotation, nous posons $\mathbb{J}$ l'ensemble constitué par $[v/2] \cap \mathbb{Z}_v^\times$. Et nous appelons par *canonical embedding vector* l'anneau d'homomorphisme

$\sigma_C: K \longrightarrow \mathbb{C}^n$ défini par : $\sigma_C(y) = \left(\sigma_j(y)\right)_{j \in \mathbb{Z}_v^\times}$ où l'addition et la multiplication dans $\mathbb{C}^n$ sont *component-wise*. En effet, pour tout $x, y \in K$, $\sigma_C(x \cdot y)$ est la multiplication component-wise de $\sigma_C(x)$ et $\sigma_C(y)$. De par l'algèbre linéaire élémentaire, nous pouvons observer que chacun des éléments de K est entièrement décrit par son canonical embedding vector.

La *trace* (le nom est la même en version française et anglaise) $\mathrm{Tr}: K \longrightarrow \mathbb{Q}$ et la *norme* (*norm* en anglais) sont respectivement définies comme suit : $\mathrm{Tr}(x) = \sum_{j \in \mathbb{Z}_v^\times} \sigma_j(x)$ et $\mathrm{N}(x) = \prod_{j \in \mathbb{Z}_v^\times} \sigma_j(x)$. Pour tout $x, y \in K$, nous avons $\mathrm{Tr}(x \cdot y) = \sum_{j \in \mathbb{Z}_v^\times} \sigma_j(x) \cdot \sigma_j(y) = \langle \sigma_C(x), \sigma_C(y) \rangle$ où $\langle \cdot, \cdot \rangle$ est le produit canonique Hermitien sur $\mathbb{C}^n$.

c. *Espace H*

Nous utiliserons le sous-espace de $\mathbb{C}$ ci-dessous :

$$H = \left\{ \left(x_j\right)_{j \in \mathbb{Z}_v^\times} \in \mathbb{C}^n : \ \forall j \in \mathbb{J}, x_{v-j} = \overline{x_j} \right\} \quad \textbf{(4.13)}$$

Soit $h_j = \frac{1}{\sqrt{2}}\left(e_j + e_{v-j}\right)$ et $h_{v-j} = \frac{i}{\sqrt{2}}\left(e_j - e_{v-j}\right)$ pour $j \in \mathbb{J}$. Les h_j forment une base de H en tant qu'espace vectoriel. Un élément $x \in K$ peut être représenté selon la base $(h_j)_j$: pour $x \in K$, nous définissons $\sigma_H(x)$ par $\sigma_H(x) = (x_j)_j \in \mathbb{R}^n$ de telle sorte que $\sigma_C(x) = \sum_j x_j \cdot h_j$. Comme $\sigma_C(x) = (\sigma_j(x))_j$, nous avons, pour $j \in \mathbb{J}$:

$$\begin{bmatrix} x_j \\ x_{v-j} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix}\begin{bmatrix} \sigma_j(x) \\ \sigma_{v-j}(x) \end{bmatrix} \text{ et } \begin{bmatrix} \sigma_j(x) \\ \sigma_{v-j}(x) \end{bmatrix} = \begin{bmatrix} 1 & i \\ 1 & -i \end{bmatrix}\begin{bmatrix} x_j \\ x_{v-j} \end{bmatrix} \quad \textbf{(4.14)}$$

L'addition dans H est component-wise. Soit $\sigma_H(x) = (x_j)_j$ et $\sigma_H(y) = (y_j)_j$, nous définissons la multiplication par $\sigma_H(x \cdot y) = (z_j)_j$ où, pour $j \in \mathbb{J}$:

$$\begin{bmatrix} z_j \\ z_{v-j} \end{bmatrix} = \begin{bmatrix} x_j & -x_{v-j} \\ x_{v-j} & x_j \end{bmatrix}\begin{bmatrix} y_j \\ y_{v-j} \end{bmatrix} \text{ et } \begin{bmatrix} z_j \\ z_{v-j} \end{bmatrix} = \begin{bmatrix} y_j & -y_{v-j} \\ y_{v-j} & y_j \end{bmatrix}\begin{bmatrix} x_j \\ x_{v-j} \end{bmatrix} \quad \textbf{(4.15)}$$

Pour simplifier la présentation, nous identifions les éléments de K par leurs embedding σ_H.

d. *Ideals*

Un (Integral) *Ideal* (*Idéal* pour sa version française) I de R est un sous-groupe additif de R fermé (ou stable) par multiplication par un élément de R. Le plus petit ideal de R contenant un ensemble S est dénoté (S). Le quotient R/I est l'ensemble des classes d'équivalence $g + I$ de R modulo I. Pour tout ideal non nul, la norme $N(I)$ de l'ideal est le nombre d'éléments de l'anneau quotient R/I. Nous avons $N\big((x)\big) = N(x)$, pour tout $x \in K$.

Soient I et J des ideals de R. Nous définissons le produit de 2 ideals par $IJ = \{\sum_i \alpha_i\beta_i : \alpha_i \in I, \beta_i \in J\}$ et leur somme $I + J = \{\alpha + \beta : \alpha \in I, \beta \in J\}$. Un ideal $I \subset R$ est *premier* si pour tout $ab \in I$ alors $a \in I$ ou $b \in I$. Tout ideal de R peut être représenté comme le produit de deux ideals premiers, et pour un ideal premier I, l'anneau quotient R/I est un corps fini d'ordre $N(I)$. Un *fractional ideal* $I \subseteq K$ est un ensemble de telle sorte que $dI \subseteq R$ pour un $d \in R$ non nul. L'*inverse* d'un fractional ideal $I^{-1} = \{\alpha \in K : \alpha I \subseteq R\}$ et est lui-même un fractional ideal. Nous avons $II^{-1} = R$. Le *dual* d'un ideal est défini par $I^\vee = \{x \in K : \mathrm{Tr}(xI) \subseteq \mathbb{Z}\}$. Nous avons $I^\vee = I^{-1} \cdot R^\vee$.

Dans notre paramétrage de corps cyclotomiques, si q est un entier premier, l'ideal premier de factorisation de $(q) \subseteq R$ peut être évalué d'une manière efficace. En particulier, si $q = 1 \bmod v$, alors $(q) = \prod_{j \in \mathbb{Z}_v^\times} q_j$ où chaque q_j est un ideal premier avec comme norme $N\big(q_j\big) = q$. Le corps K a n automorphismes $\tau_j : K \longrightarrow K$ défini par $\tau_j(\xi) = \xi^j$ (pour $j \in \mathbb{Z}_v^\times$)

e. *Isomorphisme d'anneaux quotients*

Soient R_q et $R_q^\vee$ dénotent respectivement R/qR et $R^\vee/qR^\vee$. Soit $t \in I$ de telle sorte que $(t) + qI = I$ (un tel t existe et peut être trouvé efficacement étant donné I la factorisation d'ideal premier de (q)). La fonction $\theta_I : K \longrightarrow K$ définie comme $\theta_I(x) = t \cdot x$ induit un isomorphisme de R_q vers I/qI. En addition, cet isomorphisme peut être efficacement inversé en utilisant $\theta_I^{-1} : I/qI \longrightarrow R_q$ défini par $\theta_I^{-1}(y) = t^{-1} \cdot y' \bmod qR$ où $y' = y \bmod qI^\vee$ et $y' \in (t)$. La fonction θ_I induit également un isomorphisme de $I^\vee/qI^\vee$ vers $R_q^\vee$ qui bénéficie d'une inverse en utilisant $\theta_I^{-1} : R_q^\vee \longrightarrow I^\vee/qI^\vee$ avec $\theta_I^{-1}(y) = t^{-1}.y' \bmod qR$ où $y' = y \bmod qI^\vee$ et $y' \in (t)$.

f. *Modules*

Un sous-ensemble $M \subseteq K^d$ est un R-module s'il est stable par une addition et multiplication par un

élément de R. C'est un module généré d'une manière finie s'il existe une famille finie $(b_k)_k$ de vecteurs dans K^d de telle sorte que $M = \sum_k R \cdot b_k$. En général, si l'anneau R est arbitraire, un R-module pourrait ne pas avoir de base. Mais dans le cas présent, K et un corps de nombres, donc R est un *anneau de Dedekind* (*Dedekind domain* en anglais), et nous avons l'existence de pseudo-bases : pour tout module M, il existe I_k, un ideal de R non nul, et $(b_k)_k$ vecteurs linéairement indépendants de K^d de telle sorte que $M = \sum_{k=1}^{d} I_k \cdot b_k$. Nous disons que $[(I_k)_k, (b_k)_k]$ est un *pseudo-base* de M. La terminologie pseudo-bases est utilisée car les coefficients ideals I_k peuvent être non principaux. Dans le sens que deux pseudo-bases peuvent générer le même module et donc, elles ont la même cardinalité. Cette dernière (la cardinalité) est appelée rang du module.

Nous définissons le dual d'un module par $M^\vee = \{x \in K^d, \forall y \in M : \mathrm{Tr}(\langle x, \bar{y} \rangle) \in \mathbb{Z}\}$, où $\langle \cdot, \cdot \rangle$ est le produit Hermitien sur K^d. Nous avons la propriété qui va suivre :

Lemme 4.01 :

Si $M = \sum_{k=1}^{d} I_k \cdot b_k$, alors $M^\vee = \sum_{k=1}^{d} I_k^\vee \cdot b_k^\vee$, où les $b_l^\vee$ sont définis par $\langle b_k, \overline{b_l^\vee} \rangle = 1$ si $k = l$ et $\langle b_k, \overline{b_l^\vee} \rangle = 0$ sinon.

Démonstration :

♣

Nous montrons tout d'abord que $\sum_{k=1}^{d} I_k^\vee \cdot b_k^\vee \subseteq M^\vee$. Soit $x \in \sum_{k=1}^{d} I_k^\vee \cdot b_k^\vee$. Alors, pour tout i, il existe $x_k \in I_k^\vee$ de telle sorte que $x = \sum_{k=1}^{d} x_k \cdot b_k^\vee$. Soit $y = \sum_{k=1}^{d} y_k \cdot b_k \in M$. Alors, par linéarité, nous avons $\mathrm{Tr}(\langle x, \bar{y} \rangle) = \sum_{k=1}^{d} \mathrm{Tr}(x_k, y_k)$. Pour tout i nous avons $x_k \in I_k^\vee$ et $y_k \in I_k$, et ainsi $\mathrm{Tr}(x_k, y_k) \in \mathbb{Z}$. Ainsi, nous avons $\mathrm{Tr}(\langle x, \bar{y} \rangle) \in \mathbb{Z}$ et $x \in M^\vee$.

Nous montrons tout d'abord que $M^\vee \subseteq \sum_{k=1}^{d} I_k^\vee \cdot b_k^\vee$. Soit $x \in M^\vee \subseteq K^d$. Nous pouvons écrire $x = \sum_{k=1}^{d} x_k \cdot b_k^\vee$ pour un x_k donné dans K. Il suffit de montrer que $x_k \in I_k^\vee$. Soit $y_k \in I_k^\vee$ arbitraire. Par linéarité, nous avons $\mathrm{Tr}\left(\langle x, \overline{y_k b_k} \rangle\right) = \mathrm{Tr}(y_k b_k) \in \mathbb{Z}$. Cela implique que $x_k \in I_k^\vee$.

♦

Nous généralisons l'isomorphisme θ_I défini plus haut pour les modules. Soit $\sum_{k=1}^{d} I_k \cdot b_k$, $f = I_1/qI_1 \times \ldots \times I_d/qI_d \longrightarrow M/qM$ de telle sorte que $f(x_1, \ldots, x_n) = \sum_{k=1}^{d} x_k \cdot b_k$ et $q = M/qM \longrightarrow$

$I_1/qI_1 \times \ldots \times I_d/qI_d$ de telle sorte que $g\left(\sum_{k=1}^{d} x_k \cdot b_k\right) = (x_1, \ldots, x_n)$. Les fonctions f et g sont des isomorphismes d'anneaux et $g = f^{-1}$. Soient $\theta_{I_1}, \ldots, \theta_{I_d}$ comme décrit plus haut. Nous définissons les fonctions Θ et Θ^{-1} comme suit : $\Theta = f \circ \left(\theta_{I_1}, \ldots, \theta_{I_d}\right)$ et $\Theta^{-1} = \left(\theta_{I_1}^{-1}, \ldots, \theta_{I_d}^{-1}\right) \circ g$. La fonction Θ induit un isomorphisme de R_q^d vers M/qM avec comme inverse Θ^{-1}.

4.4.1.2 Quelques notions additionnelles sur les lattices

a. *GIVP, une généralisation du SIVP*

Considérons la généralisation du SIVP qui va suivre. Soit ϕ une fonction arbitraire à valeur réelle d'une lattice (exemple, en prenant $\phi = \lambda_n$ permet de recouvrir SIVP_γ). Soit $\gamma \geq 1$ une fonction dépendant de la dimension n (dimension de la lattice), le *Generalized Independent Vector Problem* GIVP_γ^ϕ est comme suit : étant donnée une base B d'une lattice, trouver $n = dim\left(\mathcal{L}(B)\right)$ vecteurs linéairement indépendants $s_1, \ldots, s_n \in \mathcal{L}(B)$ de telle sorte que $\max_i \|s_i\| \leq \gamma . \phi\left(\mathcal{L}(B)\right)$.

Pour $\phi = \lambda_n$, ce problème est NP-difficile pour tout facteur d'approximation $\gamma \leq O(1)$. Le plus performant des algorithmes connus (même quantique) pour une solution exacte ou approximative selon un facteur polynomial γ ont tous une complexité exponentielle. Ce fait supporte la conjoncture qui suit : Il n'y a pas d'algorithme (quantique) à temps polynomial qui peut approcher les problèmes à base de lattice par respect à un facteur d'approximation polynomial.

b. *Ideal et module lattice*

Etant donné σ_H un embedding de K vers $\mathbb{R}^n$ et I un ideal de R, l'ensemble $\sigma_H(I)$ est une lattice. Nous l'appelons ideal lattice par respect à K. Nous représenterons par Id-GIVP la restriction du GIVP aux ideal lattices.

Nous définissons les module lattices d'une manière équivalente. L'application $(\sigma_H, \ldots, \sigma_H)$ est un embedding de K_d vers $\mathbb{R}^n$, avec $N = nd$ et $M \subseteq K^d$ un module de R. Par abus de notification, nous l'appelons également σ_H. L'ensemble $\sigma_H(M)$ est une module lattice. D'une manière similaire aux ideal lattices, nous notons Mod-GIVP la restriction du GIVP aux module lattices. Notons que si M est un module de rang d et si K a un degré n, alors la module lattice correspondante a une

dimension $N = nd$. Pour tout $x \in K^d$, nous définissons :

$$\begin{aligned} \|x\| &= \left(\sum_{k\in[d]} \sum_{j\in\mathbb{Z}_v^\times} \left|\sigma_j(x_k)\right|^2\right)^{1/2} \\ \|x\|_\infty &= \max_{j,k} \left|\sigma_j(x_k)\right| \\ \|x\|_{2,\infty} &= \max_j \left(\sum_k \left|\sigma_j(x_k)\right|^2\right)^{1/2} \\ \|x\|_{\infty,2} &= \max_k \left(\sum_j \left|\sigma_j(x_k)\right|^2\right)^{1/2} \end{aligned} \quad \textbf{(4.16)}$$

Quand un module est donné en tant qu'entrée d'un problème, nous considérons qu'une base du module lattice correspondant est donnée. En effet, cela est équivalent au fait de donner une base de la module lattice, et une pseudo-base du module parce que, de la première représentation, on peut évaluer en un temps polynomial la seconde.

4.4.2 Mesures Gaussiennes

Pour un facteur $c \in \mathbb{R}^n$ et un réel $s > 0$, la fonction Gaussienne est définie par $\rho_{s,c}(x) = e^{-\pi\left\|\frac{x-c}{s}\right\|^2}$, pout tout $x \in \mathbb{R}^n$. Cette fonction est étendue sur tout ensemble dénombrable $A \subset \mathbb{R}^n$, selon la manière usuelle : $\rho_{s,c}(A) = \sum_{x\in A} \rho_{s,c}(x)$. En normalisant la fonction Gaussienne, nous obtenons la Distribution de probabilité Gaussienne continue $D_s(x) = \rho_{s,c}(x)/s^n$. Pour $r = (r_1, \dots, r_n)^T \in (\mathbb{R}^+)^n$, un échantillon à partir de D_r sur $\mathbb{R}^n$ est donné par $(D_{r_i})_i$.

L'implémentation qui nous intéresse dans le présent travail est la version elliptique des distributions Gaussiennes, dans la base $\{h_j\}_{j\in\mathbb{Z}_v^\times}$. Pour $(r_j)_{j\in\mathbb{Z}_v^\times} \in \mathbb{R}^n$ de telle sorte que $r_j = r_{v-j}$ pour tout $j \in \mathbb{J}$, un échantillon $x \in D_r$ est donné par $\sigma_C(x) = \sum_j x_j \cdot h_j$, où tous les x_j sont indépendamment choisis à partir de la distribution Gaussienne D_{r_j} sur $\mathbb{R}$.

Nous définissons par $\Psi_{\alpha,\alpha'}$, pour $0 \le \alpha < \alpha'$, l'ensemble des distributions Gaussiennes D_r avec $\alpha < r_i \le \alpha'$, pour tout i. Nous utilisons la notation $\Psi_{\le\alpha'}$ quand $\alpha = 0$. Nous rappelons également la distribution Υ_α se caractérisant comme suit. La distribution $\Gamma(2,1)$ avec 2 comme *shape parameter* (*paramètre de forme* en français), et 1 le *scale parameter* (*paramètre d'échelle* en

français), a comme densité $t\exp(-t)$ pour $t \geq 0$ et zéro pour $t < 0$. Pour $\alpha > 0$, une distribution échantillonnée à partir de Υ_α est une distribution Gaussienne elliptique D_r avec comme paramètre $r_j = r_{v-j} = \alpha\sqrt{1 + \sqrt{n}x_j}$, où les x_j pour $j \in \mathbb{J}$ sont indépendamment choisis à partir de $\Gamma(2,1)$. Nous utiliserons les résultats qui suivent sur $\Gamma(2,1)$.

Lemme 4.02 :

Soit P la distribution $\Gamma(2,1)^n$, et Q la distribution $(\Gamma(2,1) - z_1) \times \ldots \times (\Gamma(2,1) - z_n)$ pour $0 \leq z_1, \ldots, z_n \leq 1/\sqrt{n}$. Alors pour tout ensemble mesurable $A \subseteq \mathbb{R}^n$, nous avons $\int_A Q \geq \frac{1}{\text{poly}(n)} \cdot \left(\int_A P\right)^2$.

Pour tout $c \in \mathbb{R}^n$, $s > 0$ et une lattice Λ, la distribution de probabilité Gaussienne discrète à support Λ, centré en c et ayant un écart-type s, est défini par :

$$\forall x \in \Lambda, D_{\Lambda,s,c} = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\Lambda)} \qquad \textbf{(}4.17\textbf{)}$$

Le théorème qui suit assure que pour s suffisamment grand, il est possible d'échantillonner efficacement suivant une distribution Gaussienne discrète.

Théorème 4.03 :

Il y a un algorithme probabiliste à temps polynomial qui, étant donné une base B d'une lattice de dimension n $\Lambda = \mathcal{L}(B)$, un écart-type $s = \left\|\tilde{B}\right\| \cdot \sqrt{\log n}$ ($\widetilde{b_i}$ est la composante de b_i orthogonale au $\text{span}(b_1, \ldots, b_{i-1})$ pour $i = 2, \ldots, n$, et $\left\|\tilde{B}\right\| = \max_{i\in[n]}\left\|\widetilde{b_i}\right\|$), et un centre $c \in \mathbb{R}^n$, renvoie une échantillon dont la distribution est $D_{\Lambda,s,c}$. [4.09]

Le *paramètre de lissage* (*smoothing parameter* en anglais, pour rappel) d'une lattice se caractérise comme suit. Pour une lattice Λ de dimension n, et un nombre réel $\varepsilon > 0$, le paramètre de lissage $\eta_\varepsilon(\Lambda)$ est le plus petit s de telle sorte que $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \varepsilon$ (Λ^* représente la dual lattice de Λ). Ce paramètre donne le seuil au-delà duquel plusieurs propriétés du Gaussien continu, est également valable pour le Gaussien discret. Ci-dessous quelques-unes, qui interviendront dans les réductions qui apparaitront plus tard dans le présent chapitre.

Lemme 4.03 :

Soit Λ une lattice de dimension n, et $\varepsilon > 0$. Alors $\eta_\varepsilon(\Lambda) \leq \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}} \cdot \lambda_n(\Lambda)$.

Ce lemme implique une réduction du SIVP_γ au $\mathrm{GIVP}_{\gamma'}^{\eta_\varepsilon}$, avec $\gamma' = \gamma\sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}}$, par exemple.

Lemme 4.04 :

Soit Λ une lattice de dimension n, et $\varepsilon > 0$. Alors $\eta_\varepsilon(\Lambda) \leq \sqrt{\frac{\ln(2n(1+1/\varepsilon))}{\pi}}/\lambda_1^\infty(\Lambda^*)$, où λ_1^∞ fait référence au minimum de la lattice par respect à la norme infinie.

Lemme 4.05 :

Soit $\Lambda' \subseteq \Lambda$ une lattice de dimension n. Alors, pour tout $\varepsilon \in (0,1)$, tout $s \geq \eta_\varepsilon(\Lambda')$, et tout $c \in \mathbb{R}^n$, la distribution $\left(D_{\Lambda,a,c} \bmod \Lambda'\right)$ est dans une distance statistique tout au plus égal à 2ε de la distribution uniforme sur Λ/Λ'.

Lemme 4.06 :

Λ une lattice de dimension n, $s \geq 2\eta_\varepsilon(\Lambda)$, pour $\varepsilon \leq 1/100$, $c \in \mathbb{R}^n$. Alors, tout hyperplan $\mathcal{H}$ de dimension $(n-1)$, la probabilité que $x \notin \mathcal{H}$, où x est choisi à partir $D_{\Lambda,a,c}$ est $\geq 1/100$.

4.4.3 Réduction du Mod-GIVP au M-LWE (Learning With Errors over Modules)

4.4.3.1 Variantes du LWE

a. *LWE*

Soit $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ dénotant le segment $[0,1)$ avec l'opération addition modulo 1. Pour une fonction de densité de probabilité χ sur $\mathbb{T}$ et un vecteur $s \in \mathbb{Z}_q^n$, nous posons $A_{s,\chi}$ la distribution sur $\mathbb{Z}_q^n \times \mathbb{T}$ obtenue en choisissant un vecteur $a \in \mathbb{Z}_q^n$ uniformément et d'une manière aléatoire, puis $e \in \mathbb{T}$ par respect à χ, et retournant $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Définition 4.01 :

La version recherche du problème Learning With Error $\mathrm{SLWE}_{q,\chi}$ est comme suit : soit $s \in \mathbb{Z}_q^n$ un

secret ; étant donnés plusieurs échantillons à partir de $A_{s,\chi}$, le but est de trouver s.

La version décision du problème Learning With Error $\mathrm{LWE}_{q,\chi}$ est comme suit : soit $s \in \mathbb{Z}_q^n$ choisi d'une manière uniforme et aléatoire ; le but est de faire la distinction entre plusieurs échantillons indépendants à partir de $A_{s,\chi}$ et le même nombre d'échantillons indépendants à partir de $U(\mathbb{Z}_q^n \times \mathbb{T})$. Où U fait référence à une distribution uniforme, sur l'ensemble $\mathbb{Z}_q^n \times \mathbb{T}$ dans le cas présent donc.

Il est également possible d'interpréter le LWE (**Figure 4.05**) en termes d'algèbre linéaire : Supposons que le nombre d'échantillons requis $\left(a_i, \frac{1}{q}\langle a_i, s\rangle + e_i\right)$ à partir de $A_{s,\chi}$ est m, alors nous considérons la matrice $A \in \mathbb{Z}_q^{m \times n}$ dont les lignes sont les a_i. Et on considèrera le vecteur $e = (e_1, \ldots, e_m)^T$. Alors le SLWE est comme suit : [4.05]

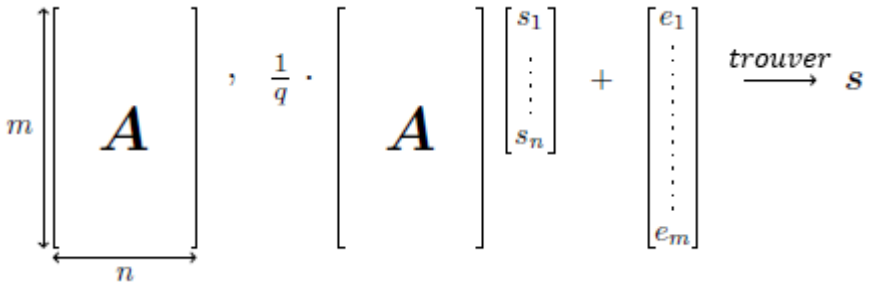


Figure 4.05 : *Le LWE en termes d'algèbre linéaire.*

Théorème 4.04 :

Soit $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ de telle sorte que $\alpha q \geq 2\sqrt{n}$. Il existe une réduction quantique de la résolution du $\mathrm{GIVP}_{\sqrt{8}n/\alpha}^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\mathrm{SLWE}_{q,D_\alpha}$ en un temps polynomial avec une probabilité non négligeable.

En assumant que q est premier, $q \leq \mathrm{poly}(n)$, et que χ soit une fonction de densité de probabilité sur $\mathbb{T}$. Alors il existe une réduction en un temps polynomial du $\mathrm{SLWE}_{q,\chi}$ au $\mathrm{LWE}_{q,\chi}$.

Le théorème qui va suivre permet d'outrepasser la condition que q doit être premier.

Théorème 4.05 :

Soit $\alpha > 0$, $\varepsilon \in (0, 1/2)$, $m \geq n \geq 1$, $p \geq 25$ et $q \in [p, 2p)$. Il existe une réduction à temps

polynomial du $\mathrm{LWE}_{q,\alpha}$ au $\mathrm{LWE}_{p,\Psi_{\leq\beta}}$ où $\beta = C\alpha\sqrt{n}\sqrt{\log(n/\varepsilon)\log(nm/\varepsilon)}$, pour une constante positive C, et qui perd tout au plus ε en avantage.

b. *R-LWE*

Soit ψ une distribution sur $\mathbb{T}_{R^\vee} = K_{\mathbb{R}}/R^\vee$ et $s \in R_q^\vee$. Nous dénotons par $A_{s,\psi}^{(R)}$ la distribution sur $R_q \times \mathbb{T}_{R^\vee}$ obtenu en choisissant $a \in R_q$ d'une manière uniformément aléatoire et $e \in \mathbb{T}_{R^\vee}$ par respect à ψ, et retournant $(a, (a.s)/q + e)$.

Définition 4.02 :

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Ring Learning With Error $\text{R-SLWE}_{q,\Psi}$ est comme suit : soit $s \in R_q^\vee$ secret, et $\psi \in \Psi$; étant donnés plusieurs échantillons à partir de $A_{s,\psi}^{(R)}$, le but est de trouver s.
Soit Υ une distribution sur une famille de distribution de bruits sur $\mathbb{T}_{R^\vee}$. La version décision du problème Ring Learning With Error $\text{R-SLWE}_{q,\Upsilon}$ est comme suit : supposons que $s \in R_q^\vee$ soit uniformément aléatoire, et ψ échantillonné à partir de Υ ; le but est de distinguer entre plusieurs échantillons indépendants partir de $A_{s,\psi}^{(R)}$ et le même nombre d'échantillons indépendants à partir de $U(R_q, \mathbb{T}_{R^\vee})$.

Théorème 4.06 :

Soit $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > \omega(\sqrt{\log n})$. Il existe une réduction quantique de la résolution du $\text{Id-GIVP}_\gamma^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\text{R-SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec une probabilité non négligeable avec $\gamma = \sqrt{n}.\omega(\sqrt{\log n})/\alpha$.

En supposant que q est premier, $q \leq \mathrm{poly}(n)$, et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial de $\text{R-SLWE}_{q,\Psi_{\leq\alpha}}$ au $\text{R-LWE}_{q,\Upsilon_\alpha}$.

c. *LWE over module*

Le M-LWE généralise le LWE et le R-LWE. La variable n et d dénotent respectivement la dimension de R et le rang du module $M \subseteq R^d$. Nous posons $N = nd$ la dimension de la module

lattice.

Soit ψ une probabilité de distribution sur $\mathbb{T}_{R^\vee}$ et $s \in \left(R_q^\vee\right)^d$ un vecteur. Nous définissons par $A_{q,s,\psi}^{(M)}$ la distribution sur $\left(R_q\right)^d \times \mathbb{T}_{R^\vee}$ obtenu en choisissant un vecteur $a \in \left(R_q\right)^d$ uniformément aléatoire, et $e \in \mathbb{T}_{R^\vee}$ par respect à ψ, et retournant $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Définition 4.03 :

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Module Learning With Error M-SLWE$_{q,\Psi}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ un vecteur secret et $\psi \in \Psi$; étant donnés arbitrairement plusieurs échantillons à partir de $A_{q,s,\psi}^{(M)}$, le but est de trouver s.
Pour un entier $q \geq 2$ et une distribution Υ sur une famille de distribution sur $K_{\mathbb{R}}$, la version décision du problème Module Learning With Error M-SLWE$_{q,\Upsilon}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ uniformément aléatoire et ψ échantillonné sur Υ ; le but est de distinguer entre arbitrairement plusieurs échantillons aléatoires à partir de $A_{q,s,\psi}^{(M)}$ et le même nombre d'échantillons à partir de $U\left(R_q^d \times \mathbb{T}_{R^\vee}\right)$.

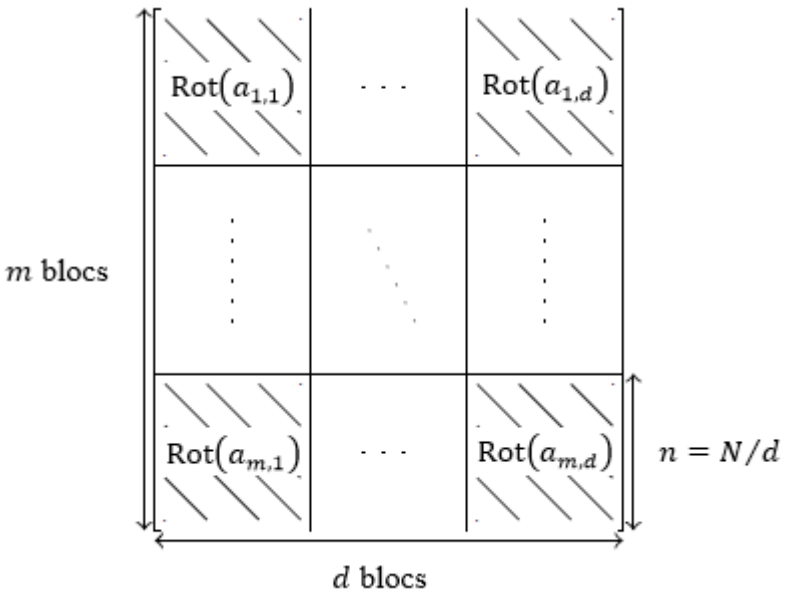


Figure 4.06 : *The M-LWE problem can be interpreted in terms of linear algebra.*

Comme c'était le cas du LWE et du R-LWE, le problème M-LWE peut être interprété en termes

d'algèbre linéaire. Il (le M-LWE) consiste à prendre la matrice A du LWE pour arriver à la forme illustrée dans la **Figure 4.06**.

La difficulté du M-LWE est soutenue par les deux théorèmes qui vont suivre.

Théorème 4.07 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > 2\sqrt{d}.\omega(\sqrt{\log n})$. Il y a une réduction quantique de la résolution du Mod-GIVP$_{\gamma}^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du M-SLWE$_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec un avantage non négligeable avec $\gamma = \sqrt{8Nd}.\omega(\sqrt{\log n})/\alpha$.

En assumant que q est premier, $q \leq \text{poly}(N)$ et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial depuis M-SLWE$_{q,\Psi_{\leq\alpha}}$ vers M-LWE$_{q,\Upsilon_\alpha}$.

Quand $n = N$ et $d = 1$, le *Théorème 4.06* est équivalent au *Théorème 4.05*. Quand $n = 1$ et $d = N$, il (le *Théorème 4.06*, toujours) est équivalent au *Théorème 4.03*.

Théorème 4.08 :

Soit $p, q \in \left[2, 2^{N^{O(1)}}\right]$ et $\alpha, \beta \in (0,1)$ de telle sorte que $\beta \geq \alpha.\max\left(1, \frac{q}{p}\right).n^{1/4}N^{1/2}.\omega(\log^2 N)$ et $\alpha q \geq \omega\sqrt{\log(N)/n}$. Il existe une réduction à temps polynomial de M-LWE$_{q,\Upsilon_\alpha}$ vers M-LWE$_{q,\Upsilon_\beta}$.

4.4.3.2 Réduction du Mod-GIVP au M-SLWE

La réduction du Mod-GIVP au M-SLWE utilise le problème intermédiaire qui va suivre, où ϕ dénote une fonction à valeur réelle sur une lattice et γ dépend de la dimension, appelé *Module Discrete Gaussian Sampling Problem* M-DGS$_{\gamma}^{\phi}$: étant donnée une module lattice M de dimension N, et un nombre $r > \gamma \cdot \phi$, le but est de retourner un échantillon à partir de $D_{M,r}$. La réduction se fait en 2 (deux) étapes :

$$\text{Mod-GIVP}_{\sqrt{8Nd}\cdot\omega(\sqrt{\log n})/\alpha}^{\eta_\varepsilon} \xrightarrow{Lemme\ 4.07} \text{M-DGS}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha}^{\eta_\varepsilon} \xrightarrow{Lemme\ 4.08} \text{M-SLWE}_{q,\Psi_{\leq\alpha}}$$

Lemme 4.07 :

Pour tout $\varepsilon = \varepsilon(N) \leq 1/10$ et tout γ et ϕ de telle sorte que $\gamma \cdot \phi(M) \leq \sqrt{2}\eta_\varepsilon(M)$, il existe une réduction à temps polynomial du $\text{Mod-GIVP}^{\phi}_{2\sqrt{N}\cdot\gamma}$ au $\text{M-DGS}^{\phi}_{\gamma}$. [4.05]

Lemme 4.08 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et q un entier de telle sorte que $\alpha q \geq 2\sqrt{d} \cdot \omega(\sqrt{\log n})$. En assumant que nous avons accès à un oracle qui résout $\text{M-SLWE}_{q,\Psi_{\leq\alpha}}$, étant donné un nombre d'échantillons polynomial. Alors il existe un algorithme quantique à temps polynomial pour $\text{M-DGS}^{\eta_\varepsilon}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha}$.

Démonstration :

♣

Considérons une module lattice M et un nombre $r \geq \sqrt{2d} \cdot \omega\left(\sqrt{\log n}\right).\eta_\varepsilon(M)/\alpha$. L'idée est de produire des échantillons pour $D_{M,r'}$ avec r' suffisamment large, et puis d'utiliser le *Lemme 4.09* plusieurs fois progressivement pour diminuer la valeur de r'. Soit $r_i = r.\left(\alpha q/\sqrt{d} \cdot \omega\left(\sqrt{\log n}\right)\right)^i$. La première itération commence avec $r_{3N} > 2^{3N} > 2^{2N}\lambda_N(M)$ (en utilisant un algorithme de réduction en LLL). Ainsi, on obtient $\text{poly}(N)$ échantillons à partir de $D_{M,r_{3N}}$ en utilisant l'algorithme du *Théorème 4.03*, et se termine avec $\text{poly}(N)$ échantillons à partir de $D_{M,r_{3N-1}}$(la réduction se répète $\text{poly}(N)$ fois la même itération avec le même nombre d'échantillons en entrée pour obtenir suffisamment d'échantillons à la sortie). L'itération se fait jusqu'à ce que l'on ait $\text{poly}(N)$ échantillons de D_{M,r_1} avec $r_1 = r \cdot \alpha q/\left(\sqrt{d} \cdot \omega\left(\sqrt{\log n}\right)\right) > \sqrt{2}q \cdot \eta_\varepsilon(M)$, pour s'itérer une dernière fois pour avoir des échantillons à partir de D_{M,r_0} avec $r_1 = r > \sqrt{d} \cdot \omega\left(\sqrt{\log n}\right) \cdot \eta_\varepsilon(M)/\alpha$. Ces échantillons sont les solutions de $\text{M-DGS}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})\cdot\eta_\varepsilon(M)/\alpha}$.

♦

Lemme 4.09 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$. Assumons que nous avons accès à un oracle qui résout M-

$\mathrm{SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec une probabilité non négligeable. Alors, il existe un algorithme quantique à temps polynomial qui, étant donnée une module lattice M de dimension N, un nombre $r > 2q\eta_\varepsilon(M)$ et $\mathrm{poly}(N)$ échantillons à partir de $D_{M,r}$, produit un échantillon à partir de $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ avec une probabilité non négligeable.

Pour prouver le *Lemme 4.09* nous utilisons un problème intermédiaire, le $\mathrm{Mod\text{-}BDD}_\delta$ (*BDD* pour *Bounded Distance Decoding*) : étant donnée une module lattice M, $\delta < \lambda_1(M)/2$ et tout point $y \in \mathbb{R}^n$ de la forme $y = x + e$ pour un certain $x \in M$ et $\|e\|_{2,\infty} \leq \delta$, trouver x. Il est à noter que la norme $\ell_{2,\infty}$ est utilisée, parce que'elle nous aidera à énoncer le Lemme 4.12 plus convenablement.

Un autre problème intermédiaire q-$\mathrm{Mod\text{-}BDD}_\delta$ sera utilisé : étant donnée une module lattice M et un point $y \in \mathbb{R}^n$ dans la distance (par respect à la norme $\ell_{2,\infty}$) δ de M, donner en sortie le coset dans M/qM du vecteur le plus proche y. La démonstration du *Lemme 4.09* consiste en la séquence de réduction ci-dessous (il est à noter que δ prend la valeur $\frac{\alpha q\cdot\omega(\sqrt{\log n})}{\sqrt{2n}r}$).

Echantillons à partir de $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ $\xrightarrow[\text{(quantique)}]{Lemme\ 4.10}$ $\mathrm{Mod\text{-}BDD}_{M^\vee,\delta}$ $\xrightarrow{Lemme\ 4.11}$ q-$\mathrm{Mod\text{-}BDD}_{M^\vee,\delta}$ $\xrightarrow{\text{Lemme 4.12}}$ $\mathrm{M\text{-}SLWE}_{q,D_\alpha}$ Echantillon à partir de $D_{M,r}$

Lemme 4.10 :

Il existe un algorithme quantique efficace qui, étant donnée une module lattice M de dimension N, un nombre $\delta < \lambda_1(M^\vee)\omega(\sqrt{\log n})/(2\sqrt{n})$, et un oracle qui résout $\mathrm{Mod\text{-}BDD}_\delta$ sur $M^\vee$, donne à la sortie des échantillons à partir de $D_{M,r\sqrt{d}\cdot\omega(\sqrt{\log n})/(\sqrt{2}\delta)}$.

Il est à noter que selon le *Lemme 4.04*, comme $r > \sqrt{2}q \cdot \eta_\varepsilon(M)$, nous avons $\delta = \frac{\alpha q\cdot\omega(\sqrt{\log n})}{\sqrt{2n}r} < \frac{\omega(\sqrt{\log n})}{\sqrt{n}\cdot\eta_\varepsilon(M)} < \frac{\lambda_1(M^\vee)\omega(\sqrt{\log n})}{2\sqrt{n}}$.

Lemme 4.11 :

Pour tout $q \geq 2$, il existe une réduction à temps polynomial du $\mathrm{Mod\text{-}BDD}_\delta$ au q-$\mathrm{Mod\text{-}BDD}_\delta$.

Lemme 4.12 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$. Soit $M \subseteq R^d$ un R-module, et $r > \sqrt{2}q \cdot \eta_\varepsilon(M)$. Etant donné un accès à un échantillonneur oracle à partir de la distribution $D_{M,r}$, il existe une réduction probabiliste du q-Mod-BDD$_{M^\vee, \frac{\alpha q \cdot \omega(\sqrt{\log n})}{\sqrt{2}nr}}$ au M-SLWE$_{q,\Psi_{\leq\alpha}}$.

Le principe de la réduction est de construire à partir de y, l'entrée de q-Mod-BDD, et à partir de quelques échantillons Gaussiens continus ou discrets, les pairs (a, b) distribués selon $A^{(M)}_{q,s,\psi}$, où s va directement dépendre du vecteur x le plus proche de y. Pour produire de tels échantillons (a, b) suivant la distribution désirée, nous combinons les preuves correspondantes pour LWE et R-LWE. Puis d'appeler l'oracle du M-SLWE pour retourner s et nous permettre de recouvrir l'information sur x.

Démonstration :

♣

Cette section sera dédiée à la démonstration du Lemme 4.12.

Soit $\mathcal{O}$ l'oracle qui, étant donné $m \leq \text{poly}(N)$ échantillons (a,b) selon $A^{(M)}_{q,s,\psi}$ pour $\psi \in \Psi_{\leq\alpha}$, donne s à la sortie, avec une probabilité $N^{-O(1)}$. Etant donné que $M = \sum_{k=1}^{d} I_K \cdot B_k$, l'entrée de la réduction est $y = x + e$ de telle sorte que $x \in M^\vee$ et $\|e\|_{2,\infty} \leq \delta = \frac{\alpha q \omega(\sqrt{\log n})}{\sqrt{2}nr}$. Le but est de trouver $x \bmod qM^\vee$. La réduction est comme suit :

- Pour tout $l \leq m$: prendre dans un premier temps un tout nouveau z_l distribué selon $D_{M,r}$, et un tout nouveau e'_l distribué selon $D_{\alpha/\sqrt{2}}$. Dans un second lieu, poser $a_l = \Theta^{-1}(z_l \bmod qM)$ et $b_l = \frac{1}{q}\langle z_l, y\rangle + e'_l \bmod R^\vee$ (la définition de Θ est dans le paragraphe 4.4.1.1)

- Faire appel à l'oracle $\mathcal{O}$ avec comme entrée $\{(a_l, b_l)\}_{l\leq m}$. Si $\mathcal{O}$ réussit, il retourne un certain $s \in (R_q^\vee)^d$.

- Retourner à la sortie $\mathcal{O}^{-1}(s) \in M^\vee/qM^\vee$.

Le *Lemme 4.13* énonce que l'oracle $\mathcal{O}$ est utilisé d'une manière appropriée, en d'autres termes que

son entrée suit la distribution $A_{q,s,\psi}^{(M)}$. En effet, dans sa démonstration, étant donné que l'entrée de $\mathcal{O}$ est dans une distance statistique négligeable de $A_{q,s,\psi}^{(M)}$ pour une distribution $\psi \in \Psi_{\leq \alpha}$ et $s = \Theta(x \bmod qM^{\vee})$, l'oracle $\mathcal{O}$ réussit avec une probabilité non négligeable. Et que lorsqu'il réussit, alors la sortie de notre réduction est $x \bmod qM^{\vee}$.

♦

Lemme 4.13 :

Soit $\varepsilon > 0$ et $s = \Theta(x \bmod qM^{\vee})$. Il existe $\psi \in \Psi_{\leq \alpha}$ de telle sorte que la distance statistique entre $A_{q,s,\psi}^{(M)}$ et la distribution de (a, b) est tout au plus 6ε.

Démonstration :

♣

Nous démontrons dans un premier lieu que la distance statistique entre a, le premier élément de chaque échantillon, et la distribution uniforme sur $(R_q^{\vee})^d$ est tout au plus 2ε. Par le *Lemme 4.05*, la distance statistique entre la distribution de z et la distribution uniforme sur M_q est tout au plus 2ε, parce que $r \geq q \cdot \eta_{\varepsilon}(M) = \eta_{\varepsilon}(qM)$. Alors, comme $\mathcal{O}^{-1}$ induit une bijection de M_q au $(R_q)^d$, la distance statistique entre la distribution de $a = \mathcal{O}^{-1}(z \bmod qM)$ et la distribution uniforme sur $(R_q)^d$ est tout au plus 2ε.

Maintenant, nous montrons que b est de la forme $b = \frac{1}{q}\langle a, s\rangle + f$, où f distribué à partir de $D_{r'}$ avec $r_i' \leq \alpha$ pour tout i. Nous avons :

$$b = \frac{1}{q}\langle z, y\rangle + e' = \frac{1}{q}\langle z, x + e\rangle + e' = \frac{1}{q}\langle z, x\rangle + \frac{1}{q}\langle z, e\rangle + e'$$

Par définition, nous avons $z = \Theta(a) = \sum_{k=1}^{d}(t_k \cdot \mathrm{a}_k) \cdot \mathrm{b}_k \bmod qM$ avec $t_k \in I_k$ et $\mathrm{a}_k \in R_q$. Par le *Lemme 4.01*, nous avons $M^{\vee} = \sum_{k=1}^{d} I_k^{\vee} \cdot b_k^{\vee}$. Soit $x = \sum_{k=1}^{d} x_k \cdot b_k^{\vee}$. Nous savons que, $x \in I_k^{\vee} = I_k^{-1} \cdot R^{\vee}$ pour tout k. Nous avons aussi, $\langle b_k, b_{k'}^{\vee}\rangle = 1$ si $k = k'$ et $\langle b_k, b_{k'}^{\vee}\rangle = 0$ sinon. Alors modulo

$qR^\vee$:

$$\langle z, x\rangle = \sum_{k,k'=1}^{d} (t_k \cdot \mathrm{a}_k) \cdot x_{k'} \cdot (b_k \cdot \mathrm{b}_{k'}^\vee) = \sum_{k=1}^{d} (t_k \cdot \mathrm{a}_k) \cdot x_k = \sum_{k=1}^{d} \mathrm{a}_k \cdot (t_k \cdot x_k)$$

Parce que $s = \Theta(x \bmod qM^\vee) = (t_1 \cdot x_1 \bmod qR^\vee, \ldots, t_d \cdot x_d \bmod qR^\vee)^T$, nous avons :

$$\langle a, s\rangle = \sum_{k=1}^{d} \mathrm{a}_k \cdot (t_k \cdot x_k) = \langle z, x\rangle \bmod qR^\vee$$

Comme conséquence, nous obtenons que $\frac{1}{q}\langle z, x\rangle = \frac{1}{q}\langle z, x\rangle \bmod R^\vee$.

Nous allons maintenant montrer que, conditionné sur a, la quantité$\langle \frac{1}{q} z, e\rangle + e'$ a une distribution $D_{r'}$ avec $r_i' \leq \alpha$ pour tout i. Premièrement, analysons la distribution de $z' = \frac{1}{q} z$ sachant a. Nous savons que z a une distribution $D_{M,r}$ et que $a = \Theta^{-1}(z \bmod qM)$. Soit $u = \Theta(a) \bmod qM$, alors la distribution résiduelle de $z' = \frac{1}{q} z$ sachant a est $D_{M+u/q,r/q}$ (avec $r/q \geq \sqrt{2}\eta_\varepsilon(M)$).

Nous montrons par la suite que e' suit la même distribution que $\langle e'', e\rangle$ avec $e'' \sim D_{s,\ldots,s}, s = (s_i)_i$ et $s_i = s_{v-i} = \alpha / \sqrt{2\sum_{k=1}^{d} |\sigma_i(e_k)|^2}$ pour $i \in \mathbb{J}$. Un lemme sur les combinaisons linéaires des Gaussiens (Lemme 2.13 dans [4.05]) stipule que comme le vecteur e'' est distribué selon $D_{s,\ldots,s}$ et que $e \in K^d$ est fixe, nous avons $\langle e'', e\rangle$ qui a une distribution $D_{s'}$ avec $s_{i'} = s_{v-i'} = s_i \sqrt{\sum_{k=1}^{d} |\sigma_i(e_k)|^2} = \alpha/\sqrt{2}$.

Nous sommes maintenant conduits à considérer la distribution de $\langle z' + e'', e\rangle$. Nous écrivons, $e'' = e_1'' + e_2''$ avec $e_1'' \sim D_{\alpha/\sqrt{2}\delta}$ et $e'' \sim D_{s''}$ avec $(s_i'')^2 = s_i^2 - \alpha^2/2\delta^2$ (qui est positive, de par l'hypothèse sur $\|e\|_{2,\infty}$). Comme nous avons $\alpha/(\sqrt{2}\delta) = r/q$ et que $r/q = \sqrt{2}\eta_\varepsilon(M)$, un autre lemme sur les combinaisons linéaires des Gaussiens (Lemme 2.08 dans [4.05]) nous donne que la distance statistique entre la distribution de $z' + e_1''$ et $D_{\alpha/\delta}$ est tout au plus 4ε. Comme conséquence,

la distance statistique entre la distribution de $z' + e_1'' + e_2''$ et $D_{r'',\ldots,r''}$ est tout au plus 4ε avec :

$$(r_i'')^2 = \frac{\alpha^2}{\delta^2} + (s_i'')^2 = \frac{\alpha^2}{\delta^2} + s_i^2 - \frac{\alpha^2}{2\delta^2} = \frac{\alpha^2}{2\sum_{k=1}^{d}|\sigma_i(e_k)|^2} + \frac{\alpha^2}{2\delta^2}$$

Nous faisons appel une seconde fois à notre premier lemme sur les combinaisons linéaires des Gaussiens (Lemme 2.13 dans [4.05]). Avec un vecteur fixe e, nous obtenons que la distance statistique entre la distribution de $\langle z' + e'', e\rangle$ et $D_{r'}$ est tout au plus 4ε, où :

$$r_i' = \sqrt{\frac{\alpha^2}{2} + \frac{\alpha^2\sum_{k=1}^{d}|\sigma_i(e_k)|^2}{2\delta^2}}$$

Etant donné que $\delta \geq \sqrt{\sum_{k=1}^{d}|\sigma_i(e_k)|^2}$, nous avons $r_i' \leq \alpha$, comme souhaité.

♦

Tout ceci conclue la démonstration de la première partie de notre *Théorème 4.07*.

Pour la seconde partie du théorème (existence d'une réduction du M-SLWE au M-LWE), nous invitons nos lecteurs à consulter la démonstration dans notre bibliographie (paragraphe 4.3 dans [4.05]).

4.5 Cryptosystème CRYSTALS

Le cryptosystème *CRYSTALS* ou *Cryptographic Suite for Algebraic Lattices* encapsule deux types de cryptosystèmes : *Kyber*, un KEM (Key Encapsulation Mechanism, pour rappel) ayant la spécificité d'être sécurisé IND-CCA2 (nous verrons la définition de ce terme un peu plus loin dans notre rédaction), et *Dilithium* un algorithme de signature sécurisé EUF-CMA (même cas que l'IND-CCA2, une définition sera donnée plus loin). Les deux algorithmes sont basés sur les problèmes difficiles relatifs aux module lattices, ont été par conséquent construits pour résister aux éventuelles attaques d'ordinateurs quantiques.

Le CRYSTALS, comme mentionné dans l'introduction du présent chapitre, a été soumis au projet NIST de cryptographie post-quantique. Il a pu progresser jusqu'à l'étape 3 dudit projet de standardisation. Mais surtout, le seul parmi les quelques-uns qui ont progressé à cette étape finale,

à posséder à la fois un cryptosystème type KEM et signature.

4.5.1 Préliminaires

4.5.1.1 Classe d'indiscernabilité pour les cryptosystèmes type KEM, le préfixe IND- (de INDistinguishability en anglais)

Tout cryptosystème se trouve être attribué par leur créateur un certain niveau de sécurité. Les plus communs sont les classifications de type *IND-* (de *INDistinguishability* en anglais) ou encore *indiscernabilité* en français pour faire simple. [4.10]

Le cryptosystème idéal E (de *Encryption* en anglais) est celui qui, pour tout texte chiffré $C = E(K, M)$ (C pour *Cyphertext*, K pour *Key* et M pour *Message*), si la clé K reste secret pour l'adversaire , alors la probabilité d'identifier M est négligeable. Etant donné que c'est difficilement atteignable en pratique. L'approche la plus raisonnable est de définir des contraintes suffisamment rigides pour satisfaire quelques définitions de sécurité. La notation en *IND-* donne une telle description en termes de jeux où un challenger (le créateur du cryptosystème) retient sa clé d'une manière secrète, et que l'adversaire possède certaines capabilités et son but est de casser le cryptosystème.

Pour rester dans la généralité, le cryptosystème aura un algorithme de génération de clé *KG* (pour *Key Generation*), qui génèrera une paire de clé K_E et K_D (respectivement *Encryption Key* et *Decryption Key*), un algorithme de chiffrement E, et un autre de déchiffrement D (pour *Decryption*). Un chiffre se doit de toujours être inversible, mais la clé de chiffrement et celle de déchiffrement peuvent être différentes (pour prendre en compte les cryptosystèmes à clé publique) : $D\big(K_D, E(K_E, M)\big) = M$.

a. *IND-CPA: INDistinguishability under Chosen Plaintext Attack*

L'adversaire génère deux messages de longueur égale, le challenger décide de chiffrer l'un d'entre eux d'une manière aléatoire. L'adversaire essaie de deviner lequel des deux messages a été chiffré. L'algorithme correspondant est décrit ci-dessous :

- Challenger : $K_E, K_D = KG$ (paramètre de sécurité) ;

- Faire des opérations en un temps polynomial incluant l'appel à l'oracle de chiffrement ;

- Adversaire : $m_0, m_1 =$ choisir deux messages de longueur identique. Envoyer m_0, m_1 au challenger ;

- Challenger : $b =$ choisir aléatoirement entre 0 et 1 ;

- Challenger : $C \coloneqq E(K_E, m_b)$. Envoyer C à l'adversaire ;

- Adversaire : faire des opérations additionnelles en un temps polynomial incluant des appels à l'oracle de chiffrement. Donner en sortie $guess$;

- Si $guess = b$, l'adversaire gagne.

Le présent concept introduit une notion importante de majoration polynomiale. Ce que l'on attendait du cryptosystème est donc un peu altéré, d'une *probabilité de gagner négligeable* pour l'adversaire à une *probabilité de gagner* ***durant un temps raisonnable***, *négligeable*. La restriction sur la longueur du message (que celle-ci soit la même) évite à l'adversaire de gagner par simple comparaison de la longueur du message crypté.

b. *IND-CCA1: INDistinguishability under (non adaptive) Chosen Ciphertext Attack*

Le but du présent jeu est le même que dans le cas du IND-CPA. L'adversaire a une additionnelle possibilité : celle de pouvoir appeler en plus de l'oracle de chiffrement, celui de déchiffrement. Pour plus de précision, l'adversaire peut faire un chiffrement ou déchiffrement de messages arbitraires avant de recevoir le message chiffré du jeu. L'algorithme en est :

- Challenger : $K_E, K_D = KG$ (paramètre de sécurité) ;

- L'adversaire (un nombre de fois limité à une grandeur polynomial) fait appel à l'oracle de chiffrement et de déchiffrement sur un message en clair et chiffré arbitraire, respectivement ;

- Adversaire : $m_0, m_1 =$ choisir deux messages de longueur identique ;

- Challenger : $b =$ choisir aléatoirement entre 0 et 1 ;

- Challenger : $C \coloneqq E(K_E, m_b)$. Envoyer C à l'adversaire ;

- Adversaire : faire des opérations additionnelles en un temps polynomial. Donner en sortie $gess$;

- Si $guess = b$, l'adversaire gagne.

IND-CCA1 considère la possibilité de répéter l'interaction, impliquant que la sécurité n'est pas altérée dans le temps.

c. *IND-CCA2: INDistinguishability under adaptive Chosen Ciphertext Attack*

En addition au IND-CCA1, l'adversaire a maintenant accès à l'oracle de déchiffrement après avoir reçu C, mais ne peut envoyer C à l'oracle de déchiffrement. L'algorithme en est :

- Challenger : $K_E, K_D = KG$ (paramètre de sécurité) ;

- L'adversaire (autant de fois qu'il le veut) fait appel l'oracle de chiffrement et de déchiffrement sur un message en clair et chiffré arbitraire, respectivement ;

- Adversaire : $m_0, m_1 =$ choisir deux messages de longueur identique ;

- Challenger : $b =$ choisir aléatoirement entre 0 et 1 ;

- Challenger : $C \coloneqq E(K_E, m_b)$. Envoyer C à l'adversaire ;

- Adversaire : faire des opérations additionnelles en un temps polynomial, incluant l'appel à l'oracle pour tout texte chiffré différent de C. Donner en sortie $gess$;

- Si $guess = b$, l'adversaire gagne.

IND-CCA2 suggère que l'utilisation de l'oracle après avoir reçu le texte chiffré peut donner un certain avantage étant donné que l'oracle aurait pu être mieux adapté à un texte chiffré spécifique.

4.5.1.2 xUF-CMA, catégories de sécurité de cryptosystèmes type signature

Il existe deux définitions formelles pour la sécurité d'algorithmes de signatures digitales. Chacune de ces définitions sont présentées en tant que jeu, ou une expérience qui est effectuée entre un adversaire et un challenger. [4.11]

a. *EUF-CMA*

L'expérience *EUF-CMA* ou *Existantial Unforgeability under Chosen Message Attack* marche

comme suit :

- Le challenger génère une paire de clé (pk, sk) (respectivement *public* et *private* key) et donne pk à l'adversaire ;

- L'adversaire pourrait maintenant, d'une manière répétée, demander la signature correspondante aux messages choisis $(M_1, \dots, M_q)$ et recevoir les signatures correspondantes et valides $(\sigma_1, \dots, \sigma_q)$ en réponse ;

- Pour conclure l'expérience, l'adversaire doit fournir un message et signature M^*, σ^* de telle sorte que (1) le message M^* ne soit pas un des messages précédemment demandé et que (2) le couple message/signature soit valide par respect à la clé publique fournie.

L'algorithme est considéré comme sécurisé si aucun adversaire n'a un avantage non-négligeable à remplir les conditions ci-dessus.

La définition donnée plus haut est assez robuste, mais il existe une autre version plus stricte.

b. *SUF-CMA*

L'expérience *SUF-CMA* ou Strong e*xistantial Unforgeability under Chosen Message Attack* marche comme suit. Les deux premières étapes sont les mêmes que précédemment :

- La même que la première étape de L'EUF-CMA ;

- La même que la seconde étape de L'EUF-CMA ;

- Pour conclure l'expérience, l'adversaire doit fournir un message et signature M^*, σ^* de telle sorte que (1) le message M^* ne soit pas un des messages précédemment demandé et que la signature ne soit renvoyée durant l'étape précédente, et que (2) le couple message/signature soit valide par respect à la clé publique fournie.

L'adversaire gagne si les conditions plus haut sont satisfaites.

La principale différence est que cette définition plus forte garantit que l'attaquant ne peut pas « altérer » la signature. Par exemple, un algorithme dans lequel l'attaquant peut re-randomiser une signature valide afin qu'elle soit toujours valide, mais qu'elle ait un aspect différent de la valeur

d'origine, ne satisferait pas SUF-CMA.

4.5.2 CRYSTALS Kyber

a. *Généralités*

CRYSTALS Kyber (ou Kyber tout court pour simplifier) est un cryptosystème type KEM (Key Encapsulation Mecanism) sécurisé IND-CCA2. La sécurité de Kyber est basée sur la difficulté de la résolution du problème Learning With Errors other Module lattices ou MLWE. La construction de Kyber est subdivisée en une approche en deux étapes : dans un premier temps, on fait appel à un algorithme de cryptage à clé publique, sécurisé IND-CPA, pour agir sur un message de longueur fixe égal à 32 bytes. Cette première étape se nomme Kyber.CPAPKE. Une version légèrement modifiée de la transformation de Fujisaki–Okamoto sera utilisée par la suite pour construire le KEM sécurisé IND-CCA2. Cette seconde étape du cryptosystème Kyber est dénommée par Kyber.CCAKEM. [4.01]

b. *Justifications de conception*

La construction de Kyber est basée sur la version module du cryptosystème Ring-LWE LPR (LPR sont des initiales provenant du nom de leurs inventeurs Lyubashevsky, Peikert et Regev [4.12], [4.13]) cryptosystème faisant intervenir la technologie *bit-dropping* (nous verrons la signification de cette technologie un peu plus loin dans nos écrits). Il est également amélioré suite à de nombreuses modifications dans les implémentations précédentes de cryptosystèmes à base de lattice tels que *NewHope*. Dans NewHope (et tous les autres cryptosystèmes basés sur le Ring-LWE), les opérations étaient de la forme $As + e$ où toutes les variables sont des polynômes dans un anneau. La principale différence dans Kyber est que A est maintenant une matrice (avec une dimension assez petite, 3 par exemple) qui agit sur un anneau polynomial de taille constante et s et e sont des vecteurs sur le même anneau. Nous nous référons à cela par le terme *cryptosystème à base de module lattices*.

- L'usage du *Module-LWE*. Les propositions précédentes de cryptosystèmes basés sur LWE utilisaient soit le problème Ring-LWE très structuré (comme, par exemple, NewHope), soit le LWE standard (comme, par exemple, Frodo). Le principal avantage des variantes LWE structurées basées sur des anneaux polynomiaux est l'efficacité en termes de rapidité d'exécution et de tailles de la clé et du texte chiffré. Les inconvénients sont les craintes que la structure additionnelle puisse permettre des attaques plus efficaces et que le compromis entre efficacité et sécurité ne peut être dimensionné

que de manière peu convenable. Les avantages du LWE standard sont l'absence de structure et la praticité du dimensionnement, mais ceux-ci se font au prix d'une efficacité considérablement réduite. Module-LWE offre un compromis entre ces deux extrêmes. Dans le cas spécifique des paramètres Module-LWE utilisés dans Kyber, nous obtenons une structure réduite d'une manière convenable, par rapport au Ring-LWE, donc une bien meilleure flexibilité. En addition à cela - lors du chiffrement de messages d'une taille fixe de 256 bits - des performances très similaires aux cryptosystèmes basés sur le Ring-LWE sont au rendez-vous.

- Utilisation de la *NTT* ou *Number Theorithic Transform*. Nous verrons la définition détaillée de cette technique dans notre chapitre final, chapitre 5. Pour le moment, il est suffisant de savoir que la NTT est un moyen très efficace d'effectuer des multiplications dans $\mathbb{R}_q$. La multiplication dans $\mathbb{R}_q$ basée sur la NTT présente de multiples avantages : elle est extrêmement rapide, ne nécessite pas de mémoires supplémentaires (comme, par exemple, la multiplication de Karatsuba ou Toom [4.01]) et peut être effectuée dans un espace de code réduit. Par conséquent, il est devenu pratique courante de choisir des paramètres de cryptosystème, à base de lattice, qui peuvent prendre en charge cet algorithme de multiplication. Certains cryptosystèmes vont plus loin et intègrent le NTT dans leurs définitions. Un exemple marquant est encore une fois NewHope, qui échantillonne la valeur publique a dans le domaine NTT et envoie également les clés publiques et les textes chiffrés dans le domaine NTT.

Dans Kyber, il a été également décidé d'intégrer la NTT dans la définition du cryptosystème, mais uniquement dans l'échantillonnage de A et de la clé publique, pas pour l'élaboration du texte chiffré. Une conséquence de cette décision est que la NTT apparaît dans la spécification de Kyber.CPAPKE. Notons que les multiplications par A doivent utiliser la NTT, simplement du fait que $\hat{A}$ ait été échantillonné dans le domaine NTT. De même, les multiplications par la clé publique $\hat{t}$ doivent utiliser la NTT, car la clé publique est transmise dans le domaine NTT. En conséquence, les implémentations voudront également utiliser la NTT pour toutes les autres multiplications, des appels explicites de la NTT et NTT^{-1} seront donc visible dans algorithmes composant le cryptosystème Kyber. Notons également que la clé privée sk est également stockée dans le domaine NTT.

Le fait de ne pas intégrer le NTT dans la définition de Kyber aurait pu être choisi, ce qui aurait pu améliorer la simplicité de sa description. Le coût de cette simplicité accrue additionnerait cependant,

k^2 opérations NTT supplémentaires à la fois dans la génération de clés et dans la partie encapsulation, ce qui entraînerait un ralentissement significatif. Le fait de ne pas encoder la clé publique dans le domaine NTT aurait pu être également choisi, cependant, cela se ferait au prix du besoin de NTT supplémentaires. Enfin, le fait d'envoyer également le texte chiffré dans le domaine NTT aurait également pu être choisi, cependant, cela serait incompatible avec la compression de texte chiffré via la fonction Compress_q (nous verrons les détails de toutes ces fonctions dans notre prochain et final chapitre, chapitre 5).

- *Contre toute autorité* ou *Against all authority* en anglais. Pour la génération de la matrice publique A, uniformément aléatoire, il a éte décidé d'adopter l'approche *« Against all authority »* de NewHope. Cela signifie que la matrice n'est pas un paramètre système mais qu'elle est plutôt générée fraîchement comme partie intégrante de chaque clé publique. Cette approche présente deux avantages. Premièrement, cela évite les discussions sur la manière suivant laquelle un paramètre uniformément aléatoire du cryptosystème, a été généré. Deuxièmement, il protège contre le scénario d'attaque tout-pour-le-prix-d'un (all-for-the-price-of-one en anglais) d'un attaquant utilisant une quantité importante de puissance de calcul pour trouver une base réduite de la lattice engendrée par la matrice A une fois, puis en utilisant cette base réduite pour attaquer tous les utilisateurs. Le coût de cette décision est l'expansion de la matrice A à partir d'une *graine aléatoire* (*random seed* en anglais) pendant la génération et l'encapsulation de clé.

- *Erreur Binomiale*, ou *Binomial Noise* en anglais. Les traitements théoriques constituant les cryptosystèmes basés sur le LWE considèrent généralement le LWE avec un bruit Gaussien, que ce soit du Gaussien arrondi ou Gaussien discret. En conséquence, de nombreuses mises en œuvre précoces ont échantillonné le bruit à partir d'une distribution Gaussienne discrète, ce qui s'avère être soit assez inefficace, soit vulnérable aux attaques de synchronisation [4.01]. Les performances des attaques les plus connues contre le chiffrement basé sur le LWE ne dépendent pas de la distribution exacte du bruit, mais plutôt de son écart type (et potentiellement de l'entropie). Cela motive l'utilisation de distributions de bruit dont nous pouvons échantillonner d'une manière simple, efficace et en toute sécurité. Un exemple en est l'utilisation de la distribution binomiale centrée. Un autre exemple est l'utilisation du LWR (du Learning With Rounding), qui ajoute un bruit uniforme déterministe en supprimant des bits comme dans la fonction $Compress_q$ de Kyber. Dans la conception de Kyber, il a été décidé d'utiliser le bruit binomial centré et donc de nous fier au LWE au lieu du LWR comme problème sous-jacent. La compression des textes chiffrés via Compress_q

introduit du bruit supplémentaire (rendant le schéma plus sécurisé), mais nous ne considérons pas ce bruit dans notre analyse de sécurité (ce choix est motivé par l'absence de variante Ring / Module d'une réduction de complexité pour le LWR).

- *Permettre les erreurs de décapsulation*. Une autre décision de conception intéressante est d'autoriser les erreurs de décapsulation (c'est-à-dire les erreurs de décryptage dans Kyber.CPAPKE) ou de choisir des paramètres qui ont, non seulement une probabilité d'erreur négligeable, mais aussi de valeur nulle. Les avantages de la probabilité de défaillance nulle sont évidents : les transformations CCA et les preuves de sécurité deviennent plus faciles et on aurait pu éviter toute une discussion sur les attaques exploitant les échecs de décapsulation. L'inconvénient de concevoir un chiffrement basé sur le LWE sans erreurs est que cela implique, soit une diminution de la sécurité contre les attaques ciblant les problèmes de lattice sous-jacente (en diminuant considérablement le bruit), soit une diminution des performances (en compensant la perte de sécurité via une augmentation de la dimension de la lattice). La décision d'autoriser des probabilités d'échec $< 2^{-160}$ dans tous les ensembles de paramètres de Kyber reflète l'intuition que les échecs de décapsulation sont un problème s'ils apparaissent avec une probabilité non négligeable. Mais, les attaques qui tentent d'exploiter les erreurs qui se produisent avec une probabilité extrêmement faible comme dans Kyber sont une menace beaucoup plus petite que, par exemple, les améliorations aux attaques hybrides ciblant des cryptosystèmes avec un bruit très faible.

4.5.3 CRYSTALS Dilithium

a. *Généralités*

Dans les paragraphes qui vont suivre, nous allons présenter le cryptosystème de signature numérique Dilithium, dont la sécurité est basée sur la difficulté du SVP dans les lattices. L'algorithme a été conçu en tenant compte des critères suivants :

- *Facile à implémenter d'une manière sécurisée*. Les cryptosystèmes les plus compacts, type signature, basés sur les lattices, nécessitent obligatoirement la génération d'une valeur aléatoire secrète à partir de la distribution gaussienne discrète. Générer de tels échantillons de manière sécurisée contre les attaques par canal auxiliaire (*side-channel attacks* en anglais) est ardu et peut facilement conduire à des implémentations non sécurisées. Bien qu'il soit possible qu'une mise en œuvre très soignée puisse empêcher de telles attaques, il n'est pas raisonnable de supposer qu'un

cryptosystème, qui va être déployé à large échelle, et contenant de nombreuses subtilités, sera toujours implémenté de manière minutieuse. Le Dilithium n'utilise donc qu'un échantillonnage uniforme. De plus, toutes les autres opérations (telles que la multiplication polynomiale et l'arrondi) sont facilement mises en œuvre en durée constante.

- *Adopter des paramètres d'efficacité qui durent dans le temps*. Puisque nous visons une sécurité à long terme, il a été analysé l'applicabilité des attaques contre les lattices d'un point de vue très favorable à l'attaquant. En particulier, des algorithmes quantiques qui nécessitent pratiquement autant d'espaces que de temps ont été envisagés. De tels algorithmes sont actuellement irréalistes, et il semble y avoir de sérieux obstacles à éliminer les besoins en espace, mais la possibilité que des améliorations puissent se produire à l'avenir a été prévue.

- *Minimisez la taille de la clé publique et aussi celle de la signature*. Étant donné que de nombreuses applications nécessitent la transmission à la fois de la clé publique et de la signature (par exemple, des chaînes de certificats), le cryptosystème Dilithium a été conçu pour minimiser la somme de ces paramètres. Etant donné que l'échantillonnage gaussien (discret) a été évité, Dilithium a la plus petite combinaison de tailles de signature et de clé publique parmi tous les cryptosystèmes basés sur une lattice, présentant les mêmes niveaux de sécurité.

- *Cryptosystème modulaire, à niveau de sécurité facilement modifiable*. Les deux opérations qui constituent la quasi-totalité des procédures de signature et de vérification sont l'expansion d'un XOF (SHAKE-128 et SHAKE-256 sont utilisés dans le cas de Dilitium), et la multiplication dans l'anneau polynomial $\mathbb{Z}_q[X]/(X^n + 1)$. Les implémentations très efficaces de notre algorithme devront donc optimiser ces opérations et s'assurer qu'elles s'exécutent en durée constante. Pour tous les niveaux de sécurité, notre cryptosystème utilise le même anneau avec $q = 2^{23} - 2^{13} + 1$ et $n = 256$. Varier la sécurité implique simplement de faire plus ou moins d'opérations sur cet anneau et de faire plus ou moins d'expansion du XOF. En d'autres termes, une fois qu'une implémentation optimisée est obtenue pour un certain niveau de sécurité, il est relativement aisé d'obtenir une implémentation optimisée pour un niveau supérieur/inférieur.

b. *Aperçu de l'approche de base*

Nous présentons dans ce qui va suivre une version simplifiée de ce qui compose le cryptosystème type signature Dilithium. Ci-dessous donc les étapes constitutives dudit cryptosystème, pour donner

au lecteur une idée de son fonctionnement :

- *Génération de la clé*. L'algorithme de génération de clé génère une matrice A de dimension $k \times l$ dont chacune des entrées est un polynôme dans l'anneau $\mathbb{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. Comme mentionné précédemment, nous aurons toujours $q = 2^{23} - 2^{13} + 1$ et $n = 256$. Ensuite, l'algorithme échantillonne les vecteurs clés secrètes aléatoires s_1 et s_2. Chaque coefficient de ces vecteurs est un élément de $\mathbb{R}_q$ avec de petits coefficients de taille tout au plus égale à η. Enfin, la deuxième partie de la clé publique est calculée comme suit $t = As_1 + s_2$. Toutes les opérations algébriques de ce cryptosystème sont supposées être effectuées sur l'anneau polynomial $\mathbb{R}_q$.

- *Procédure de signature*. L'algorithme de signature génère un vecteur de masquage de polynômes y avec des coefficients inférieurs à γ_1. Le paramètre γ_1 est défini stratégiquement - il est suffisamment grand pour que la signature éventuelle ne révèle pas la clé secrète (c'est-à-dire que l'algorithme de signature est à *divulgation nulle de connaissance*, *zero-knowledge* en anglais), mais suffisamment petit pour que la signature ne soit pas facilement copiable d'une manière frauduleuse. Le signataire calcule alors Ay et définit w_1 pour être les bits les plus significatifs des coefficients dans ce vecteur. En particulier, tout coefficient ω dans Ay peut s'écrire de manière canonique comme $\omega = \omega_1 \cdot 2\gamma_2 + \omega_0$ où $|\omega_0| \leq \gamma_2$; w_1 est alors le vecteur comprenant tous les ω_1. Le défi c est alors créé comme étant le hachage du message et de w_1. La sortie c est un polynôme dans $\mathbb{R}_q$ avec exactement ± 1 au nombre de 60 et le reste des 0. La raison de cette distribution est que c a une norme petite et qu'il provient d'un domaine de taille $> 2^{256}$. La signature potentielle est alors calculée comme $z = y + cs_1$.

Si z était rendu (son évaluation est terminée) directement à ce stade, alors le cryptosystème type signature ne serait pas sécurisé en raison du fait que la clé secrète serait divulguée. Pour éviter la dépendance de z par rapport à la clé secrète, nous utilisons un rejet d'échantillonnage. Le paramètre β est défini comme étant le coefficient maximum possible de cs_i. Puisque c a des ± 1 au nombre de 60 et que le coefficient maximum dans s_i est η, il est facile de constater que $\beta \leq 60\eta$. S'il existe un coefficient de z supérieur à $\gamma_1 - \beta$, nous rejetons et redémarrons la procédure de signature. De plus, si un coefficient des bits de poids faible de $Az - ct$ est supérieur à $\gamma_2 - \beta$, nous redémarrons également. Le premier contrôle est nécessaire pour la sécurité, tandis que le second est nécessaire à la fois pour la sécurité et l'exactitude. La boucle *while* dans la procédure de signature se répète jusqu'à ce que les deux conditions précédentes soient satisfaites. Les paramètres sont fixés de telle

sorte que le nombre de répétitions attendu ne soit pas trop élevé (dans nos instanciations, ce nombre est compris entre 4 et 7).

- *Vérification*. Le vérificateur calcule d'abord w_1' comme étant les bits de poids fort de $Az - ct$, puis accepte si tous les coefficients de z sont inférieurs à $\gamma_1 - \beta$ et si c est le hachage du message et w_1'. Voyons pourquoi la vérification fonctionne, en particulier pourquoi $\text{HighBits}(Az - ct, 2\gamma_2) = \text{HighBits}(Ay, 2\gamma_2)$. La première chose à remarquer est que $Az - ct = Ay - cs_2$. Donc, tout ce que nous devons vraiment montrer, c'est que :

$$\text{HighBits}(Ay, 2\gamma_2) = \text{HighBits}(Ay - cs_2, 2\gamma_2) \quad \textbf{(}4.18\textbf{)}$$

La raison en est qu'une signature valide aura $\|\text{LowBits}(Ay - cs_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta$. Et puisque nous savons que les coefficients de cs_2 sont plus petits que β, nous savons que l'ajout de cs_2 n'est pas suffisant pour provoquer des portées en augmentant n'importe quel coefficient d'ordre inférieur pour avoir une magnitude au moins égal à γ_2. Ainsi notre formule **(**4.18**)** est vraie et la signature correctement vérifiée.

4.6 Apport personnel et innovation des travaux de recherches

L'apport personnel et l'innovation de nos travaux de recherche reposent sur les deux principaux points ci-dessous :

- Le fait d'avoir pu mettre en exergue une équivalence, en termes de difficulté, entre les problèmes généraux relatifs à la technologie mathématique Lattice (Mod-GIVP, ou Generalized Independent Vector Problem restreint sur les Module Lattices, en particulier) et le MLWE (Learning With Error over Modules) ;

- Le fait d'avoir pu identifier que la technologie MLWE est la plus prometteuse, en termes de fondement mathématiques des cryptosystèmes post-quantiques actuellement en cours de normalisation pour être utilisés (les cryptosystèmes post-quantiques) à large échelle. Nous faisons allusion bien sûr au cryptosystème CRYSTALS, composé de Kyber (cryptosystème type KEM, ou Key Encapsulation Mechanism) et Dilithium (cryptosystème type signature). CRYSTALS est le seul gagnant du projet Post-Quantum Cryptography Standardization, initié par le NIST (National

Institute of Standards and Technology), qui a pu faire élire à la fois un cryptosystème type KEM et signature.

4.7 Conclusion

Dans le présent chapitre, nous avons pu introduire une nouvelle famille de technologie mathématique : le LWE, le R-LWE et le MLWE.

Le LWE (ou Learning With Error), dans un premier temps, est une généralisation du problème Learning from parity with error. Il a été démontré que ce problème est équivalent, en termes de difficulté, au SIVP ou Shortest Independent Vector Prolem d'une lattice. Les avantages du LWE standard sont l'absence de structure et la praticité du dimensionnement, mais ceux-ci se font au prix d'une efficacité considérablement réduite.

Dans un second lieu vient la technologie R-LWE (ou Ring Learning With Error) qui revient à rajouter une structure algébrique supplémentaire au LWE standard dans le but de le rendre plus efficace. Le principal avantage des variantes LWE structurées basées sur des anneaux polynomiaux est l'efficacité en termes de rapidité d'exécution et de tailles de la clé et du texte chiffré. Les inconvénients sont les craintes que la structure additionnelle puisse permettre des attaques plus efficaces et que le compromis entre efficacité et sécurité ne peut être dimensionné que de manière peu convenable.

Vient par la suite, en troisième point, le MLWE (ou Learning With Error other Modules). Les module lattices peuvent être considérés comme des lattices situées entre ceux utilisés dans les définitions du problème LWE et ceux utilisés pour le problème R-LWE. Le Module-LWE offre un compromis entre les deux extrêmes que constituent le LWE et le R-LWE.

Nous avons par la suite terminé le présent chapitre par un aperçu du cryptosystème CRYSTALS basé sur l'outil mathématique MLWE. CRYSTALS est composé de Kyber, un cryptosystème type KEM (Key Encapsulation Mechanism), et de Dilithium, un cryptosystème type signature.

CHAPITRE 5
APPLICATION DU MLWE DANS LE CRYPTOSYSTEME CRYSTALS - KYBER COMME KEM ET DILITHIUM COMME SIGNATURE

5.1 Introduction

Dans le présent chapitre, nous allons détailler chacun des blocs qui constituent le cryptosystème CRYSTALS, d'expliciter également au passage, comment ledit cryptosystème se comporte-t-il en terme de performance et de sécurité.

Le cryptosystème CRYSTALS est, rappelons-le, formé par un cryptosystème de type KEM et un autre de type signature, respectivement dénommés CRYSTALS Kyber et CRYSTALS Dilithium. Nous allons par conséquent entamer notre tâche en commençant par le cryptosystème CRYSTALS Kyber, pour poursuivre par la suite avec le second qu'est le cryptosystème CRYSTALS Dilithium.

Nous avons publié nos travaux dans le Journal Scientifique de notoriété nationale, MADA-ETI, et dans le Journal Scientifique, de notoriété internationale, IJARIIE :

- « Application du MLWE dans le cryptosystème CRYSTALS - Kyber comme KEM et Dilithium comme signature », MADA-ETI, ISSN 2220-0673, Vol.2, 2021, www.madarevues.gov.mg

- « Application of the MLWE in the cryptosystem CRYSTALS – KYBER as KEM and Dilithium as signature », IJARIIE-ISSN(O)-2395-4396, Vol-7 Issue-5 2021

5.2 Cryptosystème type KEM, CRYSTALS Kyber

5.2.1 Spécification écrite

5.2.1.1 Préliminaires et notations

Octets et tableau d'octets : toutes les entrées et sorties des fonctions API (pour *Application Programming Interface*) de Kyber sont des tableaux d'octets. Pour simplifier la notation, nous allons dénoter par $\mathcal{B}$ l'ensemble $\{0, \dots, 255\}$, c'est-à-dire, l'ensemble de 8 bits d'entiers non signés. Par conséquent, nous dénotons par $\mathcal{B}^k$ l'ensemble de tableaux d'octets de longueur k et par $\mathcal{B}^*$ l'ensemble de tableaux d'octets de longueur arbitraire (ou flux d'octets). Pour deux tableaux d'octet a et b, nous dénotons par $a||b$ la concaténation de a et b. Pour un tableau d'octets a, nous dénotons

par $a + k$ le tableau d'octets commençant par l'octet k de a (avec l'indexation commençant par 0). Par exemple, soit a un tableau d'octets de longueur l, et soit b un autre tableau d'octets et soit $c = a||b$ la concaténation de a et b ; alors $b = a + l$. Et comme il est plus convenable de travailler avec un tableau de bits qu'avec un tableau d'octets, nous allons rendre cette conversion explicite par la fonction BytesToBits qui prend en entrée un tableau de l octets et produit à la sortie un tableau de $8l$ bits. Le bit β_i à la position i de la sortie est obtenu depuis l'octet $b_{i/8}$ à la position $i/8$ du tableau d'entrée en calculant $\beta_i = \left(\left(b_{i/8}/2^{i \bmod 8}\right) \bmod 2\right)$.

Anneau polynomial et vecteurs : Nous dénotons par R l'anneau $\mathbb{Z}[X]/(X^n + 1)$ et par R_q l'anneau $\mathbb{Z}_q[X]/(X^n + 1)$ où $n = 2^{n'-1}$ de telle sorte que $X^n + 1$ soit le $2^{n'}$-ième polynôme cyclotomique. Dans le présent ouvrage, les valeurs de n, n' et q sont fixées à $n = 256$, $n' = 9$ et $q = 3329$. Les lettres de police régulière dénotent les éléments de R ou R_q (qui incluent les éléments de $\mathbb{Z}$ et $\mathbb{Z}_q$) et les lettres en minuscules grasses représentent les vecteurs dont les coefficients sont dans R ou R_q. Par défaut, tous les vecteurs sont des vecteurs colonnes. Les lettres en majuscules gras sont des matrices. Pour un vecteur $\mathbf{v}$ (ou une matrice $\mathbf{A}$), nous dénotons par $\mathbf{v}^{\mathrm{T}}$ ou $\mathbf{A}^{\mathrm{T}}$ sa transposée. Pour un vecteur $\mathbf{v}$ nous écrivons $\mathbf{v}[i]$ pour dénoter son i-ième entrée (avec une indexation qui commence par 0) ; Pour une matrice $\boldsymbol{A}$ nous écrivons $\boldsymbol{A}[i][j]$ pour dénoter l'entrée dans la ligne i et la colonne j, et encore une fois, les indexations commencent par 0.

Réductions modulaires : Pour un nombre entier positif α pair (resp. impair), nous définissons $r' = r \bmod^{\pm} \alpha$ comme étant l'élément unique r dans la fourchette $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ (resp. $-\frac{\alpha-1}{2} < r' \leq \frac{\alpha-1}{2}$), de telle sorte que $r' = r \bmod \alpha$. Pour tout entier positif α, nous définissons $r' = r \bmod^{+} \alpha$ pour être l'unique élément r' de la fourchette $0 \leq r' < \alpha$, de telle sorte que $r' = r \bmod \alpha$. Quand la représentation exacte n'est pas significative, nous écrivons tout simplement $r \bmod \alpha$.

Arrondissement : Pour un élément $x \in \mathbb{Q}$ nous dénotons par $\lceil x \rfloor$ l'arrondissement de x à l'entier le plus proche, le juste milieu étant arrondi au supérieur ($\lceil n + 1/2 \rfloor = n + 1$ pour n entier).

Normes des éléments : pour un élément $w \in \mathbb{Z}_q$, nous écrivons $\|w\|_\infty$ pour signifier $|w \bmod^{\pm} q|$.

Nous définissons maintenant les normes l_∞ et l_2 pour $w = w_0 + w_1X + \cdots + w_{n-1}X^{n-1} \in R$:

$$\|w\|_\infty = \max_i \|w_i\|_\infty,\ \|w\| = \sqrt{\|w_0\|_\infty^2 + \cdots + \|w_{n-1}\|_\infty^2} \quad \textbf{(}5.01\textbf{)}$$

D'une manière similaire pour $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$, nous définissons :

$$\|\mathbf{w}\|_\infty = \max_i \|w_i\|_\infty,\ \|\mathbf{w}\| = \sqrt{\|w_1\|^2 + \cdots + \|w_k\|^2} \quad \textbf{(}5.02\textbf{)}$$

Ensembles et distributions : Pour un ensemble S, nous écrivons $s \longleftarrow S$ pour dénoter que *s* est choisi d'une manière uniformément aléatoire à partir de S. Si S est une probabilité de distribution, alors cela dénote que *s* est choisi selon la distribution S.

Compression et décompression : Nous définissons la fonction $\text{Compress}_q(x, d)$ qui prend un élément $x \in \mathbb{Z}_q$ et donne à la sortie un entier dans $\{0, \ldots, 2^d - 1\}$ où $d < \lceil \log_2(q) \rceil$. Nous définissons par la suite Decompress_q de telle sorte que :

$$x' = \text{Decompress}_q\big(\text{Compress}_q(x, d), d\big) \quad \textbf{(}5.03\textbf{)}$$

soit un élément proche de x, plus spécifiquement :

$$x' - x \bmod^{\pm} q \leq B_q \coloneqq \left\lfloor \frac{q}{2^{d+1}} \right\rfloor \quad \textbf{(}5.04\textbf{)}$$

Les fonctions qui justifient ces exigences sont définies comme :

$$\begin{aligned} \text{Compress}_q(x, d) &= \lceil (2^d/q).x \rfloor \bmod^+ 2^d \\ \text{Decompress}_q(x, d) &= \lceil (q/2^d).x \rfloor \end{aligned} \quad \textbf{(}5.05\textbf{)}$$

Quand Compress_q et Decompress_q sont utilisées avec $x \in R_q$ ou $\mathbf{x} \in R_q^k$, la procédure est appliquée à chaque coefficient individuellement.

La principale raison dans le fait d'avoir défini les fonctions Compress_q et Decompress_q est

l'habilité de pouvoir écarter les bits de rang inférieur dans le texte chiffré. Ces bits n'ont pas d'effets considérables dans la probabilité d'exactitude dans le processus de déchiffrement, réduisant ainsi la taille du texte chiffré.

Les fonctions $\mathrm{Compress}_q$ et $\mathrm{Decompress}_q$ sont également utilisées à des fins autres que la compression, on pourra noter la correction d'erreur LWE durant la phase de chiffrement et de déchiffrement. Plus précisément dans la ligne 20 de la procédure de chiffrement (**Algorithme 5.5**) la fonction $\mathrm{Decompress}_q$ est utilisée pour créer des écarts de tolérance d'erreurs en envoyant le bit message 0 pour 0 et 1 pour $\lceil q/2 \rfloor$. Plus tard, à la ligne 4 de la procédure de déchiffrage (**Algorithme 5.6**), la fonction $\mathrm{Compress}_q$ est utilisée pour décrypter à 1 si $v - \mathbf{s}^{\mathrm{T}}\mathbf{u}$ est proche de $\lceil q/2 \rfloor$ que de 0, et sinon de déchiffrer à 0.

Primitifs symétriques : la conception de Kyber fait intervenir l'usage d'une fonction pseudo-aléatoire *PRF* (pour *Pseudo Random Function*) : $\mathcal{B}^{32} \times \mathcal{B} \longrightarrow \mathcal{B}^*$ et d'une fonction de sortie extensible *XOF* (pour *eXtendable Output Function*) $\mathcal{B}^* \times \mathcal{B} \times \mathcal{B} \longrightarrow \mathcal{B}^*$. Kyber utilise également deux fonctions de hachage H: $\mathcal{B}^* \longrightarrow \mathcal{B}^{32}$ et G: $\mathcal{B}^* \longrightarrow \mathcal{B}^{32} \times \mathcal{B}^{32}$ et d'une fonction de dérivation de clé *KDF* (pour *Key Derivation Function*) : $\mathcal{B}^* \longrightarrow \mathcal{B}^*$.

Multiplication NTT et ordre de bit renversé : une manière très efficace d'effectuer une multiplication dans R_q est via la tant nommée Transformée Théorique des Nombres *NTT* (pour *Number Theoritic Transform*).

Pour notre nombre premier $q = 3329$, avec $q - 1 = 2^8.13$ le corps de base $\mathbb{Z}_q$ contient la 256-ième racine primitive de l'unité. Par opposition à la 512-ième racine primitive de l'unité qui n'y est pas incluse. Par conséquent, le polynôme $X^{256} + 1$ définissant R se factorise en 128 polynômes de degré 2 modulo q et la NTT d'un polynôme $f \in R_q$ est un vecteur de 128 polynômes de degré un. Des implémentations *in-place* – nous n'avons malheureusement pas trouvé une équivalence du terme en français. Un algorithme in-place est un algorithme qui transforme l'entrée en utilisant aucune structure de données auxiliaires. Plus généralement, in-place signifie que l'algorithme n'utilise pas d'espace supplémentaire pour manipuler l'entrée mais peut nécessiter un espace supplémentaire petit mais non constant pour son fonctionnement. Habituellement, cet espace est $O(\log n)$, bien que parfois tout ce qui se trouve dans $O(n)$ est autorisé. [5.02] – simples de la NTT sans réorganisation donnent à la sortie ces polynômes avec des bits en ordre inversé. Ceci explique

pourquoi la NTT est définie de cette manière. D'une manière plus concréte, soit $\zeta = 17$ la première 256-ième racine primitive de l'unité modulo q et $\{\zeta, \zeta^3, \dots, \zeta^{255}\}$ l'ensemble de toutes les 256-ième racines de l'unité. Le polynôme $X^{256} + 1$ peut alors s'écrire comme suit :

$$X^{256} + 1 = \prod_{i=0}^{127}(X^2 - \zeta^{2i+1}) = \prod_{i=0}^{127}(X^2 - \zeta^{2br_7(i)+1}) \quad \textbf{(}5.06\textbf{)}$$

où $br_7(i)$ pour $i = 0, \dots, 127$ est l'inversion de bits des 7 bits non signés de l'entier i. Ce dernier arrangement des facteurs est nécessaire pour la particularité des instructions *AVX* (pour *Advanced Vector Extensions*). Ensuite, la NTT de $f \in R_q$ est donnée par :

$$(f \bmod X^2 - \zeta^{2br_7(0)+1}, \dots, f \bmod X^2 - \zeta^{2br_7(127)+1}) \quad \textbf{(}5.07\textbf{)}$$

Ce vecteur de polynômes linéaires est par la suite sérialisé en un vecteur dans $\mathbb{Z}_q^{256}$ d'une manière canonique. De plus, afin de ne pas introduire de types de données supplémentaires et de faciliter les implémentations in-place de la NTT. Nous définissons NTT: $R_q \longrightarrow R_q$ est la bijection qui mappe $f \in R_q$ au polynôme avec le vecteur de coefficients susmentionné. D'où :

$$\text{NTT}(f) = \hat{f} = \hat{f}_0 + \hat{f}_1 X + \cdots + \hat{f}_{255} X^{255} \quad \textbf{(}5.08\textbf{)}$$

avec :

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j}\, \zeta^{(2br_7(i)+1)j} \quad \textbf{(}5.09\textbf{)}$$

$$\hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1}\, \zeta^{(2br_7(i)+1)j} \quad \textbf{(}5.10\textbf{)}$$

Nous voudrions souligner que même si nous écrivons $\hat{f}$ comme un polynôme dans R_q, il n'a pas de sens algébrique en tant que tel. La représentation algébrique naturelle de $\text{NTT}(f) = \hat{f}$ est du fait des 128 polynômes de degré 1 vus dans **(**5.07**)** en utilisant les définitions de $\hat{f}_i$ de **(**5.09**)** et **(**5.10**)**.

C'est-à-dire :

$$\mathrm{NTT}(f) = \hat{f} = \left(\hat{f}_0 + \hat{f}_1 X, \hat{f}_2 + \hat{f}_3 X, \ldots, \hat{f}_{254} + \hat{f}_{255} X\right) \quad \textbf{(5.11)}$$

L'utilisation de NTT et de son inverse NTT^{-1}, nous pouvons calculer le produit $f.g$ de deux éléments $f, g \in R_q$ d'une manière efficace genre $\mathrm{NTT}^{-1}(\mathrm{NTT}(f)\circ NTT(g))$ où $\mathrm{NTT}(f)\circ NTT(g) = \hat{f}\circ\hat{g} = \hat{h}$ désigne la multiplication basique (*basecase multiplication*, en anglais) constituée des 128 produits :

$$\hat{h}_{2i} + \hat{h}_{2i+1} X = \left(\hat{f}_{2i} + \hat{f}_{2i+1} X\right)\left(\hat{g}_{2i} + \hat{g}_{2i+1} X\right) \bmod X^2 - \zeta^{2br_7(i)+1} \quad \textbf{(5.12)}$$

de polynômes linéaires.

Lorsque nous appliquons NTT ou NTT^{-1} à un vecteur ou une matrice d'éléments de R_q, cela signifie que l'opération respective est appliquée à chaque entrée individuellement. Lorsque nous appliquons des matrices ou des vecteurs, cela signifie que nous effectuons une multiplication matricielle habituelle, mais que les produits individuels des entrées sont les multiplications de base ci-dessus.

Dans tout le document, nous écrirons NTT ou NTT^{-1} chaque fois que nous ferons référence aux fonctions concrètes telles que définies ci-dessus et utiliserons NTT de police normale chaque fois que nous nous référons à la technique générale.

Echantillonnage uniforme dans R_q : Kyber utilise une approche déterministe pour échantillonner les éléments dans R_q qui sont statistiquement proches d'une distribution uniformément aléatoire. Pour cet échantillonnage, nous utilisons une fonction $\mathrm{Parse}: \mathcal{B}^* \longrightarrow R_q$, qui reçoit comme entrée un flux d'octets $B = b_0, b_1, b_2, \ldots$ et évalue la représentation NTT $\hat{a} = \hat{a}_0 + \hat{a}_1 X + \cdots + \hat{a}_{n-1} X^{n-1} \in R_q$ de $a \in R_q$. Parse est décrite dans l'**Algorithme 5.1** (notons que $q = 3329$ dans la description qui va suivre).

Algorithme 5.1 : $\mathrm{Parse}: \mathcal{B}^* \longrightarrow R_q$

Entrée : Flux d'octets $B = b_0, b_1, b_2, \ldots;$
Sortie : représentation NTT $\hat{a} \in R_q$ de $a \in R_q;$
$i \coloneqq 0$

$j \coloneqq 0$
pour $j < n$ **faire**
    $d_1 \coloneqq b_i + 256 \cdot (b_{i+1} \bmod^+ 16)$
    $d_2 \coloneqq \lfloor b_{i+1}/16 \rfloor + 16 \cdot b_{i+2}$
    si $d_1 < q$ **alors**
        $\hat{a}_j \coloneqq d_1$
        $j \coloneqq j + 1$
    fin si
    si $d_2 < q$ **et** $j < n$ **alors**
        $\hat{a}_j \coloneqq d_2$
        $j \coloneqq j + 1$
    fin si
    $i \coloneqq i + 3$
fin pour
retourner $\hat{a}_0 + \hat{a}_1 X + \cdots + \hat{a}_{n-1} X^{n-1}$

L'intuition derrière la fonction Parse est que si le tableau d'octets d'entrée est statistiquement proche d'un tableau d'octets uniformément aléatoire, alors le polynôme de sortie est statistiquement proche d'un élément uniformément aléatoire de R_q. Il représente un polynôme uniformément aléatoire dans R_q car NTT est bijective et mappe ainsi des polynômes avec des coefficients uniformément aléatoires en polynômes avec, encore une fois, des coefficients uniformément aléatoires.

Echantillonnage à partir d'une distribution binomiale : Le bruit dans Kyber est échantillonné à partir d'une distribution binomiale centrée B_η pour $\eta = 2$ ou $\eta = 3$. B_η est définie comme suit :

$$\begin{aligned} &\text{Echantillonner } (a_1, \dots, a_n, b_1, \dots, b_n) \longleftarrow \{0,1\}^{2\eta} \\ &\text{Et retourner } \sum_{i=1}^{\eta} (a_i - b_i) \end{aligned} \qquad \textbf{(}5.13\textbf{)}$$

Quand nous écrivons qu'un polynôme $f \in R_q$ ou un vecteur de tels polynômes est échantillonné à partir de B_η, cela signifie que chaque coefficient est échantillonné à partir de B_η.

Pour la spécification de Kyber, nous devons définir comment un polynôme $f \in R_q$ est échantillonné selon B_η de manière déterministe à partir de 64η octets de sortie d'une fonction pseudo-aléatoire (nous fixons $n = 256$ dans cette description). Ceci est fait par la fonction CBD (pour *Centered Binomial Distribution*) définie comme décrit dans l'**Algorithme 5.2**.

Algorithme 5.2 : $\mathrm{CBD}_\eta: B^{64\eta} \longrightarrow R_q$

Entrée : tableau d'octets $B = (b_0, b_1, \dots, b_{64\eta-1}) \in \mathcal{B}^{64\eta}$
Sortie : polynôme $f \in R_q$
$(\beta_0, \dots, \beta_{512\eta-1}) \coloneqq \mathrm{BytesToBits}(B)$
pour i de 0 à 255 **faire**
    $a \coloneqq \sum_{j=0}^{n-1} \beta_{2i\eta+j}$
    $b \coloneqq \sum_{j=0}^{n-1} \beta_{2i\eta+n+j}$
    $f_i \coloneqq a - b$
fin pour
retourner $f_0 + f_1 X + \cdots + f_{255} X^{255}$

Encodage et décodage : Kyber doit sérialiser deux types de données en tableaux d'octets : les tableaux d'octets et les (vecteurs de) polynômes. Les tableaux d'octets sont sérialisés de manière triviale via l'identité, nous devons donc définir comment nous sérialisons et dé-sérialisons les polynômes. Dans l'**Algorithme 5.3**, une description de pseudocode de la fonction Decode_ℓ est donnée. Cette fonction dé-sérialise un tableau de 32ℓ octets en un polynôme $f = f_0 + f_1 X + \cdots + f_{255} X^{255}$ (nous fixons à nouveau $n = 256$ dans cette description) avec chaque coefficient f_i dans $\{0, \dots, 2^\ell - 1\}$. Nous définissons la fonction Encode_ℓ comme l'inverse de Decode_ℓ. Chaque fois que nous appliquons Encode_ℓ à un vecteur de polynômes, nous encodons chaque polynôme individuellement et concaténons les tableaux d'octets de sortie.

Algorithme 5.3 : $\mathrm{Decode}_\ell: \mathcal{B}^{32\ell} \longrightarrow R_q$

Entrée : tableau d'octets $B \in \mathcal{B}^{32\ell}$
Sortie : polynôme $f \in R_q$
$(\beta_0, \dots, \beta_{256\ell-1}) \coloneqq \mathrm{BytesToBits}(B)$
pour i de 0 à 255 **faire**
    $f_i \coloneqq \sum_{j=0}^{l-1} \beta_{i\ell+j^{2^j}}$
fin pour
retourner $f = f_0 + f_1 X + \cdots + f_{255} X^{255}$

5.2.1.2 Spécification de Kyber. CPAPKE

Kyber. CPAPKE est paramétré par les entiers n, k, q, η_1, η_2, d_u et d_v. Nous tenons à repréciser que tout au long du présent chapitre, $n = 256$ et $q = 3329$.

En utilisant les notations du précédent paragraphe 5.2.1.1 , la définition de la génération de clé, du cryptage et du décryptage du schéma de cryptage à clé publique Kyber. CPAPKE est donnée dans les algorithmes **Algorithme 5.4**, **Algorithme 5.5** et **Algorithme 5.6**. Une vue plus globale de ces algorithmes est mentionnée dans les commentaires.

Algorithme 5.4 : Kyber. CPAPKE. KeyGen() – *Génération de clé*

Sortie : Clé secrète $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Sortie : Clé publique $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

1 : $d \leftarrow \mathcal{B}^{32}$
2 : $(\rho, \sigma) \coloneqq G(d)$
3 : $N \coloneqq 0$
4 : **pour** i de 0 à $k-1$ **faire** ##Générer la matrice $\widehat{\mathbf{A}} \in R_q^{k \times k}$ dans le domaine NTT
5 : **pour** j de 0 à $k-1$ **faire**
6 : $\widehat{\mathbf{A}}[i][j] \coloneqq \text{Parse}(\text{XOF}(\rho, j, i))$
7 : **fin pour**
8 : **fin pour**
9 : **pour** i de 0 à $k-1$ **faire**
10 : $s[i] \coloneqq \text{CBD}_{\eta_1}\big(\text{PRF}(\sigma, N)\big)$ ##Echantillonner $\mathbf{s} \in R_q^k$ à partir de B_{η_1}
11 : $N \coloneqq N + 1$
12 : **fin pour**
13 : **pour** i de 0 à $k-1$ **faire** ##Echantillonner $\mathbf{e} \in R_q^k$ à partir de B_{η_1}
14 : $e[i] \coloneqq \text{CBD}_{\eta_1}\big(\text{PRF}(\sigma, N)\big)$
15 : $N \coloneqq N + 1$
16 : **fin pour**
17 : $\hat{\mathbf{s}} \coloneqq \text{NTT}(\mathbf{s})$
18 : $\hat{\mathbf{e}} \coloneqq \text{NTT}(\mathbf{e})$
19 : $\hat{\mathbf{t}} \coloneqq \widehat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
20 : $pk \coloneqq (\text{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q) || \rho)$ $pk \coloneqq \mathbf{As} + \mathbf{e}$
21 : $sk \coloneqq \text{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$ $sk \coloneqq s$
22 : **retourner** (pk, sk)

Algorithme 5.5 : Kyber. CPAPKE. Enc(pk, m, r) – *Cryptage*

Entrée : Clé publique $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Entrée : Message $m \in \mathcal{B}^{32}$

Entrée : Elément aléatoire r (pour random coin), $r \in \mathcal{B}^{32}$
Sortie : Texte chiffré $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

1 : $N \coloneqq 0$
2 : $\hat{\mathrm{t}} = \mathrm{Decode}_{12}(pk)$
3 : $\rho \coloneqq pk + 12 \cdot k \cdot n/8$
4 : **pour** $i = 0$ à $k - 1$ **faire** ##Générer la matrice $\widehat{\mathbf{A}} \in R_q^{k \times k}$ dans le domaine NTT
5 : **pour** $j = 0$ à $k - 1$ **faire**
6 : $\widehat{\mathrm{A}}^T[i][j] \coloneqq \mathrm{Parse}\big(\mathrm{XOF}(\rho, i, j)\big)$
7 : **fin pour**
8 : **fin pour**
9 : **pour** i de 0 à $k - 1$ **faire** ##Echantillonner $\mathrm{r} \in R_q^k$ à partir de B_{η_1}
10 : $\mathrm{r}[i] \coloneqq \mathrm{CBD}_{\eta_1}\big(\mathrm{PRF}(r, N)\big)$
11 : $N \coloneqq N + 1$
12 : **fin pour**
13 : **pour** i de 0 à $k - 1$ **faire** ##Echantillonner $\mathrm{e}_1 \in R_q^k$ à partir de B_{η_2}
14 : $\mathrm{e}_1[i] \coloneqq \mathrm{CBD}_{\eta_2}\big(\mathrm{PRF}(r, N)\big)$
15 : $N \coloneqq N + 1$
16 : **fin pour**
17 : $e_2 \coloneqq \mathrm{CBD}_{\eta_2}\big(\mathrm{PRF}(r, N)\big)$ ##Echantillonner $\mathrm{e}_2 \in R_q$ à partir de B_{η_2}
18 : $\hat{\mathbf{r}} \coloneqq \mathrm{NTT}(\mathbf{r})$
19 : $\mathbf{u} \coloneqq \mathrm{NTT}^{-1}\big(\widehat{\mathbf{A}}^T \circ \hat{\mathbf{r}}\big) + \mathrm{e}_1$ ##$\mathbf{u} \coloneqq \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
20 : $\mathrm{v} \coloneqq \mathrm{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + \mathrm{e}_2 + \mathrm{Decompress}_q(\mathrm{Decode}_1(m), 1)$ ##$v \coloneqq \mathbf{t}^{\mathrm{T}}\mathbf{r} + e_2 + \mathrm{Decompress}_q(m, 1)$
21 : $c_1 \coloneqq \mathrm{Encode}_{d_u}\Big(\mathrm{Compress}_q(\mathbf{u}, d_u)\Big)$
22 : $c_2 \coloneqq \mathrm{Encode}_{d_v}\Big(\mathrm{Compress}_q(v, d_v)\Big)$
23 : **retourner** $c = (c_1 || c_2)$ $c \coloneqq \Big(\mathrm{Compress}_q(\mathbf{u}, d_u), \mathrm{Compress}_q(v, d_v)\Big)$

Algorithme 5.6 : Kyber. CPAPKE. Dec(sk, c) – *Décryptage*

Entrée : Clé secrète $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Entrée : Texte chiffré $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Sortie : Message $m \in \mathcal{B}^{32}$

1 : $\mathbf{u} \coloneqq \text{Decompress}_q\big(\text{Decode}_{d_u}(c), d_u\big)$

2 : $v \coloneqq \text{Decompress}_q\big(\text{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v\big)$

3 : $\hat{\mathbf{s}} \coloneqq \text{Decode}_{12}(sk)$

4 : $m \coloneqq \text{Encode}_1(\text{Compress}_q(v - \text{NTT}^{-1}\big(\hat{\mathbf{s}}^T \circ \text{NTT}(\mathbf{u})\big), 1))$ ##$m \coloneqq \text{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

5 : **retourner** m

5.2.1.3 Spécification de Kyber. CCAKEM

Algorithme 5.7 : Kyber. CCAKEM. KeyGen() – *Génération de clé*

Sortie : Clé publique $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

Sortie : Clé secrète $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$

1 : $z \leftarrow \mathcal{B}^{32}$

2 : $(pk, sk') \coloneqq$ Kyber. CPAPKE. KenyGen()

3 : $sk \coloneqq (sk'||pk||H(pk)||z)$

4 : **retourner** (pk, sk)

Algorithme 5.8 : Kyber. CCAKEM. Enc() – *Cryptage*

Entrée : Clé publique $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

Sortie : Texte chiffré $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$

Sortie : Clé partagée $K \in \mathcal{B}^*$

1 : $m \leftarrow \mathcal{B}^{32}$

2 : $m \leftarrow \text{H}(m)$ ##Ne pas envoyer la sortie du système RNG (pour Random Number Generator)

3 : $(\overline{K}, r) \coloneqq \text{G}(m||\text{H}(pk))$

4 : $c \coloneqq$ Kyber. CPAPKE. Enc(pk, m, r)

5 : $K = KDF(\overline{K}||H(c))$

6 : **retourner** (c,K)

Algorithme 5.9 : Kyber. CCAKEM. Dec(c, sk) – *Décryptage*

Entrée : Texte chiffré $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$

Entrée : Clé secrète $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$

1 : $pk \coloneqq sk + 12 \cdot k \cdot n/8$

2 : $h \coloneqq s + 24 \cdot k \cdot n/8 + 32 \in \mathcal{B}^{32}$

3 : $z \coloneqq s + 24 \cdot k \cdot n/8 + 64$

$m' \coloneqq \text{Kyber.CPAPKE.Dec}\big(s, (\mathbf{u}, v)\big)$
$(\overline{K}', r') \coloneqq \text{G}(m'||h)$
$c' \coloneqq \text{Kyber.CPAPKE.Enc}(pk, m', r')$
si $c = c'$ alors
    retourner $K \coloneqq \text{KDF}(K'||\text{H}(c))$
sinon
    retourner $K \coloneqq \text{KDF}(z||\text{H}(c))$
fin si
retourner K

Le KEM Kyber.CCAKEM sécurisé IND-CCA2 est construit à partir du schéma de cryptage à clé publique sécurisé IND-CPA décrit dans la sous-section précédente via une transformation Fujisaki-Okamoto (*FO*) légèrement modifiée. Dans les algorithmes **Algorithme 5.7**, **Algorithme 5.8** et **Algorithme 5.9**, la génération de clé, l'encapsulation et la décapsulation de Kyber.CCAKEM seront détaillés.

5.2.1.4 Jeu de paramètres de Kyber

Trois (03) ensembles de paramètres pour Kyber ont été définis, citons Kyber512, Kyber768 et Kyber1024. Les paramètres sont répertoriés dans le **Tableau 5.01**. Notez que le tableau répertorie également le paramètre dérivé, qui est la probabilité que la décapsulation d'un texte chiffré Kyber.CCAKEM valide échoue. Les paramètres ont été obtenus via l'approche suivante :

- n est défini à 256 car le but est d'encapsuler des clés avec 256 bits d'entropie (c'est-à-dire utiliser une taille de texte en clair de 256 bits dans Kyber.CPAPKE.Enc). Des valeurs plus petites de n nécessiteraient de coder plusieurs bits de clé en un seul coefficient polynomial, ce qui nécessite des niveaux de bruit plus faibles et par conséquent diminue la sécurité. Des valeurs plus élevées de n réduiraient la capacité de mettre à l'échelle facilement la sécurité via le paramètre k.

Tableau 5.01 : *Jeux de paramètres de Kyber.*

	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber512	256	2	3329	3	2	(10,4)	2^{-139}
Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11,5)	2^{-174}

- q a été choisi comme étant un nombre premier petit satisfaisant $n|(q-1)$; ceci est nécessaire pour activer la multiplication rapide basée sur NTT. Cette propriété est valable pour deux nombres premiers petits, à savoir 257 et 769. Cependant, pour ces nombres premiers, nous ne serions pas en mesure d'obtenir une probabilité de défaillance négligeable requise pour la sécurité CCA (pour *Chosen Ciphertext Attack*), nous avons donc choisi le nombre plus grand qui s'ensuit, c'est-à-dire $q = 3329$.

- k est choisi pour fixer la dimension du réseau comme un multiple de n ; le changement de k est le principal mécanisme de Kyber pour dimensionner la sécurité (et par conséquent l'efficacité) à des niveaux différents.

- Les paramètres restants η_1, η_2, d_u et d_v ont été choisis pour équilibrer la sécurité, la taille du texte chiffré et la probabilité de défaillance. Notez que les trois ensembles de paramètres atteignent une probabilité de défaillance $< 2^{-128}$ avec une certaine marge. Des détails plus poussés sur le sujet seront fournis un peu plus loin.

- Le paramètre η_1 définit le bruit de **s** et **e** dans l'**Algorithme 5.4** et de **r** dans l'**Algorithme 5.5**. Le paramètre η_2 définit le bruit de $\mathbf{e_1}$ et e_2 dans l'**Algorithme 5.5**. Remarquons que l'on a défini $\eta_1 > \eta_2$ dans le paramètre de Kyber512. Ce fait sera discuté, encore une fois, un peu plus loin dans nos écrits.

La probabilité d'échec δ est calculée à l'aide du script Python Kyber.py de notre bibliographie [5.03]. Pour le contexte théorique de ce script, voir notre bibliographie [5.04], *Théorème 5.01*.

Instanciation de PRF, XOF, H, G *et* KDF *:* Ce qui manque encore pour compléter la spécification de Kyber, c'est l'instanciation des primitives symétriques. Nous instancions toutes ces primitives avec des fonctions de la norme *FIPS*-202 (pour *Federal Information Processing Standards Publication*) comme suit [5.05] :

- XOF est instanciée avec SHAKE-128 ;

- H est instanciée avec SHA3-256, (SHA pour *Secure Hash Algorithm*) ;

- G est instanciée avec SHA3-512 ;

- PRF(s, b) est instanciée avec SHAKE-256$(s||b)$;

- KDF est instanciée avec SHAKE-256.

5.2.1.5 Justifications additionnelles de conception

Certains points de justifications de conception de Kyber ont été déjà mentionnés dans notre précédent chapitre (chapitre 4). Quelques-uns restants, et faisant intervenir des notions plus avancées seront énumérés. A savoir :

- *Sécurité active*. L'un des avantages des KEM à sécurité passive est qu'ils peuvent accepter une probabilité de défaillance plus élevée (ce qui permet soit d'augmenter la sécurité en augmentant le bruit, soit en diminuant la taille de la clé publique et du texte chiffré). L'autre avantage est qu'ils ne nécessitent pas de transformée CCA, et sont donc livrés avec une décapsulation plus rapide. Malgré ces avantages, Kyber est défini comme un KEM sécurisé IND-CCA2 uniquement. Pour de nombreuses applications comme le cryptage à clé publique (via une construction KEM-DEM, DEM pour *Data Encapsulation Mechanism*) ou dans l'échange de clés authentifiées, la sécurité active est obligatoire. Cependant, également dans les cas d'utilisation (comme l'échange de clés dans TLS, pour *Transport Layer Security*) qui ne nécessitent pas à proprement parler de sécurité active, l'utilisation d'un KEM activement sécurisé présente des avantages. Plus particulièrement, il permet la mise en cache (intentionnelle ou accidentelle) de clés éphémères. De plus, la transformation CCA de Kyber protège contre certains bogues dans les implémentations. Plus précisément, les systèmes à sécurité passive ne remarqueront pas si le partenaire de communication utilise un « mauvais » bruit, par exemple, un bruit entièrement nul. Un tel bogue dans l'encapsulation de Kyber sera immédiatement attrapé par l'étape de re-chiffrement lors de la décapsulation. En conclusion, nous pensons que les coûts liés au fait de vouloir garantir la sécurité CCA ne sont pas suffisamment importants pour justifier sa sauvegarde et rendre le système moins robuste.

- *Différentes valeurs de bruit η_1 et η_2*. Notons que dans l'**Algorithme 5.5**, il y a un bruit implicite supplémentaire créé via $\mathrm{Compress}_q$. Cela a pour effet d'ajouter du bruit (déterministe) au texte chiffré, ce qui peut être interprété comme une augmentation du bruit des polynômes d'erreur $\mathbf{e_1}$ et e_2. Si la probabilité d'erreur de décryptage est suffisamment faible, il est alors logique d'augmenter également le bruit des autres termes secrets (c'est-à-dire **s**, **e**, **r**) pour qu'il soit à un niveau similaire à $\mathbf{e_1}$ plus le bruit déterministe. Nous utilisons cette idée (exclusivement) pour le jeu de paramètres Kyber512. Se fier au bruit d'arrondi de $\mathrm{Compress}_q$ pour ajouter une erreur s'apparente à l'hypothèse LWR (pour *Learning With Rounding* pour rappel). Mais contrairement aux

schémas (Ring/Module)-LWR, où la sécurité repose entièrement sur le bruit qui est généré de manière déterministe par l'arrondi, notre dépendance au bruit déterministe est beaucoup plus faible. Premièrement, cela n'ajoute que 6 bits de difficulté du noyau-SVP, et deuxièmement, nous ajoutons du bruit et des arrondis, ce qui a vraisemblablement moins de structure algébrique qu'un simple arrondi. En bref, sans l'hypothèse LWR, notre jeu de paramètres pour Kyber512 a 112 bits de difficulté du noyau-SVP - plus précisément, les clés publiques sont protégées avec 118 bits et les textes chiffrés avec 112 ; avec une version faible de l'hypothèse LWR, il a une sécurité de 118 bits partout.

- *Hachages additionnels.* Dans la transformation CCA, nous hachons la (hachage de la) clé publique pk pour obtenir la pré-clé $\overline{K}$ et les éléments aléatoires r (voir la ligne 3 de l'**Algorithme 5.8**), et nous hachons le (hachage du) texte chiffré pour obtenir la clé finale K. Ces hachages ne sont pas nécessaires pour la réduction de la sécurité, mais ils ajoutent de la robustesse. Plus précisément, la sortie finale de la clé partagée par Kyber. CCAKEM dépend de la vue complète des messages échangés (clé publique et texte chiffré), ce qui signifie que le KEM est contributif et sûr à utiliser dans les échanges de clés authentifiés sans hachage supplémentaire du contexte. Le hachage de pk également en pièces aléatoires r ajoute une protection contre une certaine classe d'attaques multi-cibles qui tentent d'utiliser des échecs de protocole.

- *Choix de primitives symétriques.* Dans la conception de Kyber, nous avons besoin d'une XOF, de deux fonctions de hachage, d'une fonction pseudo-aléatoire et d'un KDF. Nous avons décidé de nous appuyer sur une seule primitive sous-jacente pour toutes ces fonctions. Cela aide à réduire la taille du code dans les plates-formes embarquées et (pour un choix prudent) réduit les craintes que Kyber puisse être attaqué en exploitant les faiblesses d'une primitive symétrique parmi plusieurs. Il n'y a que relativement peu de fonctions de sortie extensibles décrites dans la littérature. Les plus connues, qui ont également inventé le terme XOF, sont les fonctions SHAKE basées sur Keccak et standardisées en FIPS-202. Cette norme décrit également de manière pratique les fonctions de hachage avec les longueurs de sortie dont nous avons besoin ; en outre, SHAKE est conçu pour fonctionner également comme PRF.

- *Prise en charge des API de hachage non incrémentielles*. Dans la ligne 3 de l'**Algorithme 5.8** nous introduisons $\mathrm{H}(pk)$ (au lieu de pk) dans G et à la ligne 5 nous introduisons $\mathrm{H}(c)$ (au lieu de c) dans H. Utiliser $\mathrm{H}(pk)$ dans l'appel à G permet une petite accélération de la décapsulation.

Cependant, il y a une autre raison pour laquelle nous hachons d'abord pk et c, à savoir que cela simplifie la mise en œuvre de Kyber avec une API de hachage non incrémentielle. Si Kyber est implémenté dans un environnement qui propose déjà une bibliothèque pour le hachage, mais n'offre que des appels de la forme $h = \mathrm{H}(m)$, alors produire un hachage de la forme $h = \mathrm{H}(m_1||m_2)$ nécessiterait d'abord de copier m_1 et m_2 dans une zone de mémoire consécutive. Cela nécessiterait des copies inutiles et, plus important encore, un espace de pile supplémentaire.

- *Valeur renvoyée pour l'échec de la décapsulation*. Traditionnellement, la transformation FO retourne $\perp$ (c'est-à-dire un symbole d'échec spécial) lorsque la décapsulation échoue. Nous utilisons une variante qui définit à la place la clé partagée résultante à une valeur pseudo-aléatoire calculée comme le hachage d'un secret z et le texte chiffré c. Cette variante de la transformation FO s'est avérée sûre. En pratique, cela présente l'avantage que les implémentations de la décapsulation de Kyber sont sûres à utiliser même si les protocoles de niveau supérieur ne parviennent pas à vérifier la valeur de retour. En conséquence de cette approche de rejet implicite, nos implémentations de décapsulation retournent toujours 0.

5.2.2 Analyses de performance

Dans cette section, nous examinons les aspects de mise en œuvre de Kyber et rapportons les résultats de performance de deux implémentations : l'implémentation de référence ANSI C (ANSI pour *American National Standards Institute*) et une implémentation optimisée à l'aide d'instructions vectorielles AVX2.

Vue d'ensemble de la performance Kyber. Grâce à la multiplication efficace basées sur NTT et à l'échantillonnage de **A** dans le domaine NTT, les performances de Kyber sont largement déterminées par les performances des primitives symétriques. Ceci est illustré, par exemple, par le fait que l'implémentation optimisée AVX2 de la variante des années 90 de Kyber est presque deux fois plus rapide que Kyber avec des primitives symétriques basées sur Keccak. Cette différence sera encore plus grande sur les systèmes avec SHA-256 à accélération matérielle. Elle est également illustrée par le fait que pour des implémentations optimisées, la décapsulation est plus rapide que l'encapsulation, même si la première contient une ré-encapsulation nécessaire à la transformée FO. La raison principale en est que la décapsulation n'a pas besoin de calculer $\mathrm{H}(pk)$, ce qui compense largement la réencapsulation supplémentaire dominée par NTT.

Le fait que le hachage (séquentiel) soit très lent par rapport à l'arithmétique polynomiale sur les processeurs Intel récents signifie également que les hachages supplémentaires discutés dans la section 5.2.1.5 ont un impact très significatif sur les performances : ils représentent près de la moitié des cycles d'encapsulation de notre mise en œuvre optimisée en AVX2 sur les processeurs Intel. Comme également mentionné dans la section 5.2.1.5 , le hachage de la clé publique et du texte chiffré n'est pas nécessaire pour la sécurité CCA, mais est plutôt effectué pour rendre la fonction plus robuste et directement utilisable dans les applications qui nécessitent la dépendance de la clé partagée sur toute la transcription. La justification en est que, étant donné que les opérations de base comprenant Kyber sont extrêmement rapides, nous pouvons nous permettre de payer une pénalité de temps et de rendre la version par défaut de Kyber aussi robuste et résiliente que possible. Néanmoins, compte tenu du fait que ce hachage a un effet profond sur les performances absolues du schéma, c'est une direction de recherche future intéressante pour voir s'il existe des applications qui nécessitent effectivement le hachage de l'intégralité du texte chiffré et/ou de la clé publique. S'il suffit de hacher une petite partie d'entre eux, le temps de fonctionnement de Kyber peut être réduit de moitié (sur de nombreux processeurs) sans sacrifier la robustesse.

5.2.2.1 Considérations de mise en œuvre et compromis

Implémentation du NTT. De nombreux compromis sont possibles lors de la mise en œuvre de la transformation de la théorie des nombres. Les plus importants se situent entre la taille du code (qui devient principalement pertinente sur les processeurs embarqués) et la vitesse. Les deux implémentations de Kyber étudiées sont un NTT en sens normal (de l'ordre normal à bit inversé) et un NTT en sens inverse (de bit inversé à l'ordre normal). De plus, les deux implémentations utilisent des tables de puissances pré-calculées de ζ. Ce qui est particulièrement intéressant à propos de l'utilisation du NTT sur des plates-formes embarquées, c'est que la multiplication de deux éléments de R_q peut être calculée sans stockage temporaire supplémentaire. Ce qui est particulièrement intéressant à propos de l'utilisation du NTT sur de gros processeurs, c'est qu'il est très efficacement traitable sous forme de vecteurs. Depuis 2013, l'approche la plus efficace pour calculer le NTT sur les processeurs Intel 64 bits consistait à représenter les coefficients sous forme de valeurs à virgule flottante double précision. Dans notre implémentation de Kyber optimisée pour AVX2, nous montrons que l'optimisation minutieuse du NTT à l'aide d'instructions d'entiers AVX2 se traduit par de bien meilleures performances. Plus précisément, sur les processeurs Intel Haswell, un NTT en sens normal dans Kyber ne prend qu'environ 320 cycles ; un NTT en sens inverse ne prend

qu'environ 290 cycles.

Keccak. Le deuxième composant critique pour la vitesse à l'intérieur de Kyber sont les primitives symétriques, c'est-à-dire SHA3-256, SHA3-512, SHAKE-128 et SHAKE-256, toutes basées sur la permutation Keccak. SHA3 a la réputation de ne pas être la fonction de hachage la plus rapide des logiciels. Dans une certaine mesure, cela est compensé par le fait que la plupart des appels à Keccak sont parallèles et donc très efficacement vectorisées. Notre implémentation AVX2 utilise ce fait. En outre, ARM (pour *Advanced RISC Machines*) a récemment annoncé que les futurs processeurs ARMv8 auront un support matériel pour SHA3, il y a donc de fortes chances qu'au moins sur certaines architectures, les performances logicielles de SHA3 ne soient plus un problème à l'avenir.

AES et SHA-2. Pour illustrer les performances que Kyber peut atteindre avec des primitives symétriques accélérées par le matériel, nous incluons la variante des années 90 utilisant AES, SHA256 et SHA512 au lieu de primitives symétriques basées sur Keccak. Cette variante n'est intéressante que si au moins AES est accéléré dans le matériel, car les implémentations logicielles à temps constant (nécessaires pour l'utilisation d'AES comme PRF) ne sont pas plus rapides que Keccak parallèle.

Hardware-RNGs pour la génération de clés. Pendant la génération de clé, la génération de **s** et **e** est effectuée à l'aide de SHAKE-256 (et AES dans la variante des années 90). Cependant, ce n'est pas requise. Le choix du RNG (pour *Random Number Generator*) lors de la génération de clé est une décision locale que tout utilisateur et plateforme peuvent prendre indépendamment. En particulier sur les plates-formes avec un matériel AES rapide, on peut adapter le PRF basé sur AES de la variante des années 90 également pour le Kyber basé sur Keccak. Nous avons envisagé de l'utiliser dans notre implémentation AVX2, mais l'utilisation de cette optimisation signifie que les vecteurs de test ne correspondraient pas entre nos deux implémentations. Ce n'est pas un problème dans les déploiements réels, où randombytes ne sont pas déterministes.

Mise en cache des clés éphémères. Les applications qui sont encore plus conscientes du temps de génération de clé peuvent décider de mettre en cache les clés éphémères pendant un certain temps. Ceci est rendu possible par le fait que Kyber est sécurisé IND-CCA2.

Compromis entre la taille et la vitesse de la clé secrète. Il est possible d'utiliser différents compromis

entre la taille de la clé secrète et la vitesse de décapsulation. Si la taille de la clé secrète est critique, il est bien sûr possible de ne pas stocker H(pk) et également de ne pas stocker la clé publique dans le cadre de la clé secrète, mais de la recalculer lors de la décapsulation. De plus, ne pas conserver la clé secrète dans le domaine NTT permet de compresser chaque coefficient à seulement 3 bits, ce qui signifie qu'une clé secrète Kyber768 peut être compressée en seulement 288 octets. Cela présente l'avantage supplémentaire de rendre la récupération de clé via des attaques de démarrage à froid, un peu plus difficile. Enfin, comme tout le caractère aléatoire de la génération de clé est généré à partir de deux graines (*graine aléatoire* ou *random seed* an anglais, pour rappel) de 32 octets, il est également possible de stocker uniquement ces graines et de réexécuter la génération de clé pendant la décapsulation. Lorsqu'on opte pour de telles clés secrètes de 32 octets, l'étape de ré-encapsulation de décapsulation peut sauver l'expansion de la matrice **A**, si elle est déjà étendue (sous forme transposée) dans l'étape de génération de clé.

Dans l'autre sens, si la taille de la clé secrète n'a pas beaucoup d'importance et que la vitesse de décapsulation est critique, on peut décider de stocker la matrice étendue **A** dans le cadre de la clé secrète et d'éviter le recalcul à partir de la graine ρ pendant la partie de ré-encapsulation de décapsulation.

Les deux implémentations utilisent le format de clé secrète décrit dans l'**Algorithme 5.7**, c'est-à-dire avec des polynômes dans le domaine NTT, y compris la clé publique et H(pk), mais sans inclure la matrice **A**.

Format de stockage local des clés publiques statiques. Un utilisateur qui encapsule fréquemment des messages sur la même clé publique peut accélérer l'encapsulation en stockant localement une clé publique étendue contenant les matrices **A** et H(pk). Cela permet d'économiser le coût de l'expansion de la matrice **A** à partir de la graine ρ et le coût du hachage pk dans chaque encapsulation.

5.2.2.2 Performances de Kyber sur les processeurs Intel Haswell

Le **Tableau 5.02** présente les résultats de performance d'Intel Haswell de l'implémentation de référence et d'une implémentation optimisée AVX2 de Kyber et de la variante 90s de Kyber ainsi que les tailles des clés et des textes chiffrés. Tous les benchmarks ont été obtenus sur un cœur d'un processeur Intel Core i7-4770K (Haswell) cadencé à 3492 MHz (comme indiqué par /proc/cpuinfo)

avec TurboBoost et l'hyperthreading désactivés. La machine utilisée a 32 Go de RAM (pour *Random Access Memory*) et exécute Debian GNU/Linux (GNU de *GNU's Not Unix*) avec la version 4.19.0 du noyau Linux. Les deux implémentations ont été compilées avec gcc version 8.3.0. Tous les décomptes de cycles indiqués sont la médiane des décomptes de cycles de 10000 exécutions de la fonction respective. Les implémentations ne sont pas optimisées pour l'utilisation de la mémoire, mais en général, Kyber n'a que des besoins en mémoire très modestes.

Tableau 5.02 : *Performances de Kyber sur les processeurs Intel Haswell.*

Kyber512					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	1632 (ou 32)	gen :	122684	gen :	33856
pk :	800	enc :	154524	enc :	45200
ct :	768	dec :	187960 (ou ≈ 288912)	dec :	34572 (ou ≈ 59088)
Kyber512 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	1632 (ou 32)	gen :	213156	gen :	21880
pk :	800	enc :	249084	enc :	28592
ct :	768	dec :	277612 (ou ≈ 405268)	dec :	20980 (ou ≈ 38752)
Kyber768					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	2400 (ou 32)	gen :	199408	gen :	52732
pk :	1184	enc :	235260	enc :	67624
ct :	1088	dec :	274900 (ou ≈ 425492)	dec :	53156 (ou ≈ 82220)
Kyber768 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	2400 (ou 32)	gen :	389760	gen :	30460
pk :	1184	enc :	432764	enc :	40140
ct :	1088	dec :	473984 (ou ≈ 671864)	dec :	301108 (ou ≈ 52512)
Kyber1024					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	3168 (ou 32)	gen :	307148	gen :	73544
pk :	1568	enc :	346648	enc :	97324
ct :	1568	dec :	396584 (ou ≈ 617848)	dec :	79128 (ou ≈ 115332)
Kyber1024 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	3168 (ou 32)	gen :	636380	gen :	43212
pk :	1568	enc :	672644	enc :	56566
ct :	1568	dec :	724144 (ou ≈ 1009448)	dec :	44328 (ou ≈ 71180)

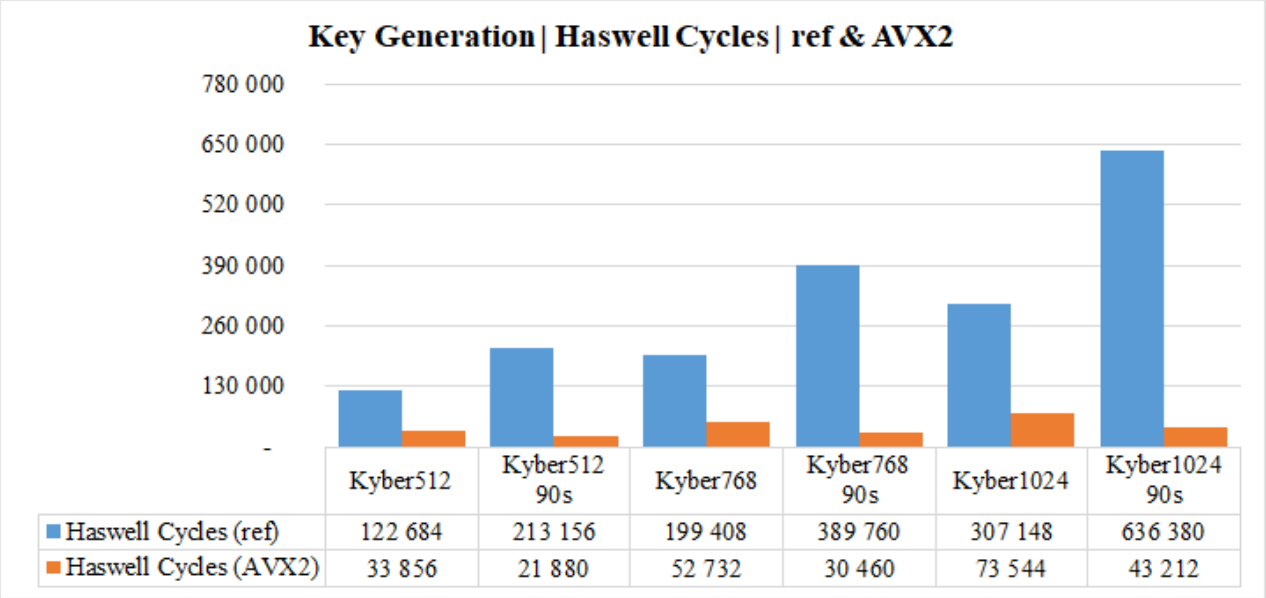


Figure 5.01 : *Nombre de cycles requis pour l'étape de génération de clé (Key generation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres de Kyber.*

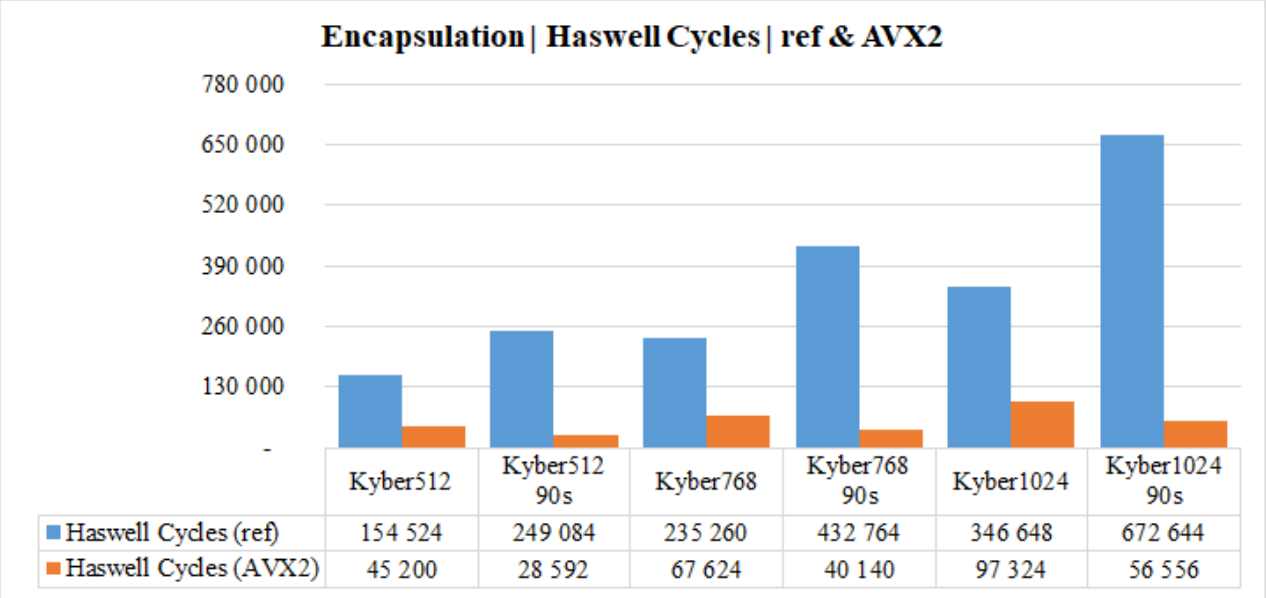
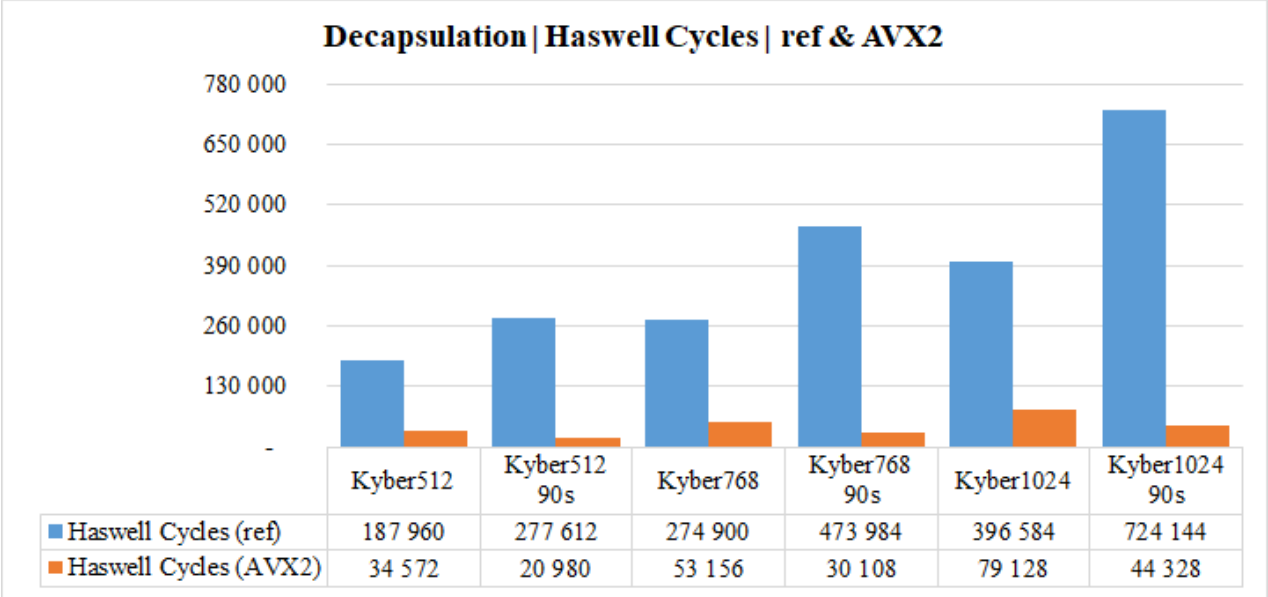


Figure 5.02 : *Nombre de cycles requis pour l'étape d'encapsulation (Encapsulation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres de Kyber.*

En addition, *ref* fait référence à l'implémentation de référence C, *AVX2* à l'implémentation utilisant des instructions vectorielles AVX2 ; *sk* signifie clé secrète, *pk* pour clé publique et *ct* pour texte chiffré. Entre parenthèses se trouvent des valeurs approximatives lors de l'inclusion de la génération

de clé dans la décapsulation pour éviter d'avoir à stocker des clés secrètes développées. Dans ce scénario, nous stockons uniquement la graine initiale d dans la ligne 1 de l'**Algorithme 5.4**. Les nombres de cycles approximatifs pour ce scénario sont calculés comme la somme des nombres de cycles pour la décapsulation standard et la génération de clé moins le nombre de cycles nécessaires pour générer la matrice **A** de la graine publique ρ. Notez qu'il s'agit d'une estimation très prudente ; les mises en œuvre réelles de l'approche peuvent également économiser, par exemple, l'échantillonnage des 32 octets de caractère aléatoire. Voir également la discussion sur les « compromis entre la taille et la vitesse de la clé secrète » dans note paragraphe 5.2.2.1 .



	Kyber512	Kyber512 90s	Kyber768	Kyber768 90s	Kyber1024	Kyber1024 90s
Haswell Cycles (ref)	187 960	277 612	274 900	473 984	396 584	724 144
Haswell Cycles (AVX2)	34 572	20 980	53 156	30 108	79 128	44 328

Figure 5.03 : *Nombre de cycles requis pour l'étape de décapsulation (Decapsulation), en langage de référence C ou en utilisant la technologie AVX2, pour les différents jeux de paramètres de Kyber.*

Dans les figures **Figure 5.01**, **Figure 5.02** et **Figure 5.03**, nous avons tenu à exprimer d'une manière visuelle les résultats obtenus dans le **Tableau 5.02**.

5.2.2.3 Performance of Kyber on ARM Cortex-M4 CPUs

Le **Tableau 5.03** rapporte le nombre de cycles et l'utilisation de la RAM d'une implémentation C (propre) et d'une implémentation optimisée (m4) de Kyber sur un ARM Cortex-M4. Tous les benchmarks sont obtenus en utilisant le framework pqm4 sur une carte Discovery STM32F407. Nous ne rapportons pas les benchmarks de la version des années 90 de Kyber sur le Cortex-M4, car pqm4 n'inclut pas encore d'implémentation en temps constant d'AES.

Tableau 5.03 : *Performances de Kyber sur les processeurs ARM Cortex-M4.*

Kyber512				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	655595	6020	463068	2844
enc :	865256	8668	561518	2484
dec :	961648	9444	519237	2508
Kyber768				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1087897	10052	756224	3292
enc :	1373744	13212	915676	2980
dec :	1491214	14308	853001	3004
Kyber1024				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1696314	15180	1213303	3804
enc :	2057522	18844	1407769	3492
dec :	2199958	20420	1326409	3516

Dans les figures **Figure 5.04**, **Figure 5.05** et **Figure 5.06**, nous avons tenu à exprimer d'une manière visuelle les résultats obtenu dans le **Tableau 5.03**.

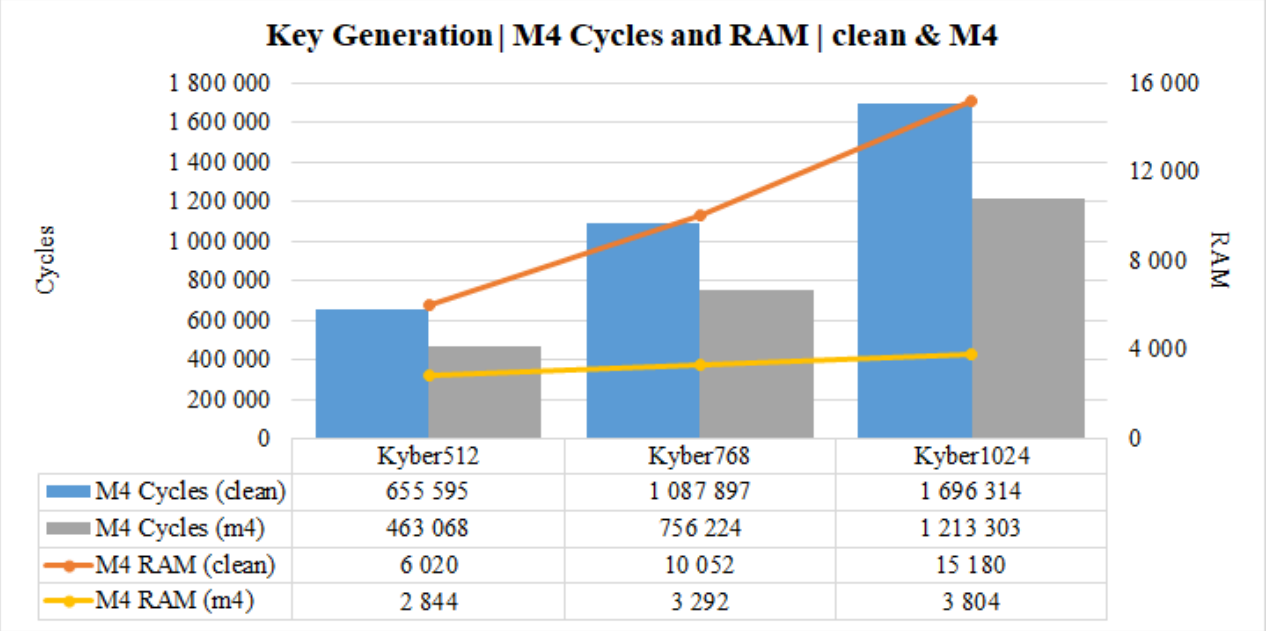
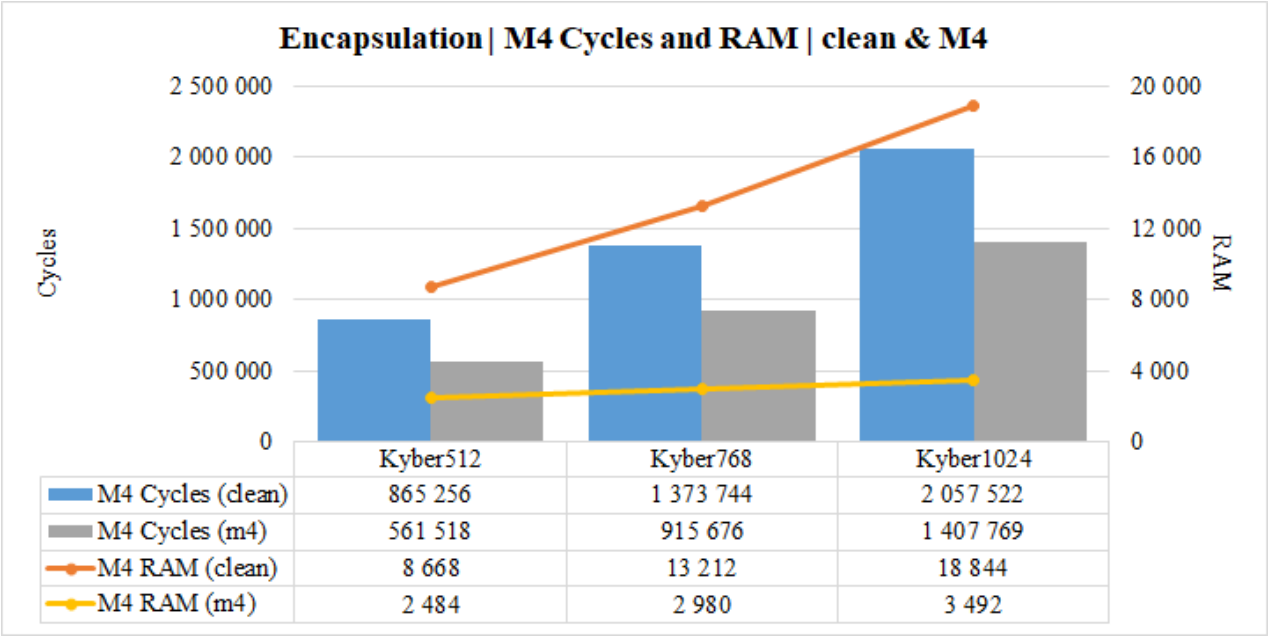
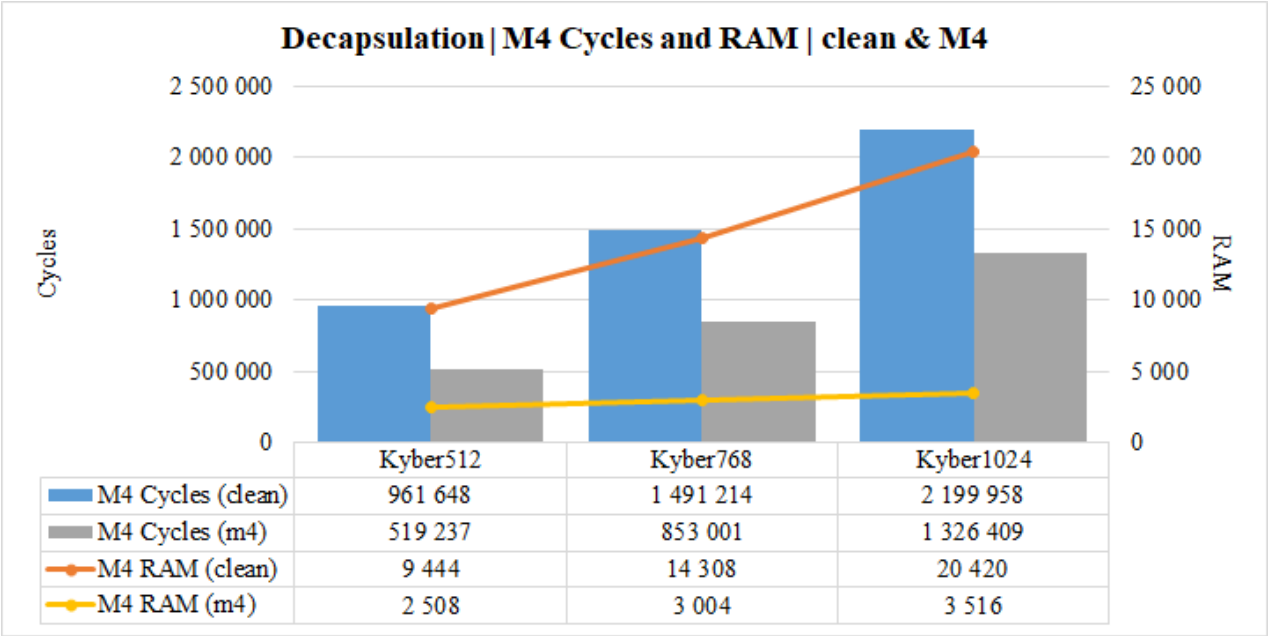


Figure 5.04 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape de génération de clé (Key generation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.*



	Kyber512	Kyber768	Kyber1024
M4 Cycles (clean)	865 256	1 373 744	2 057 522
M4 Cycles (m4)	561 518	915 676	1 407 769
M4 RAM (clean)	8 668	13 212	18 844
M4 RAM (m4)	2 484	2 980	3 492

Figure 5.05 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape d'encapsulation (Encapsulation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.*



	Kyber512	Kyber768	Kyber1024
M4 Cycles (clean)	961 648	1 491 214	2 199 958
M4 Cycles (m4)	519 237	853 001	1 326 409
M4 RAM (clean)	9 444	14 308	20 420
M4 RAM (m4)	2 508	3 004	3 516

Figure 5.06 : *Nombre de cycles et quantité de RAMs (en Octets) requis pour l'étape de décapsulation (Decapsulation), en langage de référence C ou en implémentation optimisée Cortex-M4, pour les différents jeux de paramètres de Kyber.*

5.2.3 Niveau de sécurité attendu

Kyber.CCAKEM (ou dans son format court, Kyber) est un mécanisme d'encapsulation de clé sécurisé IND-CCA2. La définition de ce dernier a été donnée dans le précédent chapitre de notre ouvrage (chapitre 4).

5.2.3.1 Justification des estimations de sécurité

Les estimations du niveau de sécurité pour les trois ensembles de paramètres différents de Kyber sont basées sur les estimations de coût des attaques contre le problème sous-jacent, le Module Learning With Errors (MLWE).

Pour appuyer le présent raisonnement, nous donnerons dans ce qui suit deux réductions de MLWE : une réduction étroite par respect au *Random Oracle Model* (*ROM*) dans le *Théorème 5.02* et une réduction non-étroite par respect au *Quantique Random Oracle Model* (*QROM*) dans le *Théorème 5.03*. Avec ces réductions à portée de main, il reste deux voies d'attaque qui briseraient Kyber sans résoudre le problème sous-jacent du MLWE, à savoir :

(1) Briser l'une des hypothèses de réductions, en attaquant en particulier les primitives symétriques utilisées dans Kyber ; ou
(2) En exploitant le fait que la réduction QROM soit non-étroite.

Pour le point (1), nous renvoyons nos lecteurs à la sous-section 5.4 de notre bibliographie [5.01]. La discussion de (2), quant à elle, nécessite d'examiner deux questions distinctes, à savoir :

- Le fait que la probabilité d'échec de décryptage de Kyber.CPAPKE est non étroite (d'une manière quadratique), et ;

- Le fait qu'entre l'attaquant MLWE et l'attaquant quantique contre Kyber, l'avantage ne soit étroite (d'une manière quadratique).

Encore une fois, pour les deux points cités plus haut, nous renvoyons nos lecteurs dans la sous-section 5.5 de notre bibliographie [5.01]. Y sont discutées, les attaques quantiques exploitant les échecs de décryptage, et dans la présentation de la réduction QROM non étroite, sont expliqués pourquoi le fait que l'avantage entre les attaques quantiques contre MLWE et les attaques quantiques contre Kyber est non étroit et peu susceptible d'avoir une importance dans la pratique.

Plus précisément, il y est montré, comment éliminer cette non-étroitesse si l'on admet l'hypothèse raisonnable, mais non standard, que les textes chiffrés de Kyber. CPAPKE sont pseudo-aléatoires, même si tout le caractère aléatoire est généré de manière pseudo-aléatoire à partir d'un hachage du message chiffré.

5.2.3.2 Hypothèse de sécurité

Le problème difficile sous-jacent à la sécurité du présent cryptosystème est Module-LWE. Il consiste à distinguer des échantillons uniformes $(\mathbf{a}_i, b_i) \longleftarrow R_q^k \times R_q$ des échantillons $(\mathbf{a}_i, b_i) \longleftarrow R_q^k \times R_q$ où $\mathbf{a}_i \in R_q^k$ est uniforme et $b_i = \mathbf{a}_i^T s + e_i$ avec $s \longleftarrow B_\eta^k$ identique pour tous les échantillons et $e_i \longleftarrow B_\eta$ renouvelé pour chaque échantillon. Plus précisément, pour un algorithme A, nous définissons :

$$\mathbf{Adv}^{\mathrm{mlwe}}_{m,k,\eta}(\mathrm{A}) = \left|\Pr\left[b' = 1 : \begin{matrix}\mathbf{A} \longleftarrow R_q^{m\times k}; (\mathbf{s},\mathbf{e}) \longleftarrow \beta_\eta^k \times \beta_\eta^m; \mathbf{b} = \mathbf{As} + \mathbf{e}; \\ b' \longleftarrow \mathrm{A}(\mathbf{A},\mathbf{b});\end{matrix}\right]\right| - \left|\Pr\left[b' = 1 : \mathbf{A} \longleftarrow R_q^{m\times k}; \mathbf{b} \longleftarrow R_q^m; b' \longleftarrow \mathrm{A}(\mathbf{A},\mathbf{b});\right]\right| \quad (5.14)$$

a. *Réduction étroite du MLWE dans le ROM*

Nous notons d'abord que Kyber. CPAPKE est étroitement sécurisé IND-CPA sous l'hypothèse de difficulté du Module-LWE.

Théorème 5.01 :

Supposons que XOF et G soient des oracles aléatoires. Pour tout adversaire A, il existe des adversaires B et C avec à peu près le même temps d'exécution que celui de A tel que :

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Kyber.CPAPKE}}(\mathrm{A}) \leq 2\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathrm{B}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{PRF}}(\mathrm{C})$$

La preuve de ce théorème est facilement obtenue en notant que, sous l'hypothèse MLWE, la clé publique et le texte chiffré sont pseudo-aléatoires.

Kyber. CCAKEM est obtenu via une transformée Fujisaki-Okamoto légèrement modifiée appliquée à Kyber. CPAPKE. La déclaration de sécurité concrète suivante prouve la sécurité IND-CCA2 de Kyber. CCAKEM lorsque les fonctions de hachage G et H sont modélisées comme des oracles

aléatoires.

Théorème 5.02 :

Supposons que XOF, H et G soient des oracles aléatoires. Pour tout adversaire classique A qui fait au plus q_{RO} de nombreuses requêtes aux oracles aléatoires XOF, H et G, il existe des adversaires B et C à peu près de même durée de fonctionnement que celui de A tels que :

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathrm{Kyber.CPAPKE}}(\mathrm{A}) \leq 2\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathrm{B}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{PRF}}(\mathrm{C}) + 4q_{\mathrm{RO}}\delta$$

Notons que la limite de sécurité est étroite. Le terme additif négligeable $4 \cdot q_{\mathrm{RO}}$ provient de la probabilité d'échec de déchiffrement δ de Kyber. CPAPKE.

b. *Réduction non étroite de MLWE dans le QROM*

Quant à la sécurité dans le QROM, [5.04] et [5.06] ont prouvé que Kyber. CCAKEM est sécurisé INDCCA2 dans le QROM, à condition que Kyber. CPAPKE soit sécurisé IND-CPA. Une réduction légèrement plus étroite peut être obtenue en exigeant que le schéma de base Kyber. CPAPKE soit pseudo-aléatoire. Le pseudo-aléatoire exige que, pour chaque message m, un texte chiffré (généré aléatoirement) $(c_1, c_2) \leftarrow \mathrm{Kyber.CPAPKE.Enc}(pk, m)$ ne puisse être distingué par calcul d'un texte chiffré aléatoire de la forme $(\mathrm{Compress}_q(\mathbf{u}, du), \mathrm{Compress}_q(v; dv))$, pour $(\mathbf{u}, v)$ uniforme. (Nous avons également besoin de la propriété de « disjonction statistique » qui est trivialement remplie pour Kyber. CPAPKE.) La preuve de la sécurité IND-CPA de Kyber. CPAPKE montre en effet que Kyber. CPAPKE est étroitement pseudo-aléatoire sous le Module-LWE hypothèse de difficulté.

Théorème 5.03 :

Supposons que XOF, H et G soient des oracles aléatoires. Pour tout adversaire quantique A qui effectue au plus q_{RO} requêtes vers des oracles quantiques aléatoires XOF, H et G, il existe des adversaires quantiques B et C ayant à peu près le même temps d'exécution que celui de A tels que :

$$\mathbf{Adv}^{\mathrm{cca}}_{\mathrm{Kyber.CCAKEM}}(\mathrm{A}) \leq 4q_{\mathrm{RO}} \cdot \sqrt{\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathrm{B})} + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{PRF}}(\mathrm{C}) + 8q_{RO}^2\delta$$

Malheureusement, la limite de sécurité ci-dessus n'est pas étroite et ne peut donc servir

qu'indication asymptotique de la sécurité CCA de Kyber.CCAKEM dans le QROM.

Réduction étroite sous une hypothèse non standard. Nous pouvons utiliser [5.04] et [5.06] pour dériver une limite de sécurité étroite dans le QROM à partir d'une hypothèse de sécurité non standard, à savoir qu'une version déterministe de Kyber.CPAPKE, appelée DKyber.CPAPKE, est pseudo-aléatoire dans le QROM. Le Kyber.CPAPKE déterministe est défini comme Kyber.CPAPKE, mais l'élément aléatoire r utilisé dans le cryptage est dérivés de manière déterministe du message m comme $r \coloneqq \mathrm{G}(m)$. Le caractère pseudo-aléatoire pour le chiffrement déterministe indique qu'un chiffrement (c_1, c_2) d'un message choisi au hasard ne peut pas être distingué par calcul d'un texte chiffré aléatoire $(\mathrm{Compress}_q(\mathbf{u}, du), \mathrm{Compress}_q(v; dv))$, pour $(\mathbf{u}, v)$ uniforme. Dans le ROM classique, pseudo-aléatoire de DKyber.CPAPKE est étroitement équivalent à MLWE mais dans le QROM la réduction n'est pas étroite (et est la raison du terme $q_{\mathrm{RO}} \cdot \sqrt{\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathrm{B})}$ dans le *Théorème 5.03*. Concrètement, on obtient la majoration suivante :

$$\begin{aligned}\mathbf{Adv}^{\mathrm{cca}}_{\mathrm{Kyber.CCAKEM}}(\mathrm{A}) \\ &\leq 2\mathbf{Adv}^{\mathrm{mlwe}}_{k+1,k,\eta}(\mathrm{B}) + \mathbf{Adv}^{\mathrm{pr}}_{\mathrm{DKyber.CPAPKE}}(\mathrm{C}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{PRF}}(\mathrm{D}) \\ &+ 8q_{RO}^2\delta\end{aligned} \qquad \textbf{(}5.15\textbf{)}$$

Nous remarquons que nous n'avons connaissance d'aucune attaque quantique sur Kyber.CPAPKE déterministe qui fonctionne mieux que de briser le problème MLWE.

5.2.3.3 Estimation de niveau de sécurité

Le **Tableau 5.04** répertorie les niveaux de sécurité selon la définition de la section 4.A.5 de l'appel à propositions du NIST [5.07] pour les différents ensembles de paramètres de Kyber. Nos affirmations sont basées sur les estimations de coût des attaques les plus connues contre le problème MLWE sous-jacent à Kyber, comme détaillé dans la sous-section 5.1 de notre bibliographie [5.01]. Plus précisément, y est listée, la difficulté classique et quantique du noyau-SVP et nous les utilisons pour dériver les niveaux de sécurité.

Tous les jeux de paramètres de Kyber ont une certaine probabilité d'échec de décryptage. Ces échecs sont un problème de sécurité (voir Section 5.5 de [5.01]) et donc les probabilités avec lesquelles ils

se produisent doivent être faibles. Mais comme dans le ROM classique, la probabilité d'échec de décryptage est théorique de l'information, nous ne voyons pas la nécessité de la diminuer avec le paramètre de sécurité. En particulier, l'échec de décryptage pour nos ensembles de paramètres de niveau 3 et 5 est inférieur à 2^{-160}, ce qui signifie que si 280 instances de Kyber étaient exécutées toutes les secondes à partir de maintenant jusqu'à ce que notre soleil devienne une naine blanche, les chances sont toujours fortement favorables à ce qu'il n'y ait jamais d'échec de décryptage. Nous excluons donc ces attaques de nos affirmations concernant les estimations de sécurité du NIST.

Tableau 5.04 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber.*

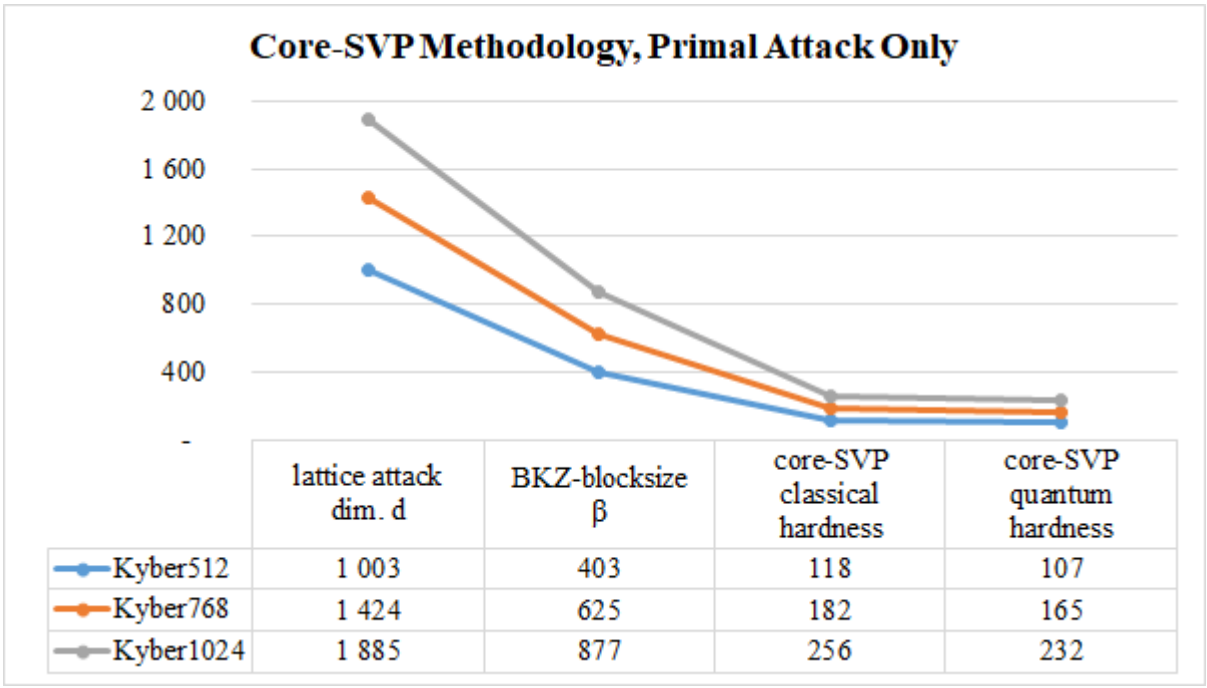
	Kyber512	Kyber768	Kyber1024
Niveau de sécurité NIST	1	3	5

Méthodologie noyau-SVP, attaque primitive uniquement			
Attaque lattice dim. d	1003	1424	1885
BKZ-blocksize β (BKZ pour Block Korkine Zolotarev)	403	625	877
Noyau-SVP difficulté classique	118	182	256
Noyau-SVP difficulté quantique	107	165	232

Estimation affinée pour les attaques classiques			
Attaque lattice dim. d	1025	1467	1918
BKZ-blocksize β	413	637	894
Tamisage dimension $\beta' = \beta - d_{4f}$	375	586	829
$\log_2$(gates)	151.5	215.1	287.3
$\log_2$(memory in bits)	93.8	138.5	189.7

L'impact du bruit déterministe causé par Compress_q *sur Kyber512*. Chaque coefficient de $\mathbf{e_1}$ (et $\mathbf{e_2}$) dans l'**Algorithme 5.5** se comporte comme une distribution binomiale avec le paramètre $\eta_2 = 2$, avec une variance $\eta_2/2 = 1$. Le paramètre $d_u = 10$ implique que la fonction Compress_q mappe les éléments modulo q à un ensemble de taille 2^{10} où la différence entre tous les pairs d'éléments de ce dudit ensemble est de 3 ou 4. Cela implique que l'erreur ajoutée par la fonction Compress_q pour chaque coefficient est soit uniforme sur$\{-1,0,1\}$, $\{-1,0,1,2\}$ ou $\{-2,-1,0,1\}$. Elle a donc une variance au moins aussi grande que la distribution uniforme sur l'ensemble $\{-1,0,1\}$, soit $2/3$. Cela rend la variance totale de chaque coefficient de $\mathbf{e_1}$ plus l'erreur déterministe au moins $1 + 2/3 =$

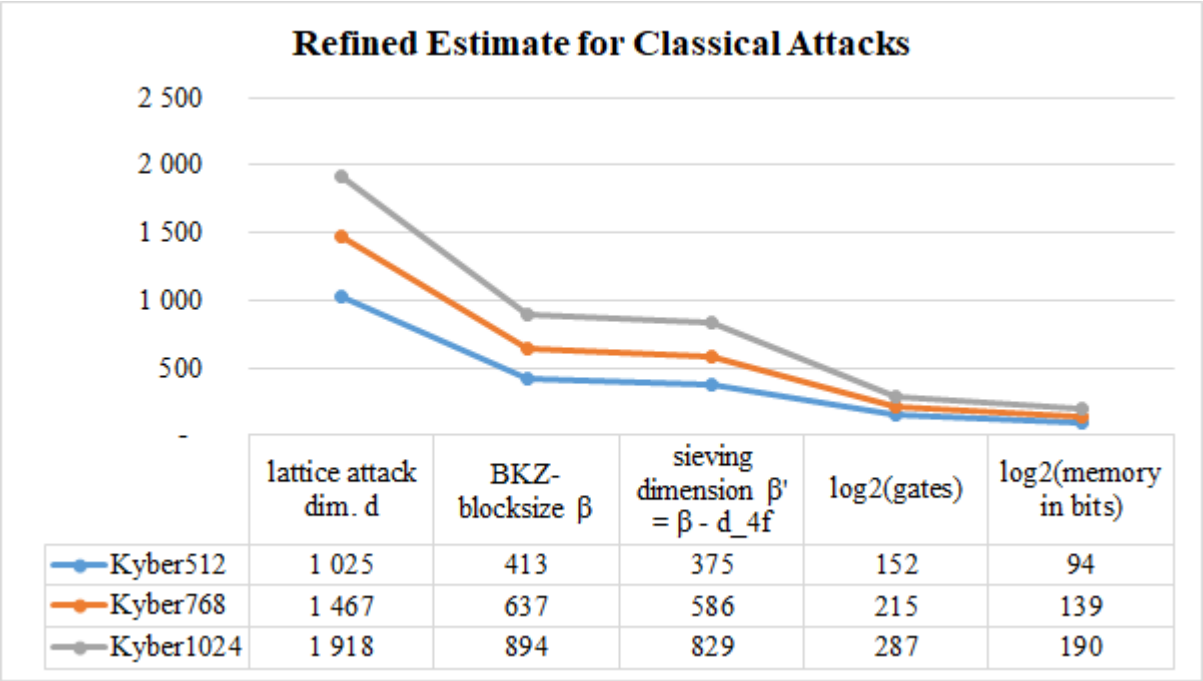
$5/3$. Les autres termes secrets et bruit ont des distributions binomiales avec le paramètre $\eta_1 = 3$ pour une variance de $\eta_1/2 = 3/2 < 5/3$. En tenant compte des erreurs ajoutées par Compress_q, nous calculons donc la difficulté de Kyber512 en supposant que la variance de chaque coefficient secret/erreur est $3/2$.



	lattice attack dim. d	BKZ-blocksize β	core-SVP classical hardness	core-SVP quantum hardness
Kyber512	1 003	403	118	107
Kyber768	1 424	625	182	165
Kyber1024	1 885	877	256	232

Figure 5.07 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber. Méthodologie noyau-SVP, attaque primitive uniquement.*

L'impact de MAXDEPTH. Les accélérations quantiques les plus connues pour l'algorithme de tamisage, qui sont plus détaillées dans l'analyse des coûts (se référer à la sous-section 5.1.1 de [5.01]), ne sont que légèrement affectées par la limitation de la profondeur d'un circuit quantique, car elles utilisent la recherche de Grover sur des ensembles de petite taille (comparée à la recherche dans tout l'espace de clés d'AES). Pour que les estimations de la difficulté de l'opération du noyau-SVP correspondent au coût de la porte quantique de la rupture d'AES aux niveaux de sécurité respectifs, un ordinateur quantique devrait prendre en charge une profondeur maximale de 70 à 80. Lors de la limitation de la profondeur maximale à des valeurs plus petites, ou lors de l'examen d'attaques classiques, les estimations de la difficulté noyau-SVP sont plus petites que le nombre de portes pour les attaques contre AES. Une étude récente sur le coût concret du tamisage suggère que les accélérations quantiques de ces algorithmes sont tenues, indépendamment de la valeur de MAXDEPTH.



	lattice attack dim. d	BKZ-blocksize β	sieving dimension β' = β - d_4f	log2(gates)	log2(memory in bits)
Kyber512	1 025	413	375	152	94
Kyber768	1 467	637	586	215	139
Kyber1024	1 918	894	829	287	190

Figure 5.08 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber. Estimation affinée pour les attaques classiques.*

Dans les figures **Figure 5.07** et **Figure 5.08**, nous avons tenu à exprimer d'une manière visuelle les résultats obtenu dans le **Tableau 5.04**.

5.2.3.4 Propriétés de sécurité additionnelles

a. *Sécurité iterative*

Kyber dispose d'une procédure de génération de clé très efficace (voir notre paragraphe 5.2.2) et est donc particulièrement bien adaptée aux applications qui utilisent des générations de clés fréquentes pour obtenir le secret de transmission.

b. *Attaques par canal auxiliaire*

Attaque temporelle. Tous les types d'implémentations de Kyber que nous avons présentés jusqu'ici n'utilisent pas de branches dépendantes de la clé secrète ou encore de recherches dans une table pré-élaborée. Cela signifie que les implémentations typiques de Kyber sont exemptes des deux sources les plus notoires de fuite de synchronisation. Les multiplicateurs à temps non constant comme l'instruction *UMULL* (pour *Unsigned Long MULtiply*) sur les processeurs ARM Cortex-M3, qui multiplient deux entiers 32 bits pour obtenir un résultat 64 bits, sont une autre source possible de

fuite de synchronisation. Cependant, les multiplications dans Kyber n'ont que des entrées de 16 bits, et la plupart des multiplicateurs à temps non constant montrent une variation de synchronisation uniquement pour les entrées de tailles conséquentes. Par exemple, sur les processeurs ARM Cortex-M3, le moyen évident d'implémenter des multiplications dans Kyber consiste à utiliser l'instruction *MUL* (pour *MULtiply*) à temps constant, qui multiplie deux entiers de 32 bits, mais ne renvoie que les 32 bits inférieurs du résultat. Ce qui reste une source de perte de temps, ce sont des réductions modulaires, qui sont parfois implémentées via des instructions conditionnelles. Cependant, les fuites de synchronisation dans les réductions modulaires sont facilement évitées en utilisant des réductions *Montgomery* et *Barrett* (se référer à notre bibliographie [5.01] pour des détails sur ces sujets) comme illustré dans nos implémentations de référence et AVX2.

Nous notons que la variante des années 90 n'est vraiment intéressante à utiliser que si le support matériel AES est disponible. Si la prise en charge matérielle n'est pas disponible, les implémentations basées sur des tables d'AES sont réputées être sujettes des fuites d'informations via la synchronisation de la cache. Dans notre implémentation de référence C de la variante des années 90, nous utilisons une implémentation en tranches de bits à temps constant d'AES, qui est basée sur le code de *BearSSL* (se référer à notre bibliographie [5.01] pour des détails sur le sujet).

Attaque différentielle. Il est probable que toute implémentation de Kyber sans protection dédiée contre les attaques différentielles basées sur la consommation énergétique (*DPA* pour *Differential Power Analysis*) mise en jeu ou encore celle relative aux rayonnements électromagnétiques (*EM*) sera vulnérable à de telles attaques. Cela est vrai pour pratiquement toute implémentation d'un schéma cryptographique qui utilise des clés à usage étendu (non éphémères). Les scénarios de déploiement de Kyber dans lesquels un attaquant est supposé apte à lancer une telle attaque nécessitent des implémentations spécialement protégées, généralement masquées. *Oder, Schneider, Pöppelmann* et *Güneysu* ont présenté (se référer à notre bibliographie [5.01]) une telle implémentation masquée du décryptage Ring-LWE avec une transformée CCA très similaire à celle utilisée dans Kyber. L'implémentation cible les microcontrôleurs Cortex-M4F ; la conclusion du travail est que la protection de l'étape de décryptage (décapsulation) contre le DPA de premier ordre entraîne une surcharge d'environ un facteur de 5,5. Les techniques présentées dans cet article s'appliquent également à Kyber et nous nous attendons à ce que le coût de protection de Kyber contre les attaques différentielles ait le même ordre de grandeur.

Attaque par modèle. Les protections contre les attaques différentielles n'aident pas si un attaquant est capable de récupérer même les clés éphémères à partir d'une seule trace de consommation énergétique ou d'EM. *Primas*, *Pessl* et *Mangard* (se référer à notre bibliographie [5.01]) ont présenté une telle attaque à trace unique contre une implémentation de Ring-LWE sur un microcontrôleur Cortex-M4F. Le modèle de l'attaquant dans cette attaque est assez solide : il s'agit du paramètre typique des attaques par modèle, qui suppose qu'un attaquant est capable de générer des traces de modèle sur des entrées connues sur un appareil avec une fuite très similaire à l'appareil cible réel. Toujours dans notre bibliographie, les auteurs ont utilisé le même dispositif pour générer les traces cibles, similaires à celle de l'attaque menée. L'attaque a été facilitée par le fait que l'implémentation attaquée utilisait des réductions modulaires à temps variable. La conclusion en est que l'une des premières mesures pour renforcer une mise en œuvre contre les attaques SPA (pour *Simple Power Analisys,* une version cadette de la DPA vue un peu plus haut) est d'assurer une exécution et un flux de contrôle constants. C'est le cas de toutes les implémentations de Kyber. L'attaque précédemment énoncée ne s'appliquerait donc certainement pas directement aux implémentations de Kyber, mais des recherches supplémentaires sont nécessaires pour déterminer si les implémentations en temps constant de Kyber (et d'autres schémas basés sur les lattices) succombent aux attaques par modèles, et de ce que serait le coût des contre-mesures appropriées.

c. *Attaques multi-cibles*

Notre analyse de sécurité ne fait aucune déclaration formelle sur les limites de sécurité dans le cadre multi-cible. Cependant, lors de la conception de Kyber, deux décisions ont été prises visant à améliorer la sécurité contre les attaquants ciblant plusieurs utilisateurs :

- Nous adoptons l'approche « contre toute autorité » de régénération de la matrice **A** pour chaque clé publique comme dans NewHope. Cela protège contre un attaquant qui tente de casser plusieurs clés en arrivant à casser uniquement une seule clé.

- Dans la transformation CCA (voir **Algorithme 5.8**), nous hachons la clé publique pour obtenir la pré-clé $\overline{K}$ et les éléments aléatoires r. Rendre les éléments aléatoires r dépendantes de la clé publique protège contre les attaques de pré-calcul qui tentent de casser une clé parmi d'autres. Pour plus de détails, voir la sous-section 5.5 de [5.01].

d. *Résilience face aux erreurs de manipulation*

La première ligne de défense contre les erreurs de manipulation est la décision de rendre la sécurité IND-CCA2 non facultative. Comme indiqué dans notre précédent chapitre, chapitre 4, auquel se sont rajoutés quelques points dans notre paragraphe 5.2.1.5 , il aurait été possible d'obtenir des clés publiques et des textes chiffrés légèrement plus courts, et une décapsulation plus rapide, dans une variante sécurisée CPA de Kyber. L'utilisation de la sécurité IND-CCA2 par défaut permet d'utiliser Kyber avec des clés statiques en toute sécurité et, par conséquent, de réutiliser les clés éphémères pendant un certain temps. Ce qui n'est pas sûr, c'est le fait de réutiliser le même élément aléatoire dans l'encapsulation, mais cet élément aléatoire n'est pas non plus exposé à l'extérieur par l'API. La transformée CCA a un second effet en termes de robustesse : elle protège contre une mise en œuvre interrompue de l'échantillonnage de bruit. Un aspect plutôt particulier de la cryptographie basée sur LWE est qu'elle passera des tests fonctionnels typiques même si un partenaire de communication n'ajoute aucun bruit.

Une ligne de défense supplémentaire contre les erreurs de manipulation consiste à hacher la clé publique pour obtenir la « pré-clé » $\overline{K}$ et ainsi s'assurer que le KEM soit contributif. Seuls quelques protocoles exigent qu'un KEM soit contributif et ces protocoles peuvent toujours transformer un KEM non contributif en un KEM contributif en hachant la clé publique pour obtenir la clé finale. Faire de ce hachage une partie de la conception de KEM dans Kyber garantit que le protocole n’affichera pas de problème si les personnes en charge de l’implémention omettent le hachage.

Une déclaration similaire est valable pour le hachage supplémentaire du texte chiffré pour obtenir la clé finale. Plusieurs protocoles doivent garantir que la clé dépend de la vue complète des messages de protocole échangés. Le fait que le hachage de la vue complète du protocole (clé publique et texte chiffré) pour obtenir la clé finale fasse partie intégrante de la définition du KEM rend inutile de prendre en charge ces hachages sur la couche de protocole supérieure.

5.3 Cryptosystème type signature, CRYSTAL Dilithium

5.3.1 Introduction

5.3.1.1 Génération de clé, signature et vérification

Un avant-gout du présent crypto système type signature a déjà été introduit dans notre précédent

chapitre, chapitre 4. Ci-dessous, nous allons illustrer en termes d'algorithme les étapes constituant le cryptosystème CRYSTAL Dilithium.

Algorithme 5.10 : *Modèle de notre schéma de signature sans compression de clé publique*

Gen

1 : $\mathbf{A} \longleftarrow R_q^{k \times l}$
2 : $(\mathrm{s}_1, \mathrm{s}_2) \longleftarrow S_\eta^l \times S_\eta^k$
3 : $\mathbf{t} \coloneqq \mathbf{A}\mathrm{s}_1 + \mathrm{s}_2$
4 : **retourner** $\big((pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathrm{s}_1, \mathrm{s}_2)\big)$

Sign(sk, M)

5 : $\mathbf{z} \coloneqq \perp$
6 : **pour** $\mathbf{z} = \perp$ **faire**
7 : $\quad \mathbf{y} \longleftarrow S_{\gamma_1 - 1}^{\ell}$
8 : $\quad \mathbf{w_1} \coloneqq \mathrm{HighBits}(\mathbf{Ay}, 2\gamma_2)$
9 : $\quad c \in B_\tau \coloneqq \mathbf{H}(M || \mathbf{w_1})$
10 : $\quad \mathbf{z} \coloneqq \mathbf{y} + c\mathrm{s}_1$
11 : $\quad$ **si** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ ou $\|\mathrm{LowBits}(\mathbf{Ay} - c\mathrm{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta$ **alors** $\mathbf{z} \coloneqq \perp$
12 : **retourner** $\sigma = (\mathbf{z}, c)$

Verify$\big(pk, M, \sigma = (\mathbf{z}, c)\big)$

13 : $\mathbf{w_1'} \coloneqq \mathrm{HighBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)$
14 : **si retourner** $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ **et** $[\![c = \mathbf{H}(M || w_1')]\!]$

Rappelons encore une fois que les interprétations de cet algorithme ont déjà été abordées dans notre chapitre précédent, chapitre 4, toujours. Nous ne faisons qu'illustrer les mêmes affirmations, mais cette fois-ci en langage algorithmique.

5.3.1.2 Information additionnelle

Le modèle de base de l'**Algorithme 5.10** est plutôt inefficace, tel qu'il est. L'inefficacité la plus flagrante en est que la clé publique consiste en une matrice de polynômes kl, qui pourrait avoir une représentation assez large. Le correctif est simplement d'avoir **A** généré à partir d'une graine en utilisant SHAKE-128, et c'est une technique standard. La clé publique est donc $(\rho, \mathbf{t})$ et sa taille est dominée par **t**. La nouveauté de Dilithium est que la taille de représentation binaire de **t** est réduite d'un facteur légèrement plus grand plus que 2 au prix de l'augmentation de la signature de moins de 100 octets.

La principale observation pour obtenir ce compromis est que lorsque le vérificateur calcule $\mathbf{w_1'}$ à la ligne 13, les bits de poids fort de $\mathbf{Az} - c\mathbf{t}$ ne dépendent pas trop des bits de poids faible de $\mathbf{t}$ car $\mathbf{t}$ est multiplié par un polynôme de très faible poids c. Dans notre algorithme, certains bits de poids faible de $\mathbf{t}$ ne sont pas inclus dans la clé publique, et donc le vérificateur ne peut pas toujours calculer correctement les bits de poids fort de $\mathbf{Az} - c\mathbf{t}$. Pour compenser cela, le signataire inclut des « indices » dans le cadre de la signature, qui sont essentiellement les portées provoquées par l'ajout du produit de c avec les bits de poids faible manquants de $\mathbf{t}$. Avec cet indice, le vérificateur est capable de calculer correctement $\mathbf{w_1'}$.

De plus, nous proposons une option pour rendre notre algorithme déterministe en utilisant la technique standard d'ajout d'une graine à la clé secrète et en utilisant cette graine avec le message pour produire le caractère aléatoire $\mathbf{y}$ dans la ligne 7. Notre bibliographe énonce des résultats récents où il est montré que moins l'adversaire voit des signatures différentes pour les mêmes messages, plus la réduction est étroite dans le modèle d'oracle quantique aléatoire entre le schéma de signature et les hypothèses de dureté sous-jacentes. Bien qu'il ne soit pas clair s'il existe une attaque quantique améliorée pour les signatures aléatoires, nous suggérons la version déterministe comme option par défaut, sauf dans les scénarios où un adversaire peut monter des attaques par canal auxiliaire qui exploitent le déterminisme. Notre schéma complet de l'**Algorithme 5.13** utilise également des optimisations de base telles que le pré-hachage du message M afin de ne pas le refaire à chaque tentative de signature.

a. *Points importants de la phase d'implémentation*

La principale opération algébrique effectuée dans l'algorithme est une multiplication d'une matrice $\mathbf{A}$, dont les éléments sont des polynômes dans $\mathbb{Z}_q[X]/(X^{256}+1)$, par un vecteur $\mathbf{y}$ de tels polynômes. Dans le paramétrage NIST Niveau 3, $\mathbf{A}$ est une matrice 6×5 et se compose donc de 30 polynômes. Ainsi, la multiplication $\mathbf{Ay}$ implique 30 multiplications polynomiales. Comme dans de nombreux algorithmes à base de lattices et faisant intervenir des opérations sur des anneaux polynomiaux, nous avons choisi notre anneau afin que l'opération de multiplication ait une implémentation très efficace via la transformation théorique des nombres (NTT), qui n'est qu'une version de FFT le corps fini $\mathbb{Z}_q$ plutôt que sur les nombres complexes. Pour activer l'algorithme NTT « full-splitting », nous devons choisir un premier q afin que le groupe $\mathbb{Z}_q$ ait un élément d'ordre $2n = 512$; ou de manière équivalente $q \equiv 1 \pmod{512}$. Si r est un tel élément, alors $X^{256} + 1 =$

$(X - r)(X - r^3) \dots (X - r^{511})$ et donc on peut représenter de manière équivalente tout polynôme $a \in \mathbb{Z}_q[X]/(X^{256} + 1)$ sous sa forme CRT (Chinese Remainder Theorem pour rappel) comme $\left(a(r), a(r^3), \dots, a(r^{511})\right)$. L'avantage de cette représentation est que le produit de deux polynômes est en coordonnées (coordinate-wise en anglais). Par conséquent, les parties les plus chères de la multiplication polynomiale sont les transformations $a \longrightarrow \hat{a}$ et l'inverse $\hat{a} \longrightarrow a$ - ce sont les opérations NTT et NTT inverse.

L'avantage mentionné ci-dessus d'utiliser la représentation NTT est encore plus prononcé du fait que lors de la multiplication **Ay**, il faut calculer beaucoup moins de NTT que la dimension de la matrice. Puisque la matrice est uniformément aléatoire, on peut la générer (et la stocker) déjà sous forme NTT. Faire la multiplication de cette matrice par un vecteur $\mathbf{y} \in R_q^5$ implique alors de seulement faire 5 calculs NTT pour convertir **y** en représentation NTT, puis de faire des multiplications par point (pointwise multiplications en anglais) pour obtenir **Ay** sous forme NTT, suivies de 6 multiplications NTT inverses.

L'autre opération importante qui prend du temps est l'expansion d'une graine dans la matrice polynomiale **A**. La matrice **A** est nécessaire à la fois pour la signature et la vérification, donc une bonne mise en œuvre de SHAKE-128 est importante pour l'efficacité de l'algorithme.

Pour notre implémentation optimisée AVX2 du NTT, nous adoptons une approche qui consiste à utiliser l'arithmétique en nombres entiers plutôt qu'en virgule flottante. Bien que nous n'emballions que 4 coefficients dans un registre vectoriel de 256 bits, ceci est la même précision utilisée par les implémentations en virgule flottante, nous pouvons améliorer la vitesse de multiplication d'environ un facteur de 2. Nous avons réalisé cette accélération suite à un ordonnancement bien planifié des instructions et aussi à un entrelacement des multiplications et des réductions pendant le NTT de sorte que des phases de latences de multiplication soient cachées.

b. *Sécurité*

La sécurité de notre algorithme de signature peut être prouvée par le biais du ROM, sur la base de la difficulté de deux problèmes. Le premier est le problème standard LWE (sur des anneaux polynomiaux) qui demande de distinguer $(\mathbf{A}, \mathbf{t} \coloneqq \mathbf{As_1} + \mathbf{s_2})$ de $(\mathbf{A}, \mathbf{u})$, où **u** est uniformément aléatoire. L'autre problème est ce qu'on a appelé le problème SelfTargetMSIS, qui est le problème

de trouver un vecteur $\begin{bmatrix}\mathbf{z}\\ c\\ \mathrm{v}\end{bmatrix}$ avec de petits coefficients et un message μ satisfaisant :

$$\mathrm{H}\left(\mu|||[\mathbf{A}|\boldsymbol{t}|\mathbf{I}]\cdot\begin{bmatrix}\mathbf{z}\\ c\\ \mathrm{v}\end{bmatrix}\right)=c \qquad \textbf{(}5.16\textbf{)}$$

où **A** et t sont uniformément aléatoires et **I** est la matrice d'identité. Dans le ROM, on peut obtenir une réduction (non étroite) en utilisant le *lemme d'embranchement* (*forking lemma* en anglais) du problème MSIS standard de trouver un z_0 avec de petits coefficients satisfaisant $\mathbf{A}z_0 = 0$ au SelfTargetMSIS. On peut suivre cette approche exacte pour prouver que Dilithium est sécurisé dans le ROM en fonction de la difficulté du MLWE et SIS.

Tableau 5.05 : *Tailles de sortie, sécurité et performances de Dilithium.*

Niveau de sécurité NIST	2	3	5
Taille de sortie			
taille de la clé publique (bytes)	1312	1952	2592
taille de la signature (bytes)	2420	3293	4595
Difficulté du LWE (Core-SVP, et pour la version améliorée également)			
taille de bloc BKZ b (GSA)	423	624	863
Core-SVP classique	123	182	252
Core-SVP quantum	112	165	229
taille de bloc BKZ b (simulation)	433	624	863
$\log_2$ portes classiques	159	217	285
$\log_2$ mémoires classiques	98	139	187
Difficulté SIS (Core-SVP)			
taille de bloc BKZ b	423 (417)	638 (602)	909 (868)
Core-SVP Classique	123 (121)	186 (176)	265 (253)
Core-SVP quantum	112 (110)	169 (159)	241 (230)
Performance (Code de référence non optimisé, Skylake)			
Gen cycle médian	300,751	544,232	819,475
Sign cycle médian	1,081,174	1,713,783	2,383,399

Sign cycle moyen	1,355,434	2,348,703	2,856,803
Verify cycle médian	327,362	522,267	871,609
Performance (AVX2, Skylake)			
Gen cycle médian	124, 031	256, 403	298, 050
Sign cycle médian	259, 172	428, 587	538, 986
Sign cycle moyen	333, 013	529, 106	642, 192
Verify cycle médian	118, 412	179, 424	279, 936
Performance (AVX2+AES, Skylake)			
Gen cycle médian	70, 548	153, 856	153, 936
Sign cycle médian	194, 892	296, 201	344, 578
Sign cycle moyen	251, 144	366, 470	418, 157
Verify cycle médian	72, 633	102, 396	151, 066

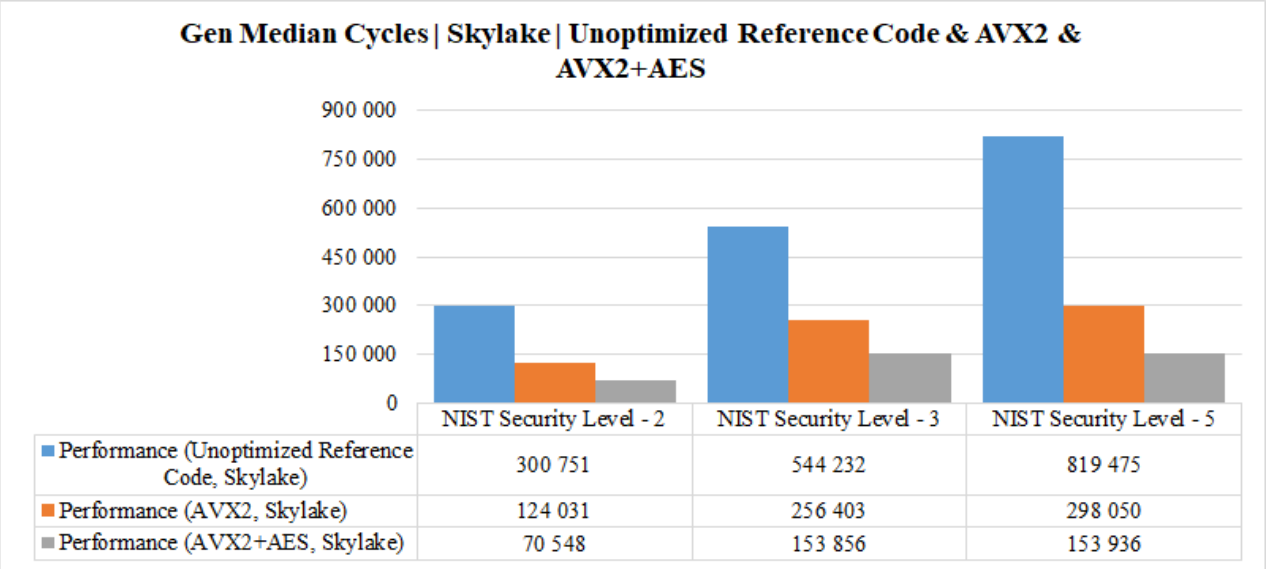


Figure 5.09 : *Nombre de cycles requis pour l'étape de génération de clé (Gen median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.*

Dans le QROM, où l'adversaire peut interroger H en superposition, la situation est un peu différente. Il a été montré dans que Dilithium est toujours basé sur MLWE et SelfTargetMSIS dans le QROM, même avec une réduction étroite lorsque l'algorithme est déterministe. Mais on ne peut plus utiliser directement le lemme d'embranchement (puisqu'il s'agit d'un type de rembobinage) pour donner une réduction quantique de MSIS à SelfTargetMSIS. Il y a encore de bonnes raisons de croire que le problème SelfTargetMSIS, et donc Dilithium, est sécurisé dans le QROM. Premièrement, il n'y a

pas de schémas de signature naturelle construits à partir de protocoles Σ utilisant la transformation Fiat-Shamir qui sont sécurisés dans le ROM mais pas dans le QROM. En outre, il est possible de définir les paramètres de Dilithium (tout en laissant la conception de l'algorithme inchangée) afin que le problème SelfTargetMSIS devienne théoriquement difficile, laissant ainsi cette version de Dilithium sécurisée dans le QROM basé uniquement sur MLWE. Une instanciation de ces paramètres aboutit [5.08] à un algorithme avec des signatures et des clés publiques qui sont respectivement 2x et 5x plus grandes. Bien que nous ne considérions pas cela comme un bon compromis, l'existence d'un tel système nous donne une confiance supplémentaire dans la sécurité d'une version optimisée de Dilithium.

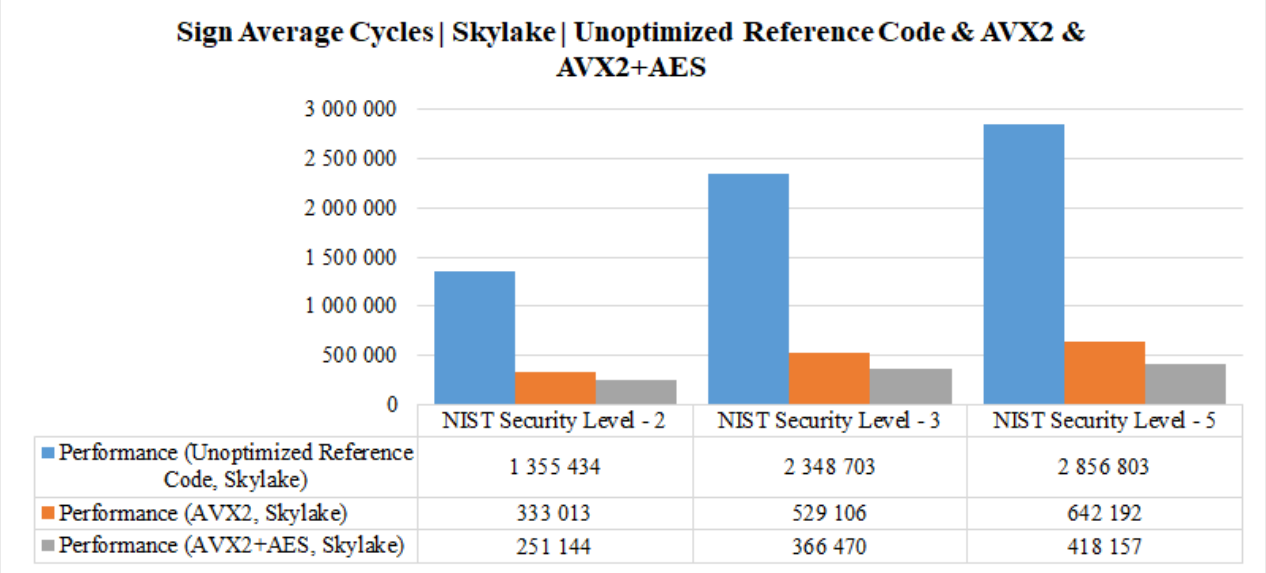
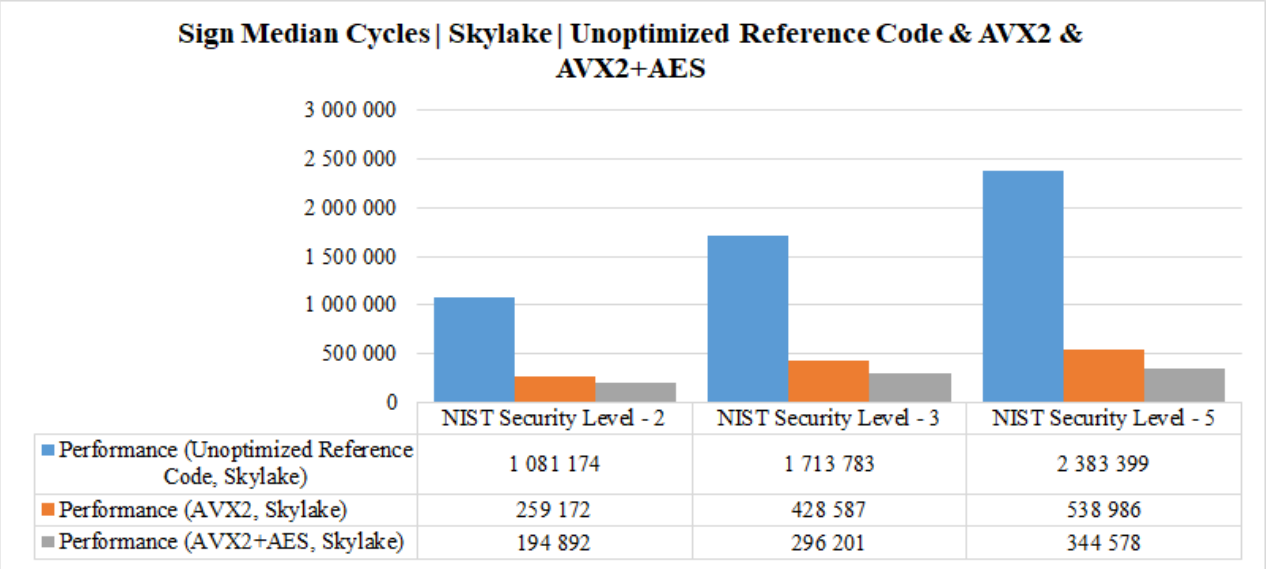


Figure 5.10 : *Nombre de cycles requis pour l'étape de signature / méthode de calcul moyenne (Sign average cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.*

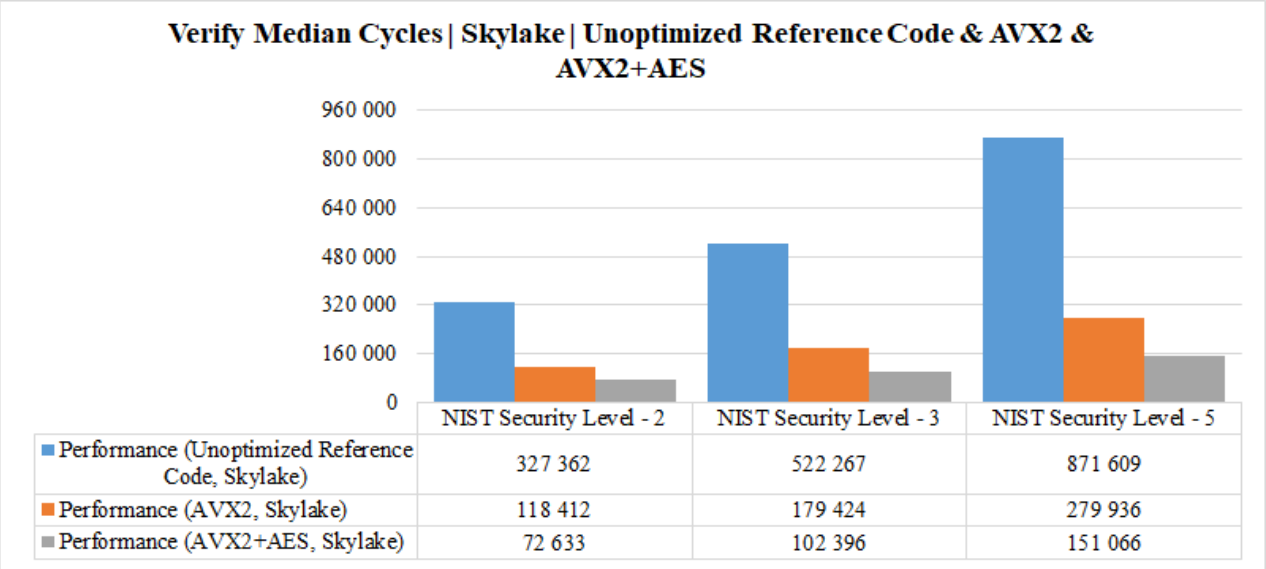
Récemment, de nouveaux travaux ont réduit encore plus l'écart entre la sécurité dans le ROM et le QROM. Ces travaux [5.08] ont montré que si le protocole Σ sous-jacent s'effondre et a une solidité particulière, alors sa transformation Fiat-Shamir est une signature sécurisée dans le QROM. La solidité particulière du protocole Dilithium est directement impliquée par la difficulté du MSIS. De plus, lesdits travaux, toujours, énoncent un probable effondrement de protocole Σ de Dilithium. Encore dans ces travaux est montré que la propriété d'effondrement a une réduction par rapport au MLWE. La réduction est plutôt non étroite, mais elle donne encore plus d'affirmation qu'il n'y a rien d'incertain dans la construction de Dilithium ou de tout algorithme naturel construit via le cadre

Fiat-Shamir dont la sécurité peut être prouvée dans le ROM. À notre avis, il est certain que la distinction entre les signatures sécurisées dans le ROM et QROM sera bientôt traitée de la même manière que la distinction entre les schémas sécurisés dans le modèle standard et le ROM - il y aura des différences théoriques, mais la sécurité dans la pratique sera la même.



	NIST Security Level - 2	NIST Security Level - 3	NIST Security Level - 5
Performance (Unoptimized Reference Code, Skylake)	1 081 174	1 713 783	2 383 399
Performance (AVX2, Skylake)	259 172	428 587	538 986
Performance (AVX2+AES, Skylake)	194 892	296 201	344 578

Figure 5.11 : *Nombre de cycles requis pour l'étape de signature / méthode de calcul médian (Sign median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.*



	NIST Security Level - 2	NIST Security Level - 3	NIST Security Level - 5
Performance (Unoptimized Reference Code, Skylake)	327 362	522 267	871 609
Performance (AVX2, Skylake)	118 412	179 424	279 936
Performance (AVX2+AES, Skylake)	72 633	102 396	151 066

Figure 5.12 : *Nombre de cycles requis pour l'étape de vérification (Verify median cycles), pour les différents jeux de paramètres (Niveau de sécurité NIST) de Dilithium.*

Dans le **Tableau 5.05**, les formules pour les tailles de la clé publique et de la signature sont dans

notre bibliographie [5.08]. Les chiffres entre parenthèses pour la difficulté SIS correspondent à la version fortement infalsifiable (c'est-à-dire qu'il est difficile de trouver une deuxième signature distincte pour un pair message/signature déjà vu) du schéma de signature. Pour la longueur du coefficient correspondant à la version sans parenthèse, nous renvoyons toujours nos lecteurs à notre bibliographie [5.08] pour plus de détails. En raison de l'échantillonnage de rejet, il y a une différence notable entre les temps de signature médians et moyens, et nous mesurons donc les deux.

Les figures **Figure 5.09**, **Figure 5.10**, **Figure 5.11** et **Figure 5.12** nous permettent d'avoir un aperçu visuel des résultats obtenus dans le **Tableau 5.05**.

5.3.2 Operations basiques

5.3.2.1 Opérations dans l'anneau

Des notions communes ont été déjà abordées dans notre paragraphe 5.2.1.1 . De ce fait, seules celles qui sont nouvelles ou où des précisions ont besoin d'être apportées feront l'objet des prochains paragraphes.

Anneau polynomial et vecteurs : même notion que dans notre paragraphe 5.2.1.1 , sauf que n et q ont des valeurs qui diffèrent légèrement. $n = 256$, toujours, mais $q = 2^{23} - 2^{13} + 1$.

Normes des éléments : idem que dans le paragraphe 5.2.1.1 . Une nouvelle notion cependant va suivre. Nous écrirons S_η pour désigner tous les éléments $w \in R$ tels que $\|w\|_\infty \leq \eta$. Nous écrirons $\tilde{S}_\eta$, pour désigner l'ensemble $\{w \bmod^{\pm} 2\eta : w \in R\}$. Notons que S_η et $\tilde{S}_\eta$ sont très similaires sauf que $\tilde{S}_\eta$ ne contient aucun polynôme avec des coefficients égaux à $-\eta$.

5.3.2.2 Représentation dans le domaine NTT

Notre module q est choisi de telle sorte que la 512ème racine de l'unité, $r \bmod q$ existe. Concrètement, on travaille toujours avec $r = 1753$. Cela implique que le polynôme cyclotomique $X^{256} + 1$ se scinde en facteurs linéaires $X - r^i \bmod q$ avec $i = 1, 3, 5, \ldots, 511$. Par le théorème du reste chinois (Chinese Reminder Theorem en anglais, pour rappel), notre anneau cyclotomique R_q est donc isomorphe au produit des anneaux $\mathbb{Z}_q[X]/(X - r^i) \cong \mathbb{Z}_q$. Dans ce produit d'anneaux,

il est facile de multiplier les éléments puisque la multiplication y est ponctuelle. L'isomorphisme :

$$a \longmapsto \left(a(r), a(r^3), \dots, a(r^{511})\right): R_q \longmapsto \prod_i \mathbb{Z}_q[X]/\left(X - r^i\right) \quad \textbf{(}5.17\textbf{)}$$

peut être évalué avec l'aide de la *Transformation de Fourier Rapide* (*Fast Fourier Transform* ou *FFT*, en anglais). Etant donné que $X^{256} + 1 = X^{256} - r^{256} = (X^{128} + r^{128})(X^{128} - r^{128})$, on pourra dans un premier lieu évaluer la correspondance :

$$\mathbb{Z}_q[X]/(X^{256} + 1) \longmapsto \mathbb{Z}_q[X]/(X^{128} - r^{128}) \times \mathbb{Z}_q[X]/(X^{128} + r^{128}) \quad \textbf{(}5.18\textbf{)}$$

et par la suite continuer séparément avec les deux polynômes réduits de degré inférieur à 128, en notant que $X^{128} + r^{128} = X^{128} - r^{324}$. La FFT est également appelée *Number Theory Transform* ou *NTT* dans le présent cas où le champ de travail est un champ fini. Les implémentations naturelles et rapides du NTT ne donnent pas à la sortie des vecteurs avec des coefficients dans l'ordre $a(r), a(r^3), \dots, a(r^{511})$. Ainsi, nous définissons le domaine de représentation de la NTT $\hat{a} = \mathrm{NTT}(a) \in \mathbb{Z}_q^{256}$ d'un polynôme $a \in R_q$ pour avoir des coefficients dans l'ordre des sorties données par notre NTT de référence. Concrètement :

$$\hat{a} = \mathrm{NTT}(a) = \left(a(r_0), a(-r_0), \dots, a(r_{127}), a(-r_{127})\right) \quad \textbf{(}5.19\textbf{)}$$

où $r_i = r^{\mathrm{brv}(128+i)}$ avec $\mathrm{brv}(k)$ représente le renversement des 8 bits du nombre k. Avec cette notation, et également grâce à la propriété de d'isomorphisme, nous avons $ab = \mathrm{NTT}^{-1}(\mathrm{NTT}(a)\mathrm{NTT}(b))$. Pour les vecteurs **y** et matrice **A**, les représentations $\hat{\mathbf{y}} = \mathrm{NTT}(\mathbf{y})$ **et** $\hat{\mathbf{A}} = \mathrm{NTT}(\mathbf{A})$ signifient que tous les polynômes y_i et a_{ij} constituant **y** et **A** suivent la représentation dans le domaine NTT.

5.3.2.3 Hachage

Le présent cryptosystème type signature utilise différents algorithmes qui hachent les chaînes de caractères $\{0,1\}^*$ dans des domaines à formes variées. Ci-dessous, nous donnons une description générale du ce concept.

Hachage en une boule. Soit B_τ dénotant un ensemble d'éléments de R ayant τ coefficient qui sont soit 1 soit -1 est le reste étant égal à 0. Nous avons $|B_\tau| = 2^\tau \cdot \binom{256}{\tau}$. Pour notre algorithme de signature, nous avons besoin d'une fonction de hachage cryptographique qui hache en B_τ en 2 étapes. La première étape consiste en l'utilisation d'une 2nde fonction de hachage cryptographique résistant à la pré-image pour faire correspondre $\{0,1\}^*$ au domaine $\{0,1\}^{256}$. La deuxième étape consiste en une XOF (exemple SHAKE-256) qui fait correspondre la sortie de la première étape à un élément de B_τ.

Algorithme 5.11 : SampleInBall(ρ)

```
1 : Initialiser c = c_0 c_1 ... c_255 = 00 ... 0
2 : pour i := 256 − τ à 255
3 :     j ← {0,1, ..., i}
4 :     s ← {0,1}
5 :     c_i := c_j
6 :     c_j := (−1)^s
7 : retourner c
```

La raison de ce processus en deux étapes est que la sortie de la première étape permet une description simple et compacte d'un élément dans B_τ (en supposant que le XOF est indiscernable d'une fonction purement aléatoire). L'algorithme que nous utiliserons pour créer un élément aléatoire dans B_τ est parfois appelé une version « inversée » du mélange *Fisher-Yates* [5.07], et sa description de haut niveau est illustrée dans l'**Algorithme 5.11**. Le XOF dans la deuxième étape est utilisé pour créer le caractère aléatoire nécessaire pour cet algorithme (dans les étapes 03 et 04) en utilisant la sortie de la première étape comme germe.

L'**Algorithme 5.11** pour faire simple permet la création d'un tableau à 256 éléments aléatoires avec des ± 1s au nombre de τ et des 0s au nombre de $256 - \tau$ en utilisant le germe ρ (et une XOF).

Etendre la matrice ***A.*** La fonction ExpandA fait correspondre un germe uniforme $\rho \in \{0,1\}^{256}$ à une matrice $\mathbf{A} \in \mathrm{R}_q^{k \times l}$ dans la représentation en domaine NTT. La matrice **A** est utile seulement durant les opérations de multiplication. Ainsi, dans le but d'avoir des implémentations rapides, la fonction d'extension ExpandA ne retourne pas $\mathbf{A} \in \mathrm{R}_q^{k \times l} = \mathbb{Z}_q[X]/(X^{256}+1)^{k \times l}$. A la place, elle retourne $\widehat{\mathbf{A}} \in \mathbb{Z}_q^{256}$ qui est interprêtée en tant que représentation de **A** dans le domaine NTT. Comme

A a besoin d'être échantillonnée uniformément, et que la NTT est un isomorphisme, ExpandA a également besoin d'être échantillonnée d'une manière uniforme dans cette représentation.

Echantillonner le vecteur $\boldsymbol{y}$**.** La fonction ExpandMask, utilisée pour générer le caractère aléatoire de notre cryptosystème type signature d'une manière déterministe, fait correspondre un germe ρ' et un nonce (un quelconque nombre aléatoire) κ à $\mathbf{y} \in \hat{S}_{\gamma_1}^l$.

Hachage résistant à la collision. La fonction CRH utilisée dans notre cryptosystème type signature est une fonction de hachage résistant aux collisions, qui fait correspondre à $\{0,1\}^{384}$.

5.3.2.4 Bits de poids fort/faible, indices h (de l'anglais hints)

Pour réduire la taille de la clé publique, nous avons besoin de quelques algorithmes simples pour extraire les « bits de poids forts » et les « bits de poids faibles » des éléments dans $\mathbb{Z}_q$. Le but en est que, quand un élément arbitraire $r \in \mathbb{Z}_q$ et un autre élément petit/négligeable $z \in \mathbb{Z}_q$, nous avons besoin d'être en mesure d'extraire les bits de poids forts de $r + z$ sans avoir besoin à sauvegarder z. Ainsi, nous avons besoin d'un algorithme qui prend à son entrée r et z et produit un indice h (de l'anglais hint) qui permettra de calculer les bits de poids fort de $r + z$ juste en utilisant r et h. Cet indice est généralement la « retenue » générée par z dans l'addition.

Il existe deux façons différentes de décomposer les éléments de $\mathbb{Z}_q$ en leurs bits « de poids fort » et « de poids faible ». Le premier algorithme, Power2Round_q, est la manière directe, bit par bit, de décomposer un élément $r = r_1 \cdot 2^d + r_0$ où $r_0 = r \bmod^{\pm} 2^d$ et $r_1 = (r - r_0)/2^d$.

Remarquez que si nous choisissons des valeurs de r_1 pour être des entiers non négatifs entre 0 et $\lfloor q/2^d \rfloor$, alors la distance (modulo q) entre deux $r_1 \cdot 2^d$ et $r_1' \cdot 2^d$ quelconques est généralement $\geq 2^d$, sauf pour le cas limite. En particulier, la distance modulo q entre $\lfloor q/2^d \rfloor \cdot 2^d$ et 0 peut être très petite. Cela pose un problème dans le cas où nous voudrions produire un indice de 1 bit, car l'ajout d'un petit nombre à r peut en fait faire varier les bits de poids fort de r de plus de 1.

Nous évitons que les bits de poids fort changent de plus de 1 avec l'astuce qui suit. Nous sélectionnons un α qui est un diviseur de $q - 1$ et écrivons $r = r_1 \cdot \alpha + r_0$ de la même manière que précédemment. Pour simplifier, nous supposons que α est pair (ce qui est possible, puisque q est impair). Les valeurs possibles de $r_1 \cdot \alpha$ sont maintenant $\{0, \alpha, 2\alpha, \ldots, q - 1\}$. Notons que la

distance entre $q-1$ et 0 est 1, et nous retirons donc $q-1$ de l'ensemble des valeurs possibles de $r_1 \cdot \alpha$, et arrondissons simplement les rs correspondants à 0. Puisque $q-1$ et 0 diffèrent de 1, tout ce que cela aura comme effet est d'augmenter éventuellement la magnitude du reste r_0 par 1. Cette procédure s'appelle Decomposer_q. En utilisant cette procédure en tant que sous-programme, nous pouvons définir les routines MakeHint_q et UseHint_q qui produisent un indice et, respectivement, utilisent l'indice pour récupérer les bits de poids fort de la somme. Pour des raisons de commodité, nous définissons également les routines HighBits_q et LowBits_q qui extraient simplement r_1 et r_0, respectivement, de la sortie de Decompose_q.

Algorithme 5.12 : *Algorithme supportant Dilithium.*

$\text{Power2Round}_q(r, d)$

1 : $r \coloneqq r \bmod^+ q$
2 : $r_0 \coloneqq r \bmod^\pm 2^d$
3 : **retourner** $((r - r_0)/2^d, r_0)$

$\text{MakeHint}_q(z, r, \alpha)$

4 : $r_1 \coloneqq \text{HighBits}_q(r, \alpha)$
5 : $v_1 \coloneqq \text{HighBits}_q(r + z, \alpha)$
6 : **retourner** $[\![r_1 \neq v_1]\!]$

$\text{UseHint}_q(h, r, \alpha)$

7 : $m \coloneqq (q-1)/\alpha$
8 : $(r_1, r_0) \coloneqq \text{Decompose}_q(r, \alpha)$
9 : **si** $h = 1$ et $r_0 > 0$ **retourner** $(r+1) \bmod^+ m$ **fin si**
10 : **si** $h = 1$ et $r_0 \leq 0$ **retourner** $(r-1) \bmod^+ m$ **fin si**
11 : **retourner** r_1

$\text{Decompose}_q(r, \alpha)$

12 : $r \coloneqq r \bmod^+ q$
13 : $r_0 \coloneqq r \bmod^+ \alpha$
14 : **si** $r - r_0 \coloneqq q - 1$ **alors**
15 : $r_1 \coloneqq 0$; $r_0 \coloneqq r_0 - 1$
16 : **sinon** $r_1 \coloneqq (r - r_0)/\alpha$ **fin si**
17 : **retourner** (r_1, r_0)

$\text{HighBits}_q(r, \alpha)$

18 : $(r_1, r_0) \coloneqq \text{Decompose}_q(r, \alpha)$
19 : **retourner** r_1

$\text{LowBits}_q(r, \alpha)$

18 : $(r_1, r_0) \coloneqq \text{Decompose}_q(r, \alpha)$
19 : **retourner** r_0

Les lemmes ci-dessous énoncent les propriétés de ces algorithmes de soutien qui sont nécessaires pour la correction et la sécurité de notre schéma.

Lemme 5.01 :

Supposons que q et α sont des entiers positifs satisfaisant à $q > 2\alpha$, $q \equiv 1 \pmod{\alpha}$ et α pairs. Soit $\mathbf{r}$ et $\mathbf{z}$ des vecteurs d'éléments dans R_q où $\|\mathbf{z}\|_\infty \leq \alpha/2$, et soit $\mathbf{h}$, $\mathbf{h}'$ des vecteurs de bits. Alors les algorithmes HighBits_q, MakeHint_q, et UseHint_q satisfont les propriétés suivantes :

(1) $\text{UseHint}_q\big(\text{MakeHint}_q(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha\big) = \text{HighBits}_q(\mathbf{z} + \mathbf{r}, \alpha)$;
(2) Soit $\mathbf{v}_1 = \text{UseHint}_q(\mathbf{h}, \mathbf{r}, \alpha)$. Alors $\|\mathbf{r} - \mathbf{v}_1 \cdot \alpha\|_\infty \leq \alpha + 1$. De plus, si le nombre de 1s dans $\mathbf{h}$ est ω, alors tous les coefficients de $\mathbf{r} - \mathbf{v}_1 \cdot \alpha$ sauf au plus ω auront une magnitude au plus égale à $\alpha/2$ après réduction centrée modulo q ;
(3) Pour tout $\mathbf{h}$, $\mathbf{h}'$. Si $\text{UseHint}_q(\mathbf{h}, \mathbf{r}, \alpha) = \text{UseHint}_q(\mathbf{h}', \mathbf{r}, \alpha)$, alors $\mathbf{h} = \mathbf{h}'$;

Lemme 5.02 :

Si $\|\mathbf{s}\|_\infty \leq \beta$ et $\big\|\text{LowBits}_q(\mathbf{r}, \alpha)\big\|_\infty \leq \alpha/2 - \beta$ alors $\text{HighBits}_q(\mathbf{r}, \alpha) = \text{HighBits}_q(\mathbf{r} + \mathbf{s}, \alpha)$;

Lemme 5.03 :

Soient $(\mathbf{r}_1, \mathbf{r}_0) = \text{Decompose}_q(\mathbf{r}, \alpha)$ et $(\mathbf{w}_1, \mathbf{w}_0) = \text{Decompose}_q(\mathbf{r} + \mathbf{s}, \alpha)$ et $\|\mathbf{s}\|_\infty \leq \beta$. Alors $\|\mathbf{s} + \mathbf{r}_0\|_\infty \leq \alpha/2 - \beta \Leftrightarrow \mathbf{w}_1 = \mathbf{r}_1 \wedge \|\mathbf{w}_0\|_\infty < \alpha/2 - \beta$;

5.3.3 Signature

Les algorithmes de génération, de signature et de vérification des clés pour notre cryptosystème type signature, sont présentés à l'**Algorithme 5.13**. Nous présentons la version déterministe du schéma dans laquelle le caractère aléatoire utilisé dans la procédure de signature est généré (en utilisant SHAKE-256 comme une fonction déterministe du message et d'une petite clé secrète). Puisque

notre procédure de signature pourrait être répétée plusieurs fois jusqu'à ce qu'une signature soit produite, nous ajoutons également un compteur dans le but de permettre à la sortie de SHAKE-256 d'être différent à chaque tentative de signature du même message. En outre, étant donné que chaque message (éventuellement long) peut nécessiter plusieurs itérations pour être signé, nous calculons une empreinte initiale du message en utilisant une fonction de hachage résistante aux collisions, et utilisons cette empreinte à la place du message tout au long de la procédure de signature.

Comme nous l'avons vu à la section 5.3.1.2 , la principale amélioration de conception de Dilithium par rapport au schéma de l'**Algorithme 5.10** est que la taille de la clé publique est approximativement divisée par deux au prix de moins d'une centaine d'octets supplémentaires dans la signature. Pour réaliser cette réduction de taille, l'algorithme de génération de clé produit $\mathbf{t}_1 := \text{Power2Round}_q(\mathbf{t}, d)$ comme clé publique au lieu de $\mathbf{t}$ comme dans l'**Algorithme 5.10**. Cela signifie qu'au lieu de $\lceil \log q \rceil$ bits par coefficient, la clé publique nécessite $\lceil \log q \rceil - d$ bits. Dans nos instanciations, $q \approx 2^{223}$ et $d = 13$, ce qui signifie qu'au lieu de 23 bits dans chaque coefficient de la clé publique, à la place il y en aura 10.

Algorithme 5.13 : *Le pseudo-code pour les versions déterministes et aléatoires de Dilithium. La seule différence entre les deux versions se trouve à la ligne 12, où ρ' est soit une fonction de la clé et du message, soit choisie complètement au hasard.*

Gen

01 : $\zeta \leftarrow \{0,1\}^{256}$

02 : $(\rho, \varsigma, K) \in \{0,1\}^{256\times 3} := \text{H}(\zeta)$ H est instancié suivant SHAKE-256

03 : $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^l \times S_\eta^k := \text{H}(\varsigma)$

04 : $\mathbf{A} \in R_q^{k\times l} := \text{ExpandA}(\rho)$ **A** est générée puis sauvegardée dans la représentation NTT en tant que $\widehat{\mathbf{A}}$

05 : $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ Calculer $\mathbf{A}\mathbf{s}_1$ en tant que $\text{NTT}^{-1}(\widehat{\mathbf{A}} \cdot \text{NTT}(\mathbf{s}_1))$

06 : $(\mathbf{t_1}, \mathbf{t_0}) := \text{Power2Round}_q(\mathbf{t}, d)$

07 : $tr \in \{0,1\}^{384} := \text{CRH}(\rho || \mathbf{t_1})$

08 : **retourner** $(pk = (\rho, \mathbf{t_1}), sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t_0}))$

Sign(sk, M)

09 : $\mathbf{A} \in R_q^{k\times l} := \text{ExpandA}(\rho)$ **A** est générée puis sauvegardée dans la représentation NTT en tant que $\widehat{\mathbf{A}}$

10 : $\mu \in \{0,1\}^{384} \coloneqq \mathrm{CRH}(tr||M)$

11 : $\kappa \coloneqq 0$, $(\mathbf{z}, \mathbf{h}) :=\perp$

12 : $\rho' \in \{0,1\}^{384} \coloneqq \mathrm{CRH}(K||\mu)$ (ou $\rho' \leftarrow \{0,1\}^{384}$ pour une signature aléatoire)

13 : **tant que** $(\mathbf{z}, \mathbf{h}) :=\perp$ **faire** Pré-calculer $\hat{\mathbf{s}}_1 \coloneqq \mathrm{NTT}(\boldsymbol{s}_1)$, $\hat{\mathbf{s}}_2 \coloneqq \mathrm{NTT}(\boldsymbol{s}_2)$ **et** $\hat{\mathbf{t}}_0 \coloneqq \mathrm{NTT}(\boldsymbol{t}_0)$

14 : $\mathbf{y} \in \tilde{S}_{\gamma_1}^{l} \coloneqq \mathrm{ExpandMask}(\rho', \kappa)$

15 : $\mathbf{w} \coloneqq \mathbf{A}\mathbf{y}$ $\mathbf{w} \coloneqq \mathrm{NTT}^{-1}\left(\hat{\mathbf{A}} \cdot \mathrm{NTT}(\mathbf{y})\right)$

16 : $\mathbf{w}_1 \coloneqq \mathrm{HighBits}_q(\mathbf{w}, 2\gamma_2)$

17 : $\tilde{c} \in \{0,1\}^{256} \coloneqq \mathrm{H}(\mu||\mathbf{w}_1)$

18 : $c \in B_\tau \coloneqq \mathrm{SampleInBall}(\tilde{c})$ sauvegarder c dans la représentation NTT en tant que $\tilde{c} \coloneqq \mathrm{NTT}(c)$

19 : $\mathbf{z} \coloneqq \mathbf{y} + c\mathbf{s}_1$ Calculer $c\mathbf{s}_1$ en tant que $\mathrm{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_1)$

20 : $\mathbf{r}_0 \coloneqq \mathrm{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ Calculer $c\mathbf{s}_2$ en tant que $\mathrm{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_2)$

21 : **si** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **ou** $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ alors $(z, h) :=\perp$

22 : **sinon**

23 : $\mathbf{h} \coloneqq \mathrm{MakeHint}_q(-c\mathbf{t}_0, wc\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ Calculer $c\mathbf{t}_0$ en tant que $\mathrm{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}}_0)$

24 : **si** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **ou** le # de 1s dans $\mathbf{h}$ est plus grand que ω **alors** $(\mathbf{z}, \mathbf{h}) :=\perp$

25 : $\kappa \coloneqq \kappa + l$

26 : **retourner** $\sigma = (\mathbf{z}, \mathbf{h}, \tilde{c})$

Verify$(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, \tilde{c}))$

27 : $\mathbf{A} \in R_q^{k \times l} \coloneqq \mathrm{ExpandA}(\rho)$ $\mathbf{A}$ est générée puis sauvegardée dans la représentation NTT en tant que $\hat{\mathbf{A}}$

28 : $\mu \in \{0,1\}^{384} \coloneqq \mathrm{CRH}(\mathrm{CRH}(\rho||\mathbf{t}_1)||M))$

29 : $c \coloneqq \mathrm{SampleInBall}(\tilde{c})$

30 : $\boldsymbol{w}_1' \coloneqq \mathrm{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$ Calculer $\mathrm{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathrm{NTT}(z) - \mathrm{NTT}(c) \cdot \mathrm{NTT}(\mathbf{t}_1 \cdot 2^d))$

31 : **retourner** $[\![\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta]\!]$ **et** $[\![\tilde{c} = \mathrm{H}(\mu||\mathbf{w}_1')]\!]$ **et** $[\![\#\text{de 1s dans } \mathbf{h} \text{ est} \leq \boldsymbol{\omega}]\!]$

Où $[\![.]\!]$ est le crochet d'Iverson (Iverson bracket en anglais). Retourne 1 si la condition qui s'y trouve est vraie. 0 sinon.

La principale complication due au fait que la clé publique ne contient pas la totalité de $\mathbf{t}$ est que l'algorithme de vérification n'est plus capable de calculer exactement $\mathbf{w}_1'$ à la ligne 13 de

l'**Algorithme 5.10**. Pour ce faire, l'algorithme de vérification aura besoin des bits d'ordre supérieur de $\mathbf{Az} - c\mathbf{t}$, mais il ne peut calculer que $\mathbf{Az} - c\mathbf{t}_1 \cdot 2^d = \mathbf{Az} - c\mathbf{t}_1 + c\mathbf{t}_0$. Même si les bits de poids fort de $c\mathbf{t}_0$ est 0, sa présence dans la somme crée des retenues (carries en anglais) qui peuvent avoir un effet sur les bits supérieurs. Le signataire envoie donc ces reports comme un indice au vérificateur. Heuristiquement, sur la base de nos choix des paramètres, il ne devrait pas y avoir plus de ω positions dans lesquelles une retenue est causée. Le signataire envoie donc simplement les positions dans lesquelles ces retenues se produisent (ce sont les octets supplémentaires dans Dilithium par rapport à la signature de l'**Algorithme 5.10**), ce qui permet au vérificateur de calculer les bits de poids fort de $\mathbf{Az} - c\mathbf{t}$.

5.3.3.1 Notes de mise en œuvre et compromis en matière d'efficacité

Pour que la taille de la clé publique (et secrète) reste petite, les procédures Sign et Verify commencent par l'extraction de la matrice $\mathbf{A}$ (ou plus précisément, sa représentation $\widehat{\mathbf{A}}$ dans le domaine NTT) de la graine ρ. Si l'espace de stockage n'est pas un facteur, alors $\widehat{\mathbf{A}}$ peut-être pré-calculée et faire partie de la clé publique/secrète. Le signataire peut en outre pré-calculer les représentations du domaine NTT de $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$ pour accélérer légèrement l'opération de signature. A l'autre extrême, si le signataire veut stocker une clé secrète aussi petite que possible, il n'a besoin de stocker qu'une graine secrète ζ de 32 octets qui est utilisée pour générer le caractère aléatoire pour créer ρ, K, $\mathbf{s}_1$ et $\mathbf{s}_1$ dans l'algorithme de génération de clé. De plus, il est possible de maintenir la mémoire pour les calculs intermédiaires à un niveau bas en ne gardant que les parties de la représentation du domaine NTT avec lesquelles on travaille actuellement.

5.3.3.2 Signatures déterministes et aléatoires

Dilithium offre la possibilité de produire des signatures déterministes (c'est-à-dire que la signature d'un message particulier est toujours la même) ou aléatoires. La seule différence d'implémentation entre les deux options se produit à la ligne 12 de l'**Algorithme 5.13** où la graine ρ' est soit dérivée du message et d'une clé (pour la signature déterministe), soit choisie de manière complètement aléatoire (pour la signature aléatoire).

On peut vouloir envisager des signatures aléatoires dans des situations où les attaques par canal latéral exploitant le déterminisme sont applicables [5.08]. Une autre situation où l'on peut vouloir éviter le déterminisme est lorsque le signataire ne souhaite pas révéler le message qui est signé. Bien

qu'il n'y ait pas de fuite temporelle de la clé secrète, il y a une fuite temporelle du message si le signataire ne veut pas révéler le message si le schéma est déterministe. Puisque le caractère aléatoire du schéma est dérivé du message, le nombre d'abandons pour un message particulier sera toujours le même.

5.3.3.3 Preuve du fait que la présente signature est correcte

Dans cette section, nous prouvons que le présent schéma de signature est correct.

Si $\|c\mathbf{t}_0\|_\infty < \gamma_2$, ainsi d'après le *Lemme 5.01* nous savons que :

$$\text{UseHint}_q(\mathbf{h}, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2) = \text{HighBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$$

étant donné que $\mathbf{w} = \mathbf{Ay}$ et $\mathbf{t} = \mathbf{As}_1 + \mathbf{s}_2$, nous avons :

$$\mathbf{w} - c\mathbf{s}_2 = \mathbf{Ay} - c\mathbf{s}_2 = \mathbf{A}(\mathbf{z} - c\mathbf{s}_1) - c\mathbf{s}_2 = \mathbf{Az} - c\mathbf{t} \quad \textbf{(}5.20\textbf{)}$$

and $\mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0 = \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d$. Aussi, celui qui vérifie calcule :

$$\text{UseHint}_q(\mathbf{h}, \mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2) = \text{HighBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$$

En addition, parce que β a été défini de telle sorte que $\|c\mathbf{s}_2\|_\infty \leq \beta$ et que le signataire vérifie à la ligne 21 que $\text{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) < \gamma_2 - \beta$, notre *Lemme 5.02* implique

$$\begin{aligned} \text{HighBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2) &= \text{HighBits}_q(\mathbf{w} - c\mathbf{s}_2 + c\mathbf{s}_2, 2\gamma_2) \\ &= \text{HighBits}_q(\mathbf{w}, 2\gamma_2) = \mathbf{w_1} \end{aligned} \quad \textbf{(}5.21\textbf{)}$$

Par conséquent, le $\mathbf{w_1'}$ calculé par le vérificateur dans l'entrée de la fonction de hachage est le même que le $\mathbf{w_1}$ du signataire. Et donc la procédure de vérification sera toujours acceptée.

5.3.3.4 Nombre d'itérations

Nous aimerions maintenant calculer la probabilité que l'étape 21 assigne $(\mathbf{z}, \mathbf{h})$ à $\perp$. La probabilité que $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ peut être calculée en considérant chaque coefficient séparément. Pour chaque coefficient σ de $c\mathbf{s}_1$ le coefficient correspondant de $\mathbf{z}$ sera entre $-\gamma_1 + \beta + 1$ et $-\gamma_1 + \beta - 1$

(inclusivement) tant que le coefficient correspondant de $\mathbf{y}_i$ est entre $-\gamma_1 + \beta + 1 - \sigma$ et $-\gamma_1 + \beta - 1 - \sigma$. Le poids de cette fourchette est $2(\gamma_1 - \beta) - 1$ et la coefficient de $\mathbf{y}$ a $2\gamma_1 - 1$ possibilités. Ainsi la probabilité pour que tout coefficient de $\mathbf{y}$ soit dans la bonne fourchette est de :

$$\left(\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1 - 1}\right)^{256 \cdot l} = \left(1 - \frac{\beta}{\gamma_1 - 1/2}\right)^{ln} \approx e^{-256 \cdot \beta l / \gamma_1} \quad \textbf{(}5.22\textbf{)}$$

où nous utilisons le fait que nos valeurs de γ_1 sont plus grandes comparées à $1/2$.

Nous nous attaquons par la suite au calcul de la probabilité que nous avons :

$$\|\mathbf{r}_0\|_\infty = \left\|\text{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\right\|_\infty < \gamma_2 - \beta$$

Si nous assumons (heuristiquement) que les bits d'ordre inférieur sont uniformément distribués modulo $2\gamma_2$, alors il y a :

$$\left(\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1 - 1}\right)^{256 \cdot k} \approx e^{-256 \cdot \beta k / \gamma_2}$$

probabilité que tous les coefficients soient dans la bonne fourchette (en utilisant le fait que nos valeurs de β sont grandes par rapport à $1/2$). Par conséquent, la probabilité que l'étape 21 réussisse est :

$$\approx e^{-256 \cdot \beta (l/\gamma_1 + k/\gamma_2)} \quad \textbf{(}5.23\textbf{)}$$

Tableau 5.06 : *Jeux de paramètres de Dilithium.*

Niveau de sécurité NIST	2	3	5
Paramètres			
q [module]	8380417	8380417	8380417
d [bits de $\mathbf{t}$ ignorés]	13	13	13
τ [# de bits 1s dans c]	39	49	60
défi en termes d'entropie $\left\lfloor \log \binom{256}{\tau} + \tau \right\rfloor$	192	225	257

γ_1 [fourchette de coefficient de **y**]	2^{17}	2^{19}	2^{19}
γ_1 [fourchette d'approximation pour l'ordre inférieur]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k,l) [dimension de **A**]	(4,4)	(6,5)	(8,7)
η [fourchette de la clé secrète]	2	4	2
β $[\tau \cdot \eta]$	78	196	120
ω [# de bits 1s max dans l'indice **h**]	80	55	75
Itérations (depuis l'équation **(**5.23**))**	4.25	5.1	3.85

Il est plus difficile de calculer formellement la probabilité que l'étape 24 aboutisse à un redémarrage. Les paramètres (se référer à notre **Tableau 5.06**) ont été fixés de telle sorte qu'heuristiquement $(\mathbf{z}, \mathbf{h}) = \perp$ avec une probabilité comprise entre 1% et 2%. Par conséquent, la grande majorité des répétitions de la boucle seront causées par l'étape 21.

Nous tenons à souligner que le nombre d'itérations prévu est indépendant de la clé secrète $\mathbf{s}_1$, $\mathbf{s}_2$ et donc aucune information à leur sujet ne peut être obtenue par une attaque qui compte les itérations.

5.4 Apport personnel et innovation des travaux de recherches

L'apport personnel du présent et dernier chapitre s'énonce suivant les deux points ci-dessous :

- Nous avons étalé d'une manière claire et nette les algorithmes définissant les cryptosystèmes CRYSTALS Kyber et Dilithium. Cryptosystèmes qui rappelons-le se basent sur l'objet mathématique MLWE ou Module Leaning With Error. Ce dernier, à son tour, possède une équivalence en termes de difficulté aux problèmes généraux relatifs à l'objet mathématique Lattice.

- Des jeux de paramètres ont été avancés pour s'aligner sur le niveau de sécurité détaillé dans le cahier des charges concocté par le NIST (National Institute of Standards and Technology). Des détails additionnels sur la performance et la sécurité de l'algorithme, entre autres, sont donnés. Pour le cryptosystème type KEM (Key Encapsulation Mechanism) surtout.

5.5 Conclusion

Ce dernier chapitre nous a permis de s'étaler sur les cryptosystèmes CRYSTALS Kyber et Dilithium.

Dans un premier lieu nous nous sommes concentrés sur le cas de CRYSTALS Kyber.

Nous avons avancé la spécification des deux parties qui le composent : le Kyber. CPAPKE et le Kyber. CCAKEM. Chacune de ces deux parties à leur tour se subdivise en 03 algorithmes : celui de la génération de clé, du cryptage, et du décryptage. Notons en passant le fait que les opérations de multiplication dans les dits algorithmes, sont effectuées dans le domaine NTT (Number Theoretic Transform). La multiplication dans R_q s'y exécute d'une manière efficace (à moindre coût).

En termes de performance, elle est à son apogée pour CRYSTALS Kyber lorsque ce dernier est écrit en prenant en compte les avantages des processeurs supportant la technologie AVX2 pour les processeurs type Intel Hashwell, et celui de la technologie m4 pour les processeurs type ARM.

Pour parler de la sécurité, celle de Kyber est basée sur la robustesse du problème sous-jacent qu'est le MLWE. Deux types de réduction ont été avancés dans le présent chapitre pour supporter ce fait : une réduction étroite par respect au ROM (Random Oracle Model) pour la partie Kyber. CPAPKE et une réduction non-étroite par respect au QROM (Quantique Random Oracle Model) pour celle Kyber. CCAKEM.

Dans un second lieu, le cas du CRYSTAL Dilithium a été également étalé, mais d'une manière plus retenue. Nos paragraphes se sont concentrés sur l'avancement de l'algorithme définissant ladite signature.

En addition à la notion de NTT qui est toujours d'actualité, des opérations de hachage sont également impliquées d'une manière plus marquée. On énumèrera en autres le hachage en boule qui est représenté en termes algorithmiques en la fonction $\mathrm{SampleInBall}(\rho)$. ρ représentant un germe.

Pour terminer, l'algorithme proprement dit de CRYSTALS Dilithium est donné. Il consiste en 03 sous parties : la génération de la clé, la signature et la vérification.

CONCLUSION GENERALE

Pour transmettre un message, l'on aurait toujours besoin que seuls ceux à qui le message est destiné soient capable de le lire. Que celui-ci (le message) soit caché (impossible à déchiffrer) pour tout personne autre que l'envoyeur et le récepteur. C'est dans ce contexte qu'entre le domaine dans lequel notre recherche s'est appliquée : la cryptographie. Son intitulé est : *« CONTRIBUTION À LA MODÉLISATION ET À L'ÉVALUATION DE LA PERFORMANCE DES CRYPTOSYSTÈMES POST QUANTIQUES À BASE DE MLWE - CAS DE CRYSTALS KYBER ET DILITHIUM »*

Un problématique que l'on pourrait avancer est le fait que de nouvelles méthodes de cryptographie et/ou le renforcement de celles déjà existantes ne cessent d'être recensées. A l'opposé, de nouvelles théories de décryptage font également surface.

Comme objectif de notre présente recherche, l'on a eu par exemple le fait de cartographier les méthodes cryptographiques existant en partant de la première fois où une trace de l'usage de la discipline fut trouvée selon la littérature. Pour continuer par celles qui se trouvent être les plus utilisées en notre ère. Et clore par celles qui sont candidates aux futurs exigences du domaine.

Dans cette thèse, nous avons contribué à la modélisation et à l'évaluation de performance des cryptosysèmes post quantiques à base de MLWE (Learning With Error over Modules).

Pour ce faire, nous avons dans un premier temps mis en exergue le fait que l'outil mathématique lattice représente une piste prometteuse en termes de problème sous-jacent sur lequel devrait se baser les cryptosystèmes post quantiques.

Nous avons par la suite démontré qu'une équivalence prenait place, en termes de difficulté, entre les problèmes généraux relatifs à la technologie mathématique lattice (Mod-GIVP, ou Generalized Independent Vector Problem restreint sur les Module Lattices, en particulier) et le MLWE.

Nous avons, toujours dans le même optique d'avancement, modélisé chacun des blocs constituant le cryptosystème CRYSTALS, faisant office d'application concrète aux théories précédemment étalées. Le cryptosystème CRYSTALS est composé de Kyber (cryptosystème type KEM, ou Key Encapsulation Mechanism) et Dilithium (cryptosystème type signature). CRYSTALS est le seul gagnant du projet Post-Quantum Cryptography Standardization, initié par le NIST (National

Institute of Standards and Technology), qui a pu faire élire à la fois un cryptosystème type KEM et signature.

Nes recherches s'est clôt sur l'analyse de la performance et de la sécurité des algorithmes composant ledit cryptosystème. Des jeux de paramètres ont été avancés pour s'aligner sur le niveau de sécurité détaillé dans le cahier des charges concocté par le NIST.

En termes de performance, elle est à son apogée pour CRYSTALS Kyber lorsque ce dernier est écrite en prenant en compte les avantages des processeurs supportant la technologie AVX2 pour les processeurs type Intel Hashwell, et celui de la technologie m4 pour les processeurs type ARM.

Côté CRYSTALS Dilithium, des opérations de hachage sont impliquées d'une manière plus marquée. On énumèrera en autres le hachage en boule qui est représenté en termes algorithmique en la fonction $\mathrm{SampleInBall}(\rho)$. ρ représentant un germe.

ANNEXES

Annexe 1 Publication de notoriété nationale et internationale

A1.1 La cryptologie, ou cryptographie et cryptanalyse, de l'ère pré-informatique

MADA-ETI, ISSN 2220-0673, Vol.2, 2021, www.madarevues.gov.mg

La cryptologie, ou cryptographie et cryptanalyse, de l'ère pré-informatique

Razafimahefa F. O. [1], Randriamitantsoa P.A. [2], Randrianandrasana M.E.[3]

Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (LR-TASI)

École Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation (ED-STII)

Équipe d'Accueil Doctorale Télécommunication, Automatique, Signal et Images (EAD-TASI)

Université d'Antananarivo

BP 1500, Ankatso – Antananarivo 101 – Madagascar

[1]ombana.razafimahefa@gmail.com , [2]rpauguste@gmail.com, [3]emile3marie@gmail.com

Résumé

L'être humain était et est fasciné par l'utilité de cacher un message et de ne divulguer celui-ci qu'à ceux ou celles à qui il est destiné. Deux disciplines distinctes existent pour ce faire. D'un côté, la stéganographie, consiste à cacher le message dans un contenu. De l'autre, la cryptographie, consiste à présenter le message sous une autre forme pour que seul celui à qui il est destiné, soit en mesure de le lire. Le présent ouvrage tournera autour de la cryptographie.

D'une manière générale la cryptographie se définit comme étant l'étude des techniques permettant la communication sécurisée en présence de partie tierce que l'on nomme adversaires. Dans la littérature, l'envoyeur est souvent appelé sous le nom Alice ou personne A. Le récepteur Bob ou personne B. L'adversaire est quant à lui dénommé Eve de l'anglais eavesdropper, l'eavesdropping se définissant comme étant l'écoute secrète de conversation privée d'autrui, sans son consentement.

Avant d'aller plus loin dans notre recherche doctorale dans le domaine de la cryptographie, il nous semble impératif de connaître dans un premier temps ce que fut l'historique dudit domaine, soit en d'autres termes en quoi consistait la cryptographie au tout début, durant l'ère pré-informatique.

Mots clés : *Cryptologie, cryptographie, cryptanalyse, pré-informatique, formulation mathématique.*

Abstract

Humans were and are fascinated by the usefulness of hiding a message and only disclosing it to those to whom it is intended. Two separate disciplines exist to do this. On the one hand, steganography consists in hiding the message in a content. On the other hand, cryptography, consists in presenting the message in another form so that only the one for whom it is intended, is able to read it. This work will revolve around cryptography.

Generally speaking, cryptography is defined as the study of techniques allowing secure communication in the presence of a third party known as adversaries. In the literature, the sender is often called under the name Alice or person A. The receiver Bob or person B. The adversary is called Eve (from the English eavesdropper), the eavesdropping being defined as listening the private conversation of others without their consent.

Before going further in our doctoral research in the field of cryptography, it seems imperative to us to know at first what was the history of the said field, that is to say in other words what consisted of cryptography at the very beginning, during the pre-Information Technology era.

Keywords: *Cryptology, cryptography, cryptanalysis, pre-computer science, mathematical formulation.*

1 Chiffrement par substitution

Le terme *substitution* se réfère au fait qu'une lettre en claire est remplacée par une lettre chiffrée suivant une règle bien définie.

1.1 Chiffrement par addition

Désignons par C le texte chiffré et par P le texte en clair. Posons k la clé de chiffrement, qui sera dans le cas du présent chiffrement par addition (et pour le cas des quelques méthodes de chiffrement qui vont suivre d'ailleurs) un nombre entier. [1], [2]

$$C \equiv P + k \ modulo\ 26 \qquad (1.01)$$

Notons que le nombre 26 n'est autre que le nombre de lettres dans l'alphabet.

Exemple : En prenant $k = 3$, la lettre a après chiffrement devient D (a est équivalent à 1, $1 + 3$ donne 4, ce qui revient à D).

Le déchiffrement se fait par la formule qui suit : [1], [3]

$$P \equiv C - k \ modulo\ 26 \qquad (1.02)$$

Le nombre de caractères dans l'alphabet étant de 26, le nombre de valeurs que peut prendre notre clé de chiffrement k est limité à 26. Cela rend la méthode de chiffrement par addition sensible à l'*attaque par force brute* (*brute force attack*). Cette attaque consiste à essayer un à un les valeurs de clés possibles jusqu'à obtenir un résultat satisfaisant. [3]

1.2 Chiffrement par multiplication

La formule utilisée revient à ce qui va suivre : [1]

$$C \equiv kP \ modulo\ 26 \qquad (1.03)$$

Notons quand même que la valeur que peut prendre k ici devra suivre certaines conditions. Etant donné que 26 (le nombre de lettres de l'alphabet pour rappel) si décomposé en nombre

premiers donne 2×13. Pour que chacune des lettres en claire de l'alphabet ait une et une seule correspondance, k devrait être ni multiple de 2 ni de 13. k peut-être donc parmi $1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23$ et 25. Soit 13 valeurs possibles au total.

Exemple : Pour $k = 3$. a devient C après chiffrement $(3 \times 1 = 3)$. x devient T (soit $3 \times 24\ modulo\ 26 = 20$).

Le déchiffrement se fait logiquement selon la formule ci-dessous : [1]

$$P \equiv \frac{1}{k} C\ modulo\ 26 \qquad (1.04)$$

Il y a cependant un léger problème à utiliser cette formule. En effet, prenons le cas où C n'est pas divisible par k (la lettre chiffrée est A par exemple, ce qui équivaut à 1 et $\frac{1}{3}$ n'est pas un entier donc difficilement transformable en lettre).

Nous allons faire entrer la notion d'inverse multiplicative modulaire $\bar{k}$ telle que $\frac{1}{k} C\ modulo\ 26 = \bar{k} C\ modulo\ 26$. Dans ce cas nous avons notre formule de déchiffrement finale :

$$P \equiv \bar{k} C\ modulo\ 26 \qquad (1.05)$$

L'*algorithme d'Euclide étendu*, nous permet d'arriver à l'égalité qui suit : $9 \times 3 - 1 \times 26 = 1$, d'où notre $a = 9$ et $b = 1$. Soit $a = \bar{k} = 9$.

Pour en revenir au déchiffrement de notre lettre encodée A. A équivaut à 1 la lettre en claire correspondant est donc $P = 9 \times 1\ modulo\ 26 = 9$, ce qui revient à la lettre i.

Une autre forme d'attaque peut en venir à bout de la présente méthode de chiffrement, celle connue sous le nom de l'*analyse de la fréquence des lettres*. Cette technique revient à l'arabe *Abu Yusuf Yaqub ibn Ishaq al-Sabbah al-Kindi*, développé aux alentours du IXème siècle.

Dans toutes les langues, certaines lettres se répètent plus souvent que d'autres. Dans la langue anglaise par exemple, les lettres e, t et a sont celles qui apparaissent les plus fréquemment, respectivement de l'ordre de 12%, 9% et 8%. Pour un message encodé relativement long, nous pourrions ainsi étudier la fréquence d'apparition des lettres, en sortir les 3 premières qui reviennent le plus. Elles correspondront probablement aux lettres en claire e, t et a. Pour les messages courts par contre, cette méthode perd en précision. [1]

1.3 Chiffrement affine

Le chiffrement se fait par la formule qui suit : [1], [3]

$$C \equiv kP + m\ modulo\ 26 \qquad (1.06)$$

Exemple : pour $k = 3$ et $m = 13$, notre lettre en claire x deviendra T $(3 \times 24\ modulo\ 26 = 20)$ après application du chiffrement par

multiplication. T deviendra par la suite G $(20 + 13\ modulo\ 26 = 7)$ après application de la seconde méthode de chiffrement, le chiffrement par addition.

Le déchiffrement se fait en ordre inversé. Effectuer le déchiffrement par addition, puis faire suivre par le déchiffrement par multiplication : [1], [3]

$$P \equiv \bar{k}(C - m)\ modulo\ 26 \qquad (1.07)$$

Exemple : G après le déchiffrement par addition devient T $(7 - 13\ modulo\ 26 = 20)$. Nous avons vu dans notre paragraphe 1.2 que $\bar{3} = 9$. Après déchiffrement par multiplication T revient à notre lettre en claire x $(9 \times 20\ modulo\ 26 = 24)$.

1.4 Chiffrement homophonique

Le *chiffrement homophonique*, consiste à attribuer plus d'une correspondance en lettre chiffrée à une lettre en claire ayant une fréquence d'apparition élevée. A la lettre en claire *e* par exemple, à laquelle la fréquence d'apparition 12% est attribuée, on en fait correspondre 4 (quatre) lettres chiffrées (*V*, *@*, *&* et *–* par exemple). De sorte que la fréquence de la lettre en claire *e* retombe à 3% $(3\% = 12\%/4)$ lorsqu'elle est chiffrée, sous condition que le choix entre les lettres chiffrées *V*, *@*, *&* et *–* se fasse d'une manière totalement aléatoire. Et comme 3% étant approximativement la fréquence d'apparition de toutes les lettres de l'alphabet, si chacune d'entre elles avaient la même probabilité d'apparition $(3\% \simeq 1/26)$, nous sommes ici donc en train d'outrepasser le problème lié à la fréquence d'apparition des lettres. [1], [2], [3]

1.5 Phi test ou calcul de l'indice de coïncidence

Le *phi test* a été introduit par *William Friedman* dans les années 1920. Il consiste à calculer l'*indice de coïncidence (IC)* correspondant à un message chiffré. L'indice de coïncidence est la probabilité pour que 2 mêmes lettres apparaissent simultanément dans le message. La formule est comme suit :

$$IC = \sum_{i=1}^{z} (\mathbb{P}(c_i))^2 \qquad (1.08)$$

Où z étant le nombre de lettres de l'alphabet ;

c_i le i-ième lettre de l'alphabet du message chiffré ;

$\mathbb{P}(c_i)$ dénote la probabilité d'apparition de la lettre c_i dans le message chiffré.

$$IC_{CUD} = \sum_{i=1}^{26} \left(\frac{1}{26}\right)^2 = 26 \times \frac{1}{26^2} = \frac{1}{26} \simeq 0.038 \qquad (1.09)$$

MADA-ETI

IC_{CUD}, pour Indice de Coïncidence pour Chiffre Uniformément Distribué.

Notre formule sera équivalente à $IC_{anglais} = (\mathbb{P}(A))^2 + (\mathbb{P}(B))^2 + \cdots + (\mathbb{P}(Z))^2$, qui revient à la valeur ci-dessous selon la bibliographie [1] :

$$IC_{anglais} = 0.066 \qquad (1.10)$$

Ce que l'on peut conclure ici est que, pour tout message chiffré, l'on devrait avoir $0.038 \leq IC \leq 0.066$. Si IC est plutôt proche de 0.038, c'est que le chiffrement est probable d'avoir utilisé la méthode homophonique. Si plutôt proche de 0.066, le chiffrement est sûrement un simple chiffrement mono-alphabétique où la fréquence d'apparition des lettres a été répercutée dans le message chiffré.

1.6 Chiffrement affine de Hill, un chiffrement type polygraphique

Notons le fait que polygraphique signifie que le processus de chiffrement traite plus d'une lettre en claire à la fois.

La présente méthode de chiffrement fut développée par *Lester Hill* dans les années 1929. Elle consiste à subdiviser le texte en clair en plusieurs blocs de *n* lettre chacun. Et d'appliquer le processus de chiffrement tour à tour sur ces blocs de *n* lettres. [1], [2], [3]

La formule pour le chiffrement est comme ce qui suit : [1]

$$C \equiv KP + M \; modulo \; 26 \qquad (1.11)$$

Où :

C et P sont des vecteurs de taille n, dont les éléments sont respectivement les n lettres chiffrées et celles en claires.

$$C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \text{ et } P = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix};$$

K est une matrice carrée inversible de dimension $n \times n$. Elle représente un genre de facteur multiplicatif comme celui que nous avons vu dans notre paragraphe 1.2. Cette fois ci, c'est le déterminant de la matrice K, noté $det(K)$ et le nombre de lettre de l'alphabet 26 qui doivent être premiers entre eux :

$$K = \begin{pmatrix} k_{1,1} & \cdots & k_{1,n} \\ \vdots & \ddots & \vdots \\ k_{n,1} & \cdots & k_{n,n} \end{pmatrix};$$

M est également un vecteur de taille n et représente le facteur additionnel qui assure le chiffrement par addition, se référer au paragraphe 1.1 :

$$M = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix};$$

Exemple : Prenons $n = 2$, $K = \begin{pmatrix} 3 & 5 \\ 6 & 1 \end{pmatrix}$, et $M = \begin{pmatrix} 13 \\ 9 \end{pmatrix}$. Et essayons de chiffrer le digramme en clair *et*, soit $P = \begin{pmatrix} 5 \\ 20 \end{pmatrix}$:

MADA-ETI

$C \equiv \begin{pmatrix} 3 & 5 \\ 6 & 1 \end{pmatrix}\begin{pmatrix} 5 \\ 20 \end{pmatrix} + \begin{pmatrix} 13 \\ 9 \end{pmatrix} \; modulo\; 26 = \begin{pmatrix} 128 \\ 59 \end{pmatrix} \; modulo\; 26 = \begin{pmatrix} 24 \\ 7 \end{pmatrix}$, soit le digramme chiffré *XG*.

Le déchiffrement s'effectue par la formule qui suit : [1]

$$P \equiv \overline{\det(K)}\, {}^{t}com(K)(C - M) \; modulo\; 26 \qquad (1.12)$$

Où :

$\det(K)$ se réfère au déterminant de la matrice K. $\overline{\det(K)}$ donc représente la multiplicative inverse de $\det(K)$ modulo 26 ;

$com(K)$ représente la comatrice de la matrice K et ${}^{t}com(K)$ se réfère à la transposée de $com(K)$;

Exemple : Déchiffrons le premier digramme *XG* que nous avons obtenu dans notre paragraphe précédent 1.5.2.

$det(K) = (3)(-1)^{1+1}(1) + (6)(-1)^{1+2}(5) = 3 - 30 = -27$;

L'inverse multiplicatif de -27 modulo 26 en suivant la méthode détaillée dans le paragraphe 1.3.3 donne $\overline{det(K)} = -1$.

La matrice comatrice de K est :

$com(K) = \begin{pmatrix} (-1)^{1+1}(1) & (-1)^{1+2}(6) \\ (-1)^{1+2}(5) & (-1)^{2+2}(3) \end{pmatrix} = \begin{pmatrix} 1 & -6 \\ -5 & 3 \end{pmatrix}$;

${}^{t}com(K) = \begin{pmatrix} 1 & -5 \\ -6 & 3 \end{pmatrix}$;

Notre digramme en clair s'obtient donc comme suit :

$P \equiv (-1)\begin{pmatrix} 1 & -5 \\ -6 & 3 \end{pmatrix}\left(\begin{pmatrix} 24 \\ 7 \end{pmatrix} - \begin{pmatrix} 13 \\ 9 \end{pmatrix}\right) \; modulo\; 26 = \begin{pmatrix} -27 \\ 72 \end{pmatrix} \; modulo\; 26 = \begin{pmatrix} 5 \\ 20 \end{pmatrix}$, soit *ET*.

Nous avons connu jusqu'ici des attaques qui se basent uniquement sur la connaissance du texte chiffré. Un autre cas qui pourrait se présenter est que l'attaquant pourrait être en possession d'une portion de texte en clair également. L'attaquant peut venir à bout du présent chiffrement s'il est en possession d'autant de couple de (lettre en claire, lettre chiffrée) que d'inconnus dans notre équation de chiffrement (1.11). [1]

Prenons le cas par exemple des 2 digrammes que nous avons chiffrés précédemment (*et*, *XG*) et (*es*, *SF*) auxquels nous allons rajouter le couple de (lettre en claire, lettre chiffrée), également obtenu par le même processus de chiffrement, (*bi*, *LD*). Nous aurons :

(*et*, *XG*) donne $\begin{pmatrix} 24 \\ 7 \end{pmatrix} = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix}\begin{pmatrix} 5 \\ 20 \end{pmatrix} + \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$;

MADA-ETI

(*es*, *SF*) donne $\binom{19}{6} = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix} \binom{5}{19} + \binom{m_1}{m_2}$;

Et (*bi*, *LD*) donne $\binom{12}{4} = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix} \binom{2}{9} + \binom{m_1}{m_2}$;

Notre attaquant serait alors en possession d'une sorte de 6 équations à 6 inconnues, à l'issue desquelles il est en mesure de reconstituer les paramètres K et M du chiffrement affine de Hill utilisé.

$$\begin{cases} 24 = 5k_{1,1} + 20k_{1,2} + m_1 \ modulo\ 26\ (1) \\ 7 = 5k_{2,1} + 20k_{2,2} + m_2 \ modulo\ 26\ (2) \\ 19 = 5k_{1,1} + 19k_{1,2} + m_1 \ modulo\ 26\ (3) \\ 6 = 5k_{2,1} + 19k_{2,2} + m_2 \ modulo\ 26\ (4) \\ 12 = 2k_{1,1} + 9k_{1,2} + m_1 \ modulo\ 26\ (5) \\ 4 = 2k_{2,1} + 9k_{2,2} + m_2 \ modulo\ 26\ (6) \end{cases} ;$$

1.7 Chiffre d'Alberti

Le chiffre d'Alberti est notre premier exemple de chiffre *poly-alphabétique*. Un chiffre est dit comme tel (poly-alphabétique) lorsqu'il fait correspondre plus d'une lettre chiffrée à chacune des lettres en claires existantes.

Pour chiffrer un texte en clair, Alberti introduisit un outil nommé *Disque d'Alberti*. Il consistait en 2 cercles concentriques. [1]

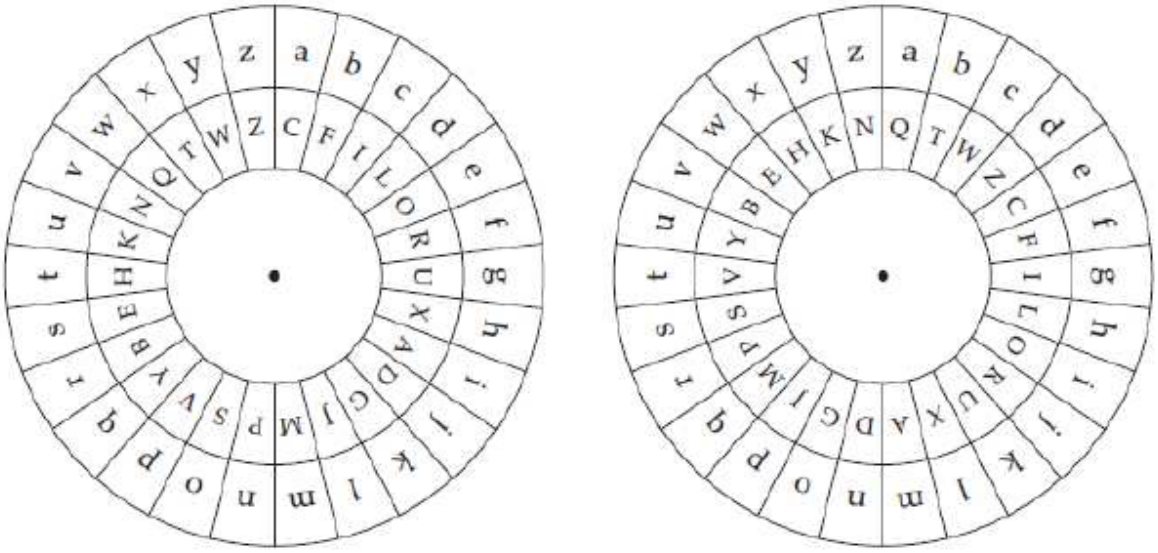


Figure 1.01 : *Disque d'Alberti avec dans un premier temps* $\boldsymbol{m_1 = 0}$ *à gauche, puis* $\boldsymbol{m_2 = -4\ modulo\ 26 = 22}$ *s à droite.*

En termes de modélisation mathématique, le chiffre d'Alberti est une combinaison d'un chiffre de permutation de clé k, que nous avons noté $f(k, P)$, et d'un chiffre d'addition de clé m_i.

La clé du chiffre d'addition, comme nous avons vu plus haut, a été proposé à être changer tous les 3 à 4 mots. D'où l'indice i dans m_i.

$$C \equiv f(k, (P + m_i))\ modulo\ 26 \qquad (1.13)$$

Une simplification que nous pouvons proposer est de prendre $f(k, P) = kP\ modulo\ 26$ Cela nous amène à un genre de chiffre affine comme suit :

$$C \equiv k(P + m_i)\ modulo\ 26 \qquad (1.14)$$

Exemple : Nous allons chiffrer la lettre *a* de notre premier syllabe *as* : $a \equiv 1, C \equiv 3 \times (1 + 0)\ modulo\ 26 \equiv 3 \equiv C$.

Chiffrons maintenant la première lettre *o* de notre seconde syllabe plus haut *ont*. $o \equiv 15, C \equiv 3 \times (15 + 22)\ modulo\ 26 \equiv 111\ modulo\ 26 \equiv G$.

Mentionnons quand même notre message chiffré final qui est *aCEeGDV* où *a* étant notre première lettre clé, et *e* en est la seconde.

1.8 La Tabula Recta

Le chiffrement utilisant le *Tabula Recta* ou *tableau dans le sens normal* (par opposition à sens inverse) fut introduit par *Johannes Trithemius* dans les années 1508. [1], [3]

En d'autres termes la modélisation mathématique est comme suit :

$$c_i \equiv p_i + i\ modulo\ 26 \qquad (1.15)$$

Où c_i étant la i-ième lettre chiffrée, p_i la i-ième lettre en claire et i étant le numéro d'itération en commençant par $i = 1$.

Exemple : Si nous voulons chiffrer la syllabe *as*, cela deviendra *BU*.

Il va de soi qu'une certaine Eve qui connait que la méthode de chiffrement utilisée par Alice et Bob se trouve être un simple Tabula Recta sans clé, déchiffrera sans problème le texte chiffré. Ce qui nous amène à l'amélioration proposée par *Vigenère*, celle d'introduire une notion de clé dans le présent mode de chiffrement.

1.9 Chiffre de Vigenère

Vigenère a suggéré qu'Alice et Bob devrait s'entendre sur un mot ou une phrase à définir comme clé à utiliser dans leur chiffrement faisant introduire le Tabula Recta.

La modélisation mathématique est donc comme suit : [1], [2], [3]

$$c_i \equiv p_i + m_i\ modulo\ 26 \qquad (1.16)$$

m_i étant la i-ième lettre du mot de passe, revenir à la première lettre à chaque fois que toutes les lettres aient été utilisées.

Exemple : Chiffrons la syllabe *ont* par le chiffre de Vigenère, en utilisant comme clé *as*.

- La lettre en claire *o* avec la lettre clé *a* donnent *P* si directement lu sur le Tabula Recta. Si calculé $c \equiv 15\ (soit\ o) + 1(soit\ a)\ modulo\ 26\ \equiv$

$16\ modulo\ 26 \equiv 16 \equiv P$;

- n et s donnent G ;

- t et a (nous étions arrivés à la dernière lettre s de la clé et revenons à la première, d'où a) donnent U.

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

Figure 1.02 : *Tabula Recta par Johannes Trithemius et amélioré par Vigenère.*

1.10 Indice de coïncidence pour un chiffre à clé périodique

L'indice de Coïncidence pour un Chiffre à Clé Périodique ou IC_{CCP}, est donné par la formule (1.17). [1]

Où n représente la longueur du message ;

l la longueur de la clé utilisée ;

IC_{CUD} et $IC_{anglais}$ respectivement l'indice de coïncidence d'un chiffre uniformément distribué et celui de la langue anglaise ;

$$IC_{CCP} = \frac{n - \frac{n}{l}}{n-1} \times IC_{CUD} + \frac{\frac{n}{l} - 1}{n-1} \times IC_{anglais} = \frac{n - \frac{n}{l}}{n-1} \times 0.038 + \frac{\frac{n}{l} - 1}{n-1} \times 0.066 \quad (1.17)$$

Remarquons que $IC_{CCP} = IC_{CUD} = 0.038$ pour $l = n$, en d'autres termes toutes les lettres du message chiffré ont été chiffrées différemment. Et $IC_{CCP} = IC_{anglais} = 0.066$ pour $l = 1$, en d'autres termes pour un chiffrement mono-alphabétique.

La connaissance de IC_{CCP} (peut être calculée vu que le message chiffré est connu par Eve) et de n (connu également donc) nous emmène à un genre d'équation à une inconnue l. D'où Eve qui peut suggérer une valeur de la longueur de la clé utilisée.

Exemple : Pour $n = 235$ et $IC_{CCP} = 0.044$, nous avons :

$$0.044 = \frac{235 - \frac{235}{l}}{235-1} \times 0.038 + \frac{\frac{235}{l} - 1}{235-1} \times 0.066;$$

Soit après résolution, $l = 4.59$. Une bonne suggestion en est $l = 5$.

1.11 Superposition et réduction

Eve a maintenant à grouper chacune des lettres du message chiffré selon qu'elles aient été chiffrées par la 1ère lettre du mot clé, par la 2nde, … jusqu'à la 5ème pour le cas de notre exemple de méthode de chiffrement qui est de période 5. Cette première action est connue sous le terme *superposition*. [1]

L'on obtient donc une portion de message chiffré par la même lettre clé, en d'autre termes utilisant une méthode de chiffrement mono-alphabétique – voire un chiffre additif – dans le cas du chiffre de Vigenère. Cette passage du chiffre de Vigenère, chiffre plus ardu à déchiffrer, à un plus simple chiffre qu'est le chiffre additif, se nomme *réduction*.

L'analyse de la fréquence des lettres de la première colonne nous indique que la lettre chiffrée L est celle qui revient le plus, elle pourrait correspondre à la lettre en claire e donc. Ce qui entraine que la première lettre du mot clé utilisé est G. $7\ (G) \equiv 12(L) - 5\ (e)$.

1.12 Amélioration de la robustesse du chiffre de Vigenère

Les ouvrages que nous avons lus parlent d'amélioration conséquente de la robustesse du chiffre si utilisé à deux reprises par exemple. Sous conditions que les deux clés utilisées aient une longueur dont les valeurs sont premiers entre eux, soit $PGCD(l_1, l_2) = 1$. [1]

Le double chiffre est équivalent à un chiffre unique dont le mot clé est de longueur l_3 avec :

MADA-ETI

$$l_3 = l_1 \times l_2 \qquad (1.18)$$

Exemple : Considérons les deux mots clés *OK* et puis *CURIOUSER*, de longueurs respectivement 2 et 9. Comme ces deux derniers sont premiers entre eux, la formule (1.18) dit qu'elles (les deux clés, et par conséquent le chiffre qui en résulte) équivaut à une clé de longueur $18 = 2 \times 9$. Cette clé résultat est *RFGTDFHPGNJCXZJDTC*, le calcul étant explicité dans le **Tableau 1.01**.

Clé 1	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*	*O*	*K*
Equivalent en nombre	15	11	15	11	15	11	15	11	15	11	15	11	15	11	15	11	15	11

Clé 2	*C*	*U*	*R*	*I*	*O*	*U*	*S*	*E*	*R*	*C*	*U*	*R*	*I*	*O*	*U*	*S*	*E*	*R*
Equivalent en nombre	3	21	18	9	15	21	19	5	18	3	21	18	9	15	21	19	5	18

Equivalent en nombre	18	6	7	20	4	6	8	16	7	14	10	3	24	26	10	4	20	3
Clé 3	*R*	*F*	*G*	*T*	*D*	*F*	*H*	*P*	*G*	*N*	*J*	*C*	*X*	*Z*	*J*	*D*	*T*	*C*

Tableau 1.01 : *Clé résultant de OK et CURIOUSER.*

1.13 Enigma, un type de machine cryptographique à rotor

La machine *Enigma* est une machine cryptographique à rotors dont la première version fut inventée vers la fin de la première guerre mondiale par un ingénieur allemand connu sous le nom d'*Arthur Scherbius*. Toujours dans cette première version, la machine possédait : [1], [2]

- 3 *rotors* qui vont être dans un premier temps traversés, de droite vers la gauche, par le courant électrique représentant la première étape de chiffrement ;

- Un *réflecteur*, qui effectue encore une fois, un chiffre par substitution. Le réflecteur subdivise l'alphabet de 26 lettres en 2 groupes de 13 lettres chacun. Fait correspondre par la suite chacune des lettres du premier groupe, à une autre dans le second groupe. Le réflecteur assure dans un premier temps que chacune des lettres en claires ne corresponde à une lettre chiffrée égale à elle (la lettre en claire) même. Et dans un second temps, de rendre réciproque le chiffre effectué par la machine Enigma ;

- Le courant électrique après avoir passé dans le réflecteur, ré-entre dans le labyrinthe des 3 rotors cités dans notre premier point, mais cette fois ci d'un ordre inversé de gauche vers la droite ;

- Dans les versions améliorées de la machine, l'on trouvait un *tableau de connexion*. Qui une n-ième fois encore, effectue une substitution. Le

tableau de connexion permet de définir manuellement le comportement du chiffre par substitution qui est appliqué. Le chiffre effectué par le tableau de connexion enveloppe (Effectue un chiffrement de la lettre en claire en premier. Passe la lettre ainsi chiffré à l'ensemble rotors, réflecteur puis rotors. Effectue le déchiffrement) tous les chiffre précédemment effectués dans nos précédents points ;

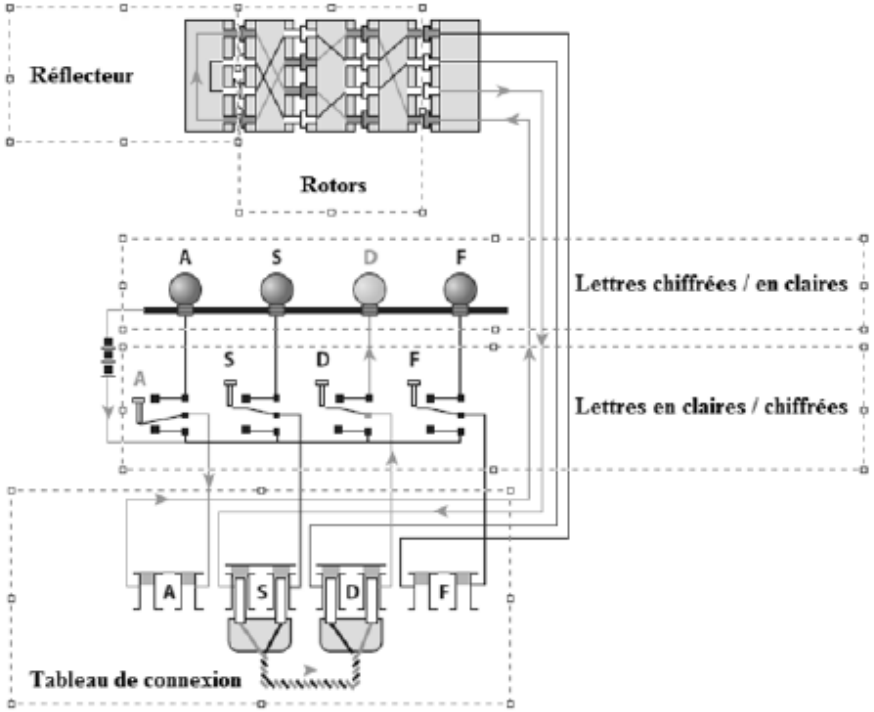


Figure 1.03 : *Le couple de (lettre en claire, lettre chiffré) ou (lettre chiffrée, lettre en claire) (A, D) traité par la machine Enigma.*

La formule du chiffrement effectué par une machine Enigma est représentée comme suit :

$$E \equiv TRMLUL^{-1}M^{-1}R^{-1}T^{-1} \qquad (1.19)$$

Où E représente le chiffrement (Encryption) de la machine Enigma dans sa globalité ;

T et T^{-1} représentent le chiffre du tableau de connexion (Table of connection ou plugboard) et son inverse ;

R et R^{-1} représentent le chiffre du rotor de droite (R pour Right) et son inverse ;

M et M^{-1} représentent le chiffre du rotor du milieu (M pour Middle) et son inverse ;

L et L^{-1} représentent le chiffre du rotor de gauche (L pour Left) et son inverse ;

U représente le chiffre symétrique du réflecteur.

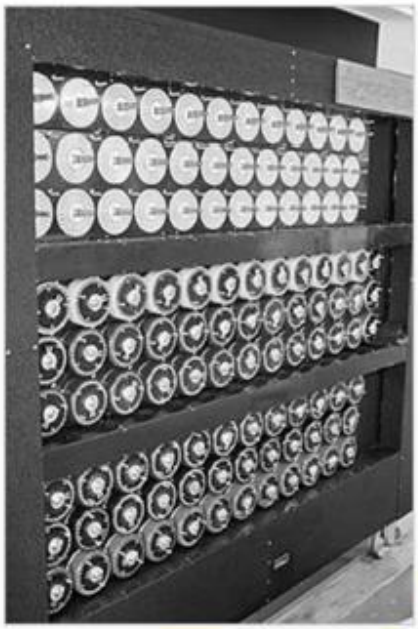


Figure 1.04 : *Une version reconstruite de la bombe qui marche réellement, exposée au musée du parc de Bletchley. La bombe possède 3 groupes de 36 rotors. Chaque rotor de la bombe simulant l'action d'un rotor de la machine Enigma.*

En termes de cryptanalyse de la machine Enigma, l'on peut avancer la machine connue sous le nom de *La bombe* et illustrée dans la **Figure 1.04**. Cette dernière est une machine électromécanique utilisée par les cryptologues britanniques afin de casser les codes allemands d'Enigma pendant la Seconde Guerre mondiale. La version la plus connue fut celle conçue par *Alan Turing* en 1939. Celle-ci (la version d'Alan Turing) tire toute fois son idée d'un même genre de machine créé par le cryptologue et mathématicien polonais *Marian Rejewski* un an plus tôt (en 1938). Pour être un an plus tard (en 1940) améliorée avec la participation cette fois ci de *Gordon Welchman*, un mathématicien anglais. [1], [4]

2 Chiffrement par transposition

Ce que nous avons vu jusqu'ici est que pour une lettre en claire donnée, ou un groupe de lettre en claires données, l'on associe une lettre chiffrée, ou groupe de lettres chiffrées. C'est ce qui a été connu sous le terme chiffrement par substitution.

Une autre manière de procéder est de laisser les lettres en claires telles qu'elles sont, mais d'uniquement chambouler leur position, pour donner vers la fin le message chiffré. Une telle méthode est connue sous le nom de chiffrement par transposition.

2.1 Méthodes de chiffrement dérivées du remplissage d'une forme géométrique

Il existe différentes manières de remplir une forme géométrique. Le colonel américain *Parker Hitt* en a fait l'objet de son ouvrage sorti durant la première guerre mondiale. [1], [3]

(a) Horizontal simple			
ABCDEF	FEDCBA	STUVWX	XWVUTS
GHIJKL	LKJIHG	MNOPQR	RQPONM
MNOPQR	RQPONM	GHIJKL	LKJIHG
STUVWX	XWVUTS	ABCDEF	FEDCBA

(c) Horizontal alterné			
ABCDEF	FEDCBA	XWVUTS	STUVWX
LKJIHG	GHIJKL	MNOPQR	RQPONM
MNOPQR	RQPONM	LKJIHG	GHIJKL
XWVUTS	STUVWX	ABCDEF	FEDCBA

(b) Vertical simple			
AEIMQU	DHLPTX	UQMIEA	XTPLHD
BFJNRV	CGKOSW	VRNJFB	WSOKGC
CGKOSW	BFJNRV	WSOKGC	VRNJFB
DHLPTX	AEIMQU	XTPLHD	UQMIEA

(d) Vertical alterné			
AHIPQX	DELMTU	XQPIHA	UTMLED
BGJORW	CFKNSV	WROJGB	VSNKFC
CFKNSV	BGJORW	VSNKFC	WROJGB
DELMTU	AHIPQX	UTMLED	XQPIHA

Tableau 2.01 : *Quelques manières de remplir une forme rectangulaire.*

2.2 Le chiffre par permutation

Le chiffre par permutation consiste à assigner une autre position à toutes les lettres en claires données du message. Pour praticité, le message est subdivisé en un nombre donné de lettres, puis la permutation est effectuée sur chaque groupe de lettres. Encore pour praticité, la longueur de subdivision du message en claire ainsi que l'ordre de permutation proviennent d'un mot que l'on pourrait désigner comme clé du chiffre. [1]

Le chiffrement est représenté sous la forme d'une matrice à 2 lignes et l colonnes, l étant la longueur du mot clé utilisé. La première ligne représente tout simplement un nombre croissant allant de 1 au nombre de lettres du mot clé (ici 4). La seconde, la succession de chiffre qui représente la permutation à effectuer : [1], [2]

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix} \qquad (2.01)$$

La première colonne de la matrice dans la formule (2.01) dicte donc que la première lettre du groupe de lettre du message chiffré est en fait la 4ème lettre du groupe de lettre correspondant dans le message en clair.

Pour obtenir la matrice de déchiffrement, échanger la première et la seconde ligne de la matrice. Ce qui donne $\begin{pmatrix} 4 & 1 & 3 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix}$. Puis trier suivant la première ligne, ce qui revient à la formule (2.02).

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix} \qquad (2.02)$$

La plage de clés (le nombre de clé possible) pour un chiffre par permutation est donnée par la valeur $l!$ (l factorielle) pour une clé de longueur l. Cette valeur est égale à $4! = 24$ pour un chiffre par permutation dont la clé est de longueur 4.

$$l! = \sum_{1 \le n \le l} n = l \times (l-1) \times \ldots \times 2 \times 1 \qquad (2.03)$$

2.3 Cryptanalyse de chiffre de type colonnaire ou à permutation. Méthode de contact de William Friedman

Nous allons accepter que l'on ait déjà pu savoir la longueur de la clé k utilisée. Il nous reste donc

à détecter, quelle colonne va en premier, suivie par laquelle.

Pour ce faire, William Friedman a proposé ce que l'on connait par méthode de contact. Cette méthode consiste à calculer la probabilité pour que les lettres de 2 colonnes données se suivent. En effet, l'on connait déjà pour la langue anglaise la probabilité d'apparition de tout digraphe donné.

Le produit de chacune de ces valeurs, ou la somme de leur logarithme donne la probabilité pour que chacune des lettres de 2 colonnes données se suivent. Une probabilité proche de 0 (soit 1 en valeur pour laquelle le logarithme n'a pas été appliqué) équivaut à deux colonnes qui a une forte probabilité de se suivre.

I	VII	Probabilité d'apparition	Logarithme - Probabilité d'apparition
O	H	0.001	-3.301
H	A	0.013	-1.886
I	T	0.010	-2.000
V	E	0.008	-2.097
R	H	0.001	-3.000
S	M	0.001	-3.301
V	E	0.008	-2.097
A	I	0.001	-3.000
H	E	0.017	-1.783
T	L	0.002	-2.824
B	B	-	
L	O	0.002	-2.699
R	H	0.001	-3.000
H	I	0.006	-2.222
L	U	0.002	-2.824
H	S	-	
L	I	0.005	-2.347
B	E	0.006	-2.260
Probabilité sur la totalité du message :			**-40.640**

I	VIII	Probabilité d'apparition	Logarithme - Probabilité d'apparition
O	K	-	
H	I	0.006	-2.222
I	U	-	
V	E	0.008	-2.097
R	U	0.002	-2.824
S	H	0.005	-2.301
V	E	0.008	-2.097
A	S	0.008	-2.097
H	L	0.001	-3.301
T	M	0.001	-3.301
B	T	0.001	-3.301
L	K	-	
R	S	0.005	-2.347
H	E	0.017	-1.783
L	C	0.002	-2.699
H	R	0.001	-3.000
L	E	0.009	-2.046
B	P	-	
Probabilité sur la totalité du message :			**-35.415**

Tableau 2.02 : *Méthode de contact appliqué aux couples de colonnes (I, VII) et (I, VIII). (I, VIII) a une plus forte probabilité de se suivre.*

3 Conclusion

Cet article nous a permis de se former une solide base sur ce qu'était la cryptographie depuis son apparition jusqu'à la fin de l'ère pré-informatique. Et en parallèle de s'initier à la cryptanalyse des différentes méthodes citées.

Rappelons que les méthodes citées dans le présent article constituent le fondement de ceux qui vont se faire en termes d'améliorations dans la discipline qu'est la cryptologie, cette fois-ci, pour l'ère de l'arrivée de l'informatique. Notre ère donc.

Mais le domaine de l'informatique étant surtout caractérisé par une capacité de faire faire des calculs complexes en un temps réduit, il est primordial pour nous de passer par des fondements mathématiques indispensable à la cryptologie. Pour ne citer que la théorie de l'information, celle des nombres ainsi que les mathématiques discrètes. Ceci fera l'objet de notre 2nd article qui s'intitulera *Quelques fondements mathématiques indispensables à la cryptologie*.

4 Bibliographie

[1]. J. Holden, « *The mathematics of secrets: cryptography from Caesar ciphers to digital encryption* », Princeton University Press, New Jersey, Feb. 2017 ;

[2]. W. Stallings, « *Cryptography and network security, principles and practice, seventh edition* », Pearson Education Limited, England, 2017 ;

[3]. C. Easttom, « *Modern cryptography, applied mathematics for encryption and information security* », McGraw-Hill Education, New York, 2016 ;

[4]. F. Carter, « *The Turing bombe* », http://www.rutherfordjournal.org/article030108.html, Jun. 2018 ;

MADA-ETI

A1.2 Théorie de l'information, théorie des nombres et mathématique discrète

MADA-ETI, ISSN 2220-0673, Vol.2, 2021, www.madarevues.gov.mg

Théorie de l'information, théorie des nombres et mathématique discrète

Razafimahefa F. O. [1]***, Randriamitantsoa P.A.*** [2]***, Randrianandrasana M.E.***[3]

Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (LR-TASI)

École Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation (ED-STII)

Équipe d'Accueil Doctorale Télécommunication, Automatique, Signal et Images (EAD-TASI)

Université d'Antananarivo

BP 1500, Ankatso – Antananarivo 101 – Madagascar

[1]*ombana.razafimahefa@gmail.com* , [2]*rpauguste@gmail.com*, [3]*emile3marie@gmail.com*

Résumé

Dans cet article, nous allons nous étaler sur des notions mathématiques indispensables à la cryptographie. Il est divisé en deux grandes parties. Celle relative à la théorie de l'information dans un premier temps. La seconde partie en est la théorie des nombres et la mathématique discrète.

Dans la théorie de l'information, nous allons voir entre autres les deux principaux théorèmes de Shannon : celui du codage source et du codage canal. Le premier stipule que le taux du code ne devrait être inférieur à l'entropie de la source, sinon l'on perdrait l'information. Le second affirme que pour un niveau de bruit donné, il existe un débit suivant lequel, l'on pourrait transmettre un message dans le canal.

Dans la seconde partie qu'est la théorie des nombres et la mathématique discrète, nous allons remettre en exergue des notions d'arithmétiques modulaires comme quoi les propriétés (commutativité, associativité, distributivité, identité et inverse) que nous connaissons avec les opérations standards (+,− et ×) tiennent également dans l'ensemble des restes de la division par N, l'ensemble $\mathbb{Z}_N$. Nous allons également développer la méthode de Miller-Rabin qui est une approche probabiliste en termes de confirmation ou non si un nombre n donné est premier.

Mots clés : *Théories de l'information, théorie des nombres, mathématique discrète.*

Abstract

In this article, we will expand on the mathematical concepts essential to cryptography. It is divided into two main parts. The first relating to information theory. The second part is number theory and discrete mathematics.

In information theory, we will see among other things the two main theorems of Shannon: that of source coding and channel coding. The first

MADA-ETI

stipulates that the code rate should not be lower than the entropy of the source, otherwise the information will be lost. The second asserts that for a given noise level, there is a rate at which a message could be transmitted in the channel.

In the second part, which is number theory and discrete mathematics, we are going to highlight notions of modular arithmetic such as the properties (commutativity, associativity, distributivity, identity and inverse) that we know with standard operations ($+$, $-$ and $\times$) also hold in the set of the remains of the division by N, the set $\mathbb{Z}_N$. We will also develop the Miller-Rabin method which is a probabilistic approach in terms of confirmation or not if a given number n is prime.

Keywords: *Information theories, source coding, channel coding, confusion, diffusion, complete transformation, avalanche effect, Hamming distance and weight, Kerckhoffs principle, and/or Shannon's maxim, binary mathematics, number theory, discrete mathematics, divisibility, division algorithm, GCD or Greatest Common Divisor, modular arithmetic, Euclid algorithm, extended Euclid algorithm, Chinese remainder theorem, factorization, prime numbers, Fermat theorem, Miller-Rabin algorithm.*

1 Théorie de l'information

1.1 Premier théorème de Claude Shannon ou théorème du codage source

Le théorème stipule qu'il est impossible de compresser une chaine de variables aléatoires indépendantes, identiquement distribuées (i.i.d.) – la longueur de la chaîne tendant donc vers l'infini – de telle sorte que le taux du code (la longueur moyenne des codes des variables) soit inférieur à l'entropie de la source, sans avoir la certitude que l'on perde de l'information. L'on peut cependant avoir une compression avec un taux de code proche de ladite entropie. [2], [3]

1.2 Deuxième théorème ou théorème du codage canal

Ce théorème statue qu'il est possible de transmettre des données numériques sur un canal, même bruité, presque sans erreur à un débit maximum calculable.

La limite ou capacité de Shannon d'un canal est le débit maximal de transfert d'information sur ce canal pour un certain niveau de bruit donné. [2], [4]

La formulation mathématique du théorème est attribuée par la littérature au binôme *Shannon-Hartley* et est donnée par ce qui suit :

$$C = B \log\left(1 + \frac{S}{N}\right) \qquad (1.01)$$

Où :

- C est la capacité du canal en bits par seconde (bits/s) ;

- B est la bande passante du canal en hertz (Hz) ;

- S/N est le rapport signal sur bruit (*SNR* ou *Signal to Noise Ratio* en anglais) ;

1.3 Entropie d'une information

L'entropie d'une information se définit comme étant la mesure de la quantité d'information contenue dans un message. Cette définition équivaut au fait que l'entropie est également la mesure de l'incertain dans un message. [2], [5], [6]

Elle est donnée par la formule (1.02) qui suit.

$$H(X) = -\sum_{k=1}^{K} p_k \log p_k \qquad (1.02)$$

Où :

X dénote une variable aléatoire i.i.d. et $H(X)$ l'entropie de X ;

K le nombre de lettres possibles de l'alphabet dans lequel sont tirés les lettres du message composant X ;

p_k la probabilité d'apparition de chaque lettre a_k de l'alphabet ;

1.4 Confusion et diffusion

Confusion et diffusion sont deux propriétés de méthode cryptographique énumérées par Claude Shannon. [2]

La confusion se veut d'être une propriété selon laquelle la relation entre le texte en clair et la clé de chiffrement doit être difficilement décelable. Une lettre chiffrée devrait donc dépendre de plusieurs lettres de la clé de chiffrement à la fois. Par opposition au fait de dépendre uniquement d'une seule lettre de la clé de chiffrement.

La diffusion quant à elle est une propriété qui garantit que toute redondance que l'on pourrait trouver dans le texte en clair ne le soit plus dans le texte chiffré. Soit en d'autres termes, elle (la redondance) est dissipée d'une manière convenable à une partie du texte chiffré. [2]

1.5 Transformation complète et effet d'avalanche

Une transformation est dite complète si chacune des lettres du texte chiffré dépend de toutes les lettres du texte en clair. Ainsi, s'il est possible de trouver une version simplifiée de ladite transformation pour chacune des lettres chiffrées, alors elle dépendrait en son entrée de toutes les lettres en claires. [6]

L'avalanche est une propriété de méthode cryptographique qui se veut qu'une petite modification dans le texte en claire ait une

MADA-ETI

répercussion importante dans le texte chiffré, comme une avalanche donc. Cette propriété a une similitude à celle qu'est la diffusion de Claude Shannon, l'avalanche est cette fois-ci attribuée par la littérature à *Horst Feistel*. Ce dernier était un chercheur américano-allemand en cryptographie, recherches qui ont culminé avec l'apparition de la *Data Encryption Standard* ou *DES* dans les années 1970. [6]

Le critère d'avalanche strict est une formalisation de l'effet d'avalanche. Il est satisfait si le fait de modifier un seul bit du texte en clair conduit à une modification de tous les bits de sortie du message chiffré avec une probabilité de 0.5, soit la moitié des bits chiffrés change en d'autres termes. Le concept de critère d'avalanche strict a été introduit par *Webster* et *Tavares* en 1985. [6]

1.6 Distance et poids de Hamming

La distance de Hamming est donnée par le nombre de caractères qui diffèrent entre deux textes donnés. Elle est dénotée par $h(x, y)$. La distance de Hamming est en d'autres termes le nombre de substitutions nécessaires pour aller d'un texte à un autre. [2]

Pour notre ère, qui est caractérisée par la cryptographie moderne, l'information représentée sous forme de bits. La distance de Hamming est donc donnée par le nombre de bits 1 résultant de l'opération eXclusive OR ou XOR entre les deux textes x et y.

$$h(x, y) = x \oplus y \qquad (1.03)$$

Où $\oplus$ représente l'opération XOR.

Le poids de Hamming est un peu identique au concept de distance de Hamming. Il s'agit de compter le nombre de bits égal à 1 dans un texte donné. Elle équivaut donc à calculer la distance de Hamming du texte à un second autre dont chacun des bits est égal à 0. [2]

1.7 Principe de Kerckhoffs et/ou maxime de Shannon

Le principe de Kerckhoffs se veut que « la sécurité d'un chiffre (méthode cryptographique) dépende uniquement du fait que la clé de chiffrement est tenue secrète et non du fait que les détails de l'algorithme lui-même soient tenus secrets ». Shannon, de par sa maxime a évoqué la formulation qui suit : « Nous devrions construire notre chiffre (méthode cryptographique toujours) sous la connaissance que l'ennemi pourrait être en possession de ladite algorithme ». [2]

Le principe de Kerckhoffs et la maxime de Shannon sont donc d'accords sur un point : seul la clé doit être tenue secret. Le fait de connaitre donc que quelqu'un ait utilisé du SHA-256 bits, du Blowfish, du Serpent ou quoi que ce soit d'autres comme type d'algorithme, ne met en rien en périls la sécurité de son message chiffré tant qu'il garde bien secrète sa clé de chiffrement.

1.8 Mathématique binaire

La mathématique binaire se définit comme étant la mathématique qui utilise la base 2 au lieu de la base ordinaire 10. La base 10 est plus connue par la plupart des personnes lambda suite au fait qu'il est plus facile de compter 10 par 10, le nombre de doigts et d'orteils que nous avons aidant sûrement.

Les opérations binaires nécessaires à la cryptographie sont principalement le OR, le AND et le XOR. L'on pourrait également ajouter l'opération NOT. Cette dernière, contrairement au OR, AND and XOR, requiert uniquement une entrée, si deux sont nécessaires pour les 3 premières opérations citées. [2]

Le résultat du OR est égal à 1 si au moins un de ses deux entrées est égale à 1. 0 sinon.

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Tableau 1.01 : *Opération binaire OR.*

Le résultat du AND est égal à 1 si toutes ses deux entrées sont égales à 1. 0 sinon.

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

Tableau 1.02 : *Opération binaire OR.*

Le résultat du XOR est égal à 1 ses deux entrées sont différentes. 0 sinon.

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Tableau 1.03 : *Opération binaire OR.*

L'opération XOR est très importante dans le domaine de la cryptographie. Ceci est due au fait que cette opération est non seulement réversible, mais que son inverse aussi n'est autre qu'elle-même. En d'autres termes, si l'on a $a\ XOR\ b\ =\ c$, alors l'on pourra retrouver a en évaluant $b\ XOR\ c$.

Le NOT consiste à complémenter un bit donné. Si son entrée est 0, la sortie en est 1. Si à l'entrée l'on a 1, sa sortie est 0.

a	NOT a
0	1
1	0

Tableau 1.04 : *Opération binaire NOT.*

2 Théorie des nombres et mathématique discrète

La théorie des nombres se définit comme étant l'étude des nombres entiers positifs. L'étude des nombres premiers et de leur factorisation constitue un aspect important de la théorie des nombres, du moins pour son application dans le domaine de la cryptographie.

MADA-ETI

Comme avant-goût, l'on pourrait citer par exemple l'usage des nombres premiers (ici 2) dans la constitution du corps de Galois (ou Galois Field an anglais) $GF(2^8)$ sur lequel se base la S-box (boîte à substitution) du chiffre SHARK. SHARK étant un algorithme de cryptographie prédécesseur à l'AES (Advanced Encryption Standard).

2.1 *Divisibilité et algorithme de division*

L'on dit qu'un nombre entier b divise a si l'on a $a = mb$ où m est également un nombre entier. La notation $b|a$ est majoritairement utilisé pour exprimer le fait que b divise a. b est dans ce cas également appelé *diviseur* de a. [1]

$$b|a \Leftrightarrow a = mb \qquad (2.01)$$

Exemple : Les diviseurs positifs de 24 sont $1, 2, 3, 4, 6, 8, 12$ et 24 lui-même.

Etant donné un nombre entier n, et un autre non négatif entier a, si nous divisons a par n, nous aurons deux entiers q (pour quotient) et r (pour reste) qui obéissent à la formulation : [1]

$$a = qn + r \text{ où } 0 \leq r < n \text{ et } q = \lfloor a/n \rfloor \qquad (2.02)$$

Où l'opérateur $\lfloor x \rfloor$ représente l'arrondissement a l'entier directement inférieur ou égal à x.

Exemple : $a = 11, n = 7$, l'on a $11 = 1 \times 7 + 4$ donc $q = 1$ et $r = 4$.

2.2 *Plus Grand Commun Diviseur ou PGCD*

Le *Plus Grand Commun Diviseur* de deux nombres entiers a et b est noté $\text{PGCD}(a, b)$. Comme son nom l'indique, c'est le plus grand nombre entier c qui divise à la fois a et b. [1]

Une définition équivalente en est, c est le PGCD de a et b si :

- c divise a et b ;

- Tout diviseur de a et b divise c ;

Exemple : $PGCD(24,60) = 12$.

a et b sont premiers entre eux si $\text{PGCD}(a, b) = 1$.

2.3 *Arithmétique modulaire*

Si a et n sont deux entiers, alors $a \bmod n$ se défini comme étant le reste de la division de a par n. L'on a : [1]

$$a = q \times n + r \text{ avec } q = \lfloor a/n \rfloor \text{ et que } 0 \leq r < n \qquad (2.03)$$
$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Où $\lfloor x \rfloor$ représente le nombre entier immédiatement inférieur ou égal à x.

Exemple : $11 \bmod 7 = 4$. Etant donné que $11 = 1 \times 7 + 4$.

MADA-ETI

Deux nombre a et b sont dit congruent modulo n si $a \bmod n = b \bmod n$. Cette propriété est dénoté par $a \equiv b \pmod n$.

$$a \equiv b \pmod n \Leftrightarrow a \bmod n = b \bmod n \quad (2.04)$$

Exemple : $73 \equiv 4 \pmod{23}$

Rappelons que d'après la formule (2.03), l'opération modulo associe à tout entier a un nombre entier de l'ensemble réduit $\{0,1,\dots,n-1\}$. La question se pose alors s'il serait possible d'effectuer des opérations dans cette ensemble. Et il s'avère que la réponse à cette question est oui. Elles (lesdites opérations) se définissent comme suit : [1]

- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$;

- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$;

- $[[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$;

Nous allons énumérer dans le tableau ci-dessous (**Tableau 2.01**) quelques propriétés relatives à l'arithmétique modulaire. [1]

Exemple : Pour l'additive inverse dans $\mathbb{Z}_8$, 2 est l'additive inverse de 6. De même 5 est l'additive inverse de 3. En effet, l'on a respectivement $(2+6) \bmod 8 = 8 \bmod 8 = 0$ et $(5+3) \bmod 8 = 8 \bmod 8 = 0$;

Pour la multiplicative inverse, dans le même ensemble $\mathbb{Z}_8$, la multiplicative inverse de 3 est 3. En effet, $(3 \times 3) \bmod 8 = 9 \bmod 8 = 1$. Rappelons que 3 a une multiplicative inverse dans $\mathbb{Z}_8$ car 3 et 8 sont premiers entre eux. Ainsi, 2 par exemple n'a pas de multiplicative inverse dans $\mathbb{Z}_8$ car 2 et 8 ne sont pas premiers entre eux. En effet $\mathrm{PGCD}(2,8) = 2$.

Propriétés	**Expression**
Loi commutatives	$(w+x) \bmod n = (x+w) \bmod n$; $(w \times x) \bmod n = (x \times w) \bmod n$;
Loi associative	$[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$; $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$;
Loi distributive	$[w \times (x+y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$;
L'identité	$(0+w) \bmod n = w \bmod n$; $(1 \times w) \bmod n = w \bmod n$;
Existence d'une additive inverse	Pour tout $w \in \mathbb{Z}_n$, $\exists\, z$ tel que $w + z = 0 \bmod n$;

MADA-ETI

Existence d'une multiplicative inverse, si w et n sont premiers entre eux	Pour tout $w \in \mathbb{Z}_n$, et que w et n sont premiers entre eux, $\exists\, z$, tel que $w \times z \bmod n = 1$

Tableau 2.01 : *Quelques propriétés de l'arithmétique modulaire.*

2.4 Algorithme d'Euclide et Algorithme d'Euclide Etendu

L'algorithme d'Euclide est un algorithme permettant de calculer le PGCD de deux nombres a et b donnés. [1]

Rechercher $\mathrm{PGCD}(a,b)$ revient à rechercher le $\mathrm{PGCD}(b,r_1)$. Ou encore d'une manière plus générale, à rechercher le $\mathrm{PGCD}(r_i,r_{i+1})$

$$\mathrm{PGCD}(a,b) = \mathrm{PGCD}(r_n,r_{n+1}) \text{ où } a \le b, n \ge 0, r_0 = b\,; \quad (2.05)$$

$$\mathrm{PGCD}(a,b) = \mathrm{PGCD}(r_n,r_{n+1}) = r_n \Leftrightarrow r_{n+1} = 0\,; \quad (2.06)$$

Exemple : Recherchons le $\mathrm{PGCD}(24,60)$.

L'on a donc $a = 24$ et $b = 60$. Comme $a < b$, l'on a à intervertir a et b. Donc nous allons continuer avec $a = 60$ et $b = 24$.

Maintenant, nous avons $60 = 2 \times 24 + 12$. L'on a donc $r = 12$. Comme $r = 12 > 0$, l'on a à continuer l'algorithme donc, cette fois ci en prenant $a = 24$ et $b = 12$.

L'algorithme d'Euclide étendu, en plus de calculer le $\mathrm{PGCD}(a,b)$, calcul également les deux entiers x et y justifiant l'équation ci-dessous : [1]

$$ax + by = \mathrm{PGCD}(a,b) \quad (2.07)$$

Pour ce faire, en plus de calculer les quotients et restes q_n et r_n, l'algorithme d'Euclide étendu calcule également les entiers additionnels x_n et y_n. Les étapes à suivre sont décrites dans le **Tableau 2.02**. L'algorithme s'arrête à l'itération $n+1$ où l'on obtient un reste $r_{n+1} = 0$. Ainsi, les valeurs de r_n, x_n et y_n justifient notre formule (2.07) précédemment citée, soit $ax_n + by_n = r_n = \mathrm{PGCD}(a,b)$.

i	q_i	r_i	x_i	y_i
−1	−	a	1	0
0	−	b	0	1
1	$\lfloor a/b \rfloor$	$a \bmod b$	$x_{-1} - q_1 x_0$	$y_{-1} - q_1 y_0$
2	$\lfloor b/r_1 \rfloor$	$b \bmod r_1$	$x_0 - q_2 x_1$	$y_0 - q_2 y_1$
3	$\lfloor r_1/r_2 \rfloor$	$r_1 \bmod r_2$	$x_1 - q_3 x_2$	$y_1 - q_3 y_2$

MADA-ETI

...	...	...	...	...
$\boldsymbol{n}$	$\lfloor r_{n-2}/r_{n-1} \rfloor$	$r_{n-2} \bmod r_{n-1}$	$x_{n-2} - q_n x_{n-1}$	$y_{n-2} - q_n y_{n-1}$
$\boldsymbol{n+1}$	$\lfloor r_{n-1}/r_n \rfloor$	$r_{n-1} \bmod r_n = 0$		

Tableau 2.02 : *Algorithme d'Euclide étendu.*

Exemple : effectuons l'algorithme d'Euclide étendu pour $a = 1212$ et $b = 630$. L'on a donc $n = 3, r_n = r_3 = 6, x_n = x_3 = 13$ et $y_n = y_3 = -25$. Soit $13 \times 1212 - 25 \times 630 = 6$.

2.5 Théorème des restes chinois

Le théorème des restes chinois peut s'écrire de différentes manières, mais celle qui nous concerne se détaille comme ce qui va suivre. Soit un entier N tel que : [1]

$$N = \prod_k n_k \text{ avec } 1 \leq k \leq K \qquad (2.08)$$

Où les n_k sont deux à deux premiers entre eux, soit $\forall\, 1 \leq k, l \leq K$ avec $k \neq l$, l'on a $\mathrm{PGCD}(n_k, n_l) = 1$. Nous pouvons représenter tout entier A dans $\mathbb{Z}_N$ par un K-uplet dont chaque éléments appartient à $\mathbb{Z}_{n_k}$.

$$A \leftrightarrow (a_1, \dots, a_K) \qquad (2.09)$$

Où $A \in \mathbb{Z}_N$, $a_k \in \mathbb{Z}_{n_k}$et $a_k = A \bmod n_k, \forall\, 1 \leq k \leq K$. Deux assertions en découlent :

- La correspondance dans l'équation (2.09) est bijective entre $\mathbb{Z}_N$ et l'ensemble produit cartésien $\mathbb{Z}_{n_1} \times \dots \times \mathbb{Z}_{n_K}$. Ainsi, pout tout entier $A \in \mathbb{Z}_N$ il existe un unique K-uplet $(a_1, \dots, a_K) \in \mathbb{Z}_{n_1} \times \dots \times \mathbb{Z}_{n_K}$ qui le représente. Et inversement tout K-uplet $(a_1, \dots, a_K) \in \mathbb{Z}_{n_1} \times \dots \times \mathbb{Z}_{n_K}$ correspond à un unique entier $A \in \mathbb{Z}_N$;

- Tout opération effectuée sur un élément $A \in \mathbb{Z}_N$ peut être effectuée indépendamment sur les K-uplet $(a_1, \dots, a_K) \in \mathbb{Z}_{n_1} \times \dots \times \mathbb{Z}_{n_K}$.

Une des propriétés utiles du théorème des reste chinois est qu'il permet de manipuler un nombre entier très large N en utilisant à sa place ses K-uplet, qui sont des entiers plus petits. Le prérequis en est cependant la nécessité de connaitre la factorisation de N.

Pour la formulation de A en fonction du K-uplet $(a_1, \dots, a_K)$, nous procèderons par sa construction. Soit $N_k = {}^{N}/_{n_k}$ pour $1 \leq k \leq K$. Remarquons que $N_k = n_1 \times \dots \times n_{k-1} \times n_{k+1} \times \dots \times n_K$. Soit $N_l \equiv 0 \pmod{n_k}\ \forall\, l \neq k$. Maintenant soit :

$$m_k = N_k \times (N_k^{-1} \bmod n_k) \equiv 1 \pmod{n_k} \text{ pour } 1 \leq k \leq K \qquad (2.10)$$

L'entier m_k de notre formule (2.10) est bien défini et est unique dû au fait que N_k est premier par rapport à n_k et donc possède une

MADA-ETI

multiplicative inverse $\bmod n_k$. Maintenant, A peut se calculer comme suit :

$$A \equiv \sum_{k=1}^{K} (a_k m_k) \bmod M \qquad (2.11)$$

Pour montrer que notre formulation de A dans l'équation (2.11) est juste, nous devons montrer que $A \bmod n_k = a_k$ pout tout $1 \leq k \leq K$. Rappelons que $m_l \equiv 0 \pmod{n_k}$ pour tout $l \neq k$. Ce qui revient à dire que $A \bmod n_k \equiv a_k m_k \bmod n_k = a_k$ étant donné que $m_k \bmod n_k = 1$ d'après la définition de m_k dans la formule (2.10).

Exemple : Considérons le cas où $N = 1813 = 37 \times 49$ soient $n_1 = 37$ et $n_2 = 49$, $A = 973$.

Nous avons donc aussi $N_1 = 49$ et $N_2 = 37$. En appliquant l'algorithme d'Euclide étendu, l'on a $N_1^{-1} = 34 \bmod n_1$ et $N_2^{-1} = 4 \bmod n_2$. En prenant le modulo de 973 respectivement par 37 et 49, l'on a la représentation de $A = 973$ qui est $(11, 42)$.

Maintenant, supposons que nous aimerions additionner 678 à 973. L'on a $678 \leftrightarrow (678 \bmod 37, 678 \bmod 49) \equiv (12, 41)$. Il nous suffit d'additionner les 2-uplets un à un et de les réduire. Soit $(11 + 12 \bmod 37, 42 + 41 \bmod 49) = (23, 34)$. Pour voir que notre calcul est correct, recherchons l'équivalent de $(23, 34)$ dans $\mathbb{Z}_{1813}$. Selon le formule (2.11), l'on a $(23, 34) \leftrightarrow (23 \times 49 \times 34 + 34 \times 37 \times 4) \bmod 1813 = 43350 \bmod 1813 = 1651$.

2.6 Factorisation de nombres premiers

Un nombre premier se définit comme étant un nombre positif ayant exactement deux diviseurs : 1 et lui-même. 1 n'est pas considéré comme étant un nombre premier.

Tout nombre entier $a > 1$ peut être écrit d'une manière unique sous la forme [1], [2] :

$$a = p_1^{a_1} \times p_2^{a_2} \times \ldots \times p_K^{a_K} \qquad (2.12)$$

Où $p_1 < p_2 < \cdots < p_K$ sont des nombres premiers et $a_1, a_2, \ldots, a_K$ sont des nombres entiers positifs.

Exemple : $3600 = 24^4 \times 32^2 \times 52^2$; $11011 = 7 \times 11^2 \times 13$;

Soit en d'autres termes si P étant l'ensemble regroupant tous les nombres premiers p_k. L'on aura :

$$a = \prod_{p_k \in P} p_k^{a_k} \qquad (2.13)$$

Où $a_k \geq 0$;

2.7 Théorème de Fermat et quelques propriétés utiles au test de primalité

Le théorème de Fermat statue que, si p est premier et que a est un nombre entier non divisible par p, l'on a : [1], [2]

MADA-ETI

$$a^{p-1} \equiv 1 \pmod{p} \qquad (2.14)$$

Avant d'entamer les détails du test de primalité de Miller-Rabin, il est nécessaire d'énumérer les quelques notions qui vont suivre. [1], [7]

Pour tout nombre impair n, l'on peut écrire :

$$n - 1 = 2^k q \qquad (2.15)$$

Où k est entier et q est impair.

Une première propriété utile des nombres premiers que nous aurons besoin ici également est que si p est un nombre premier et que a est un nombre entier inférieur à p, alors l'on a :

$$a^2 \bmod p = 1 \Leftrightarrow a \bmod p = 1 \text{ ou } a \bmod p = -1 \equiv p - 1 \qquad (2.16)$$

Où p est premier et $a < p$.

La deuxième propriété d'un nombre premier que nous aurons besoin également est que, si p est un nombre premier plus grand que 2, nous pouvons écrire $p - 1 = 2^k q$ avec k positif et q impair. Soit a un nombre entier tel que $1 < a < p - 1$. Alors, une des deux conditions suivantes est vraie :

- a congrue vers $1 \bmod p$. En d'autres termes, $a^q \bmod p = 1$, ce qui équivaut à $a^q \equiv 1 \pmod{p}$;

- Un des nombre $a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}$ congruent vers $-1 \bmod p$. En d'autres termes, il y a un nombre j tel que $1 \leq j \leq k$ tel que $a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$. Ce qui équivaut à $a^{2^{j-1}q} \equiv -1 \bmod p$.

2.8 *Algorithme de Miller-Rabin, un exemple de test de primalité*

Nous avons vu dans le paragraphe précédent que si n est premier, alors soit le premier élément de la liste $\left(a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}, a^{2^k q}\right) \bmod n$ est égal à 1, soit un élément de la liste est égal à $n - 1$; sinon n est un nombre composite. D'un autre côté, si un nombre n vérifie une des deux conditions précédemment énumérées, cela ne signifie pas forcément que n est premier. Considérons par exemple le nombre $n = 2047 = 23 \times 89$. $n - 1 = 2046 = 2 \times 1023$. L'on a $2^{1023} \bmod 2047 = 1$, qui justifie une de nos conditions, alors que 2047 n'est pas premier. [1], [7]

Nous pouvons cependant utiliser le présent algorithme pour diviser le fait qu'un nombre est définitivement composite, ou qu'il pourrait être premier ou non. Dénommons $TEST(n)$ cet algorithme, les étapes correspondant sont comme suit :

$TEST(n)$:

- Trouver les entiers k et q, avec $k > 0$ et q impair tel que $n - 1 = 2^k q$;

- Sélectionner un nombre aléatoire a, tel que $1 <$

$a < n - 1$;

- Si $a^q \bmod n = 1$ alors retourner « n pourrait être premier » ;

- Pour $j = 0$ à $k - 1$ faire si $a^{2^j q} \bmod n = n - 1$ alors retourner « n pourrait être premier » ;

- Retourner « n est composite » ;

Exemple : Essayons de voir si 29 est premier. L'on a $29 - 1 = 28 = 2^2 \times 7 = 2^k q$. Premièrement, essayons $a = 10$. L'on a $10^7 \bmod 29 = 17$. Ce qui n'est ni 1 ni 28. On continue avec $(10^7)^2 \bmod 29 = 28$. Ce qui nous emmène au résultat que 29 pourrait être premier. En essayant encore avec $a = 2$, l'on trouve $2^7 \bmod 29 = 12$ et $(2^7)^2 \bmod 29 = 28$. Ce qui nous emmène, encore une fois, au résultat que 29 pourrait être premier. Si nous réitérons le calcul pour tout $1 < a < 28$, nous obtiendrons toujours le résultat comme quoi n pourrait être premier.

L'algorithme de Miller-Rabin est une méthode à approche probabiliste, comme nous l'avons déjà mentionné un peu plus haut. Il est avancé dans la littérature que cet algorithme a une probabilité inférieure à ¼ de ne pas réussir à détecter qu'un nombre composite l'est (retourne en d'autres termes un résultat comme quoi le nombre pourrait être premier). Répéter l'algorithme de Miller-Rabin t fois revient donc à réduire à $(1/4)^t$ la probabilité de cette erreur. [1], [7]

Procéder donc comme ce qui va suivre. Pour un nombre impair n donné, choisir aléatoirement un nombre $1 < a < n - 1$. Si $TEST(n)$ retourne « composite » alors n est composite. Si par contre l'algorithme retourne « n pourrait être premier », choisir une autre valeur de a. Si après un nombre assez élever de t tests l'algorithme continue à retourner « n pourrait être premier », assumer que n est premier.

La démonstration de ce qui a été annoncé précédemment (celle qui avance la valeur ¼ comme borne supérieure de la probabilité d'erreur de l'algorithme de Miller-Rabin, pour rappel) s'avère être assez longue. Nous renvoyons ainsi nos lecteurs dans notre bibliographie [7], si besoin est.

3 Conclusion

Cet article nous a permis de se former une solide base sur les fondements mathématiques indispensable à la cryptologie. Pour ne citer que la théorie de l'information, celle des nombres ainsi que les mathématiques discrètes.

Il est quand même assez difficile de couvrir la totalité du domaine. L'on pourrait citer entre autres - les notions de calcul matricielle, la loi des grands nombres, la règle de l'hôpital, l'inégalité de Jensen - notions que nous nous sommes permis d'omettre pour le moment, mais qui ont été utilisées dans nous ouvrages jusqu'ici.

Nous essayerons cependant de trouver une place pour ces notions (en annexe ou autres) dans notre ouvrage final qui consistera en notre thèse doctorale.

4 Bibliographie

[1]. W. Stallings, « *Cryptography and network security, principles and practice, seventh edition* », Pearson Education Limited, England, 2017 ;

[2]. C. Easttom, « *Modern cryptography, applied mathematics for encryption and information security* », McGraw-Hill Education, New York, 2016 ;

[3]. K. Bostrom, « *Shannon source coding theorem* », Institut fur physik - Universitat Potsdam, Germany, Apr. 2015 ;

[4]. P. Milne « *Channel Capacity* », http://www.dip.ee.uct.ac.za/~nicolls/lectures/eee482f/04_chancap.pdf , 2003 ;

[5]. « Entropy (Information Theory) », https://en.wikipedia.org/wiki/Entropy_(information_theory), 2019 ;

[6]. A. Webster and S. Tavares, « *Chapter On The Design Of S-Boxes* », Electrical Engineering - Queen's University, Kingston Ont. Canada, 1986 ;

[7]. K. Conrad, « *Miller-Rabin Test* », https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf, 2019 ;

MADA-ETI

A1.3 Application du MLWE dans le cryptosystème CRYSTALS - Kyber comme KEM et Dilithium comme signature

MADA-ETI, ISSN 2220-0673, Vol.2, 2021, www.madarevues.gov.mg

Application du MLWE dans le cryptosystème CRYSTALS - Kyber comme KEM et Dilithium comme signature

Razafimahefa F.O.[1], Randriamitantsoa P. A.[2], Randrianandrasana M.E.[3]

Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (LR-TASI)

École Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation (ED-STII)

Équipe d'Accueil Doctorale Télécommunication, Automatique, Signal et Images (EAD-TASI)

Université d'Antananarivo

BP 1500, Ankatso – Antananarivo 101 – Madagascar

[1]ombana.razafimahefa@gmail.com , [2]rpauguste@gmail.com, [3]emile3marie@gmail.com

Résumé

Les ordinateurs quantiques, mais toujours avec un faible nombre de qubits, commencent à émerger au moment où nous écrivons. Il est plus crucial que jamais de travailler sur de nouveaux cryptosystèmes. Des cryptosystèmes qui seront basés sur des problèmes mathématiques suffisamment robustes pour résister à la puissance de calcul accrue que ces nouveaux types d'ordinateurs fourniront. Au cours de nos recherches, nous sommes tombés sur l'objet mathématique qu'est la Lattice. Cette dernière est l'un des candidats prometteurs en termes de problème fondamental pour les cryptosystèmes post-quantiques. Mais l'outil mathématique sur lequel nous allons ouvrir cet article est le MLWE ou Module Learning With Error. Le LWE (ou Learning With Error), en premier lieu, est une généralisation du problème Learning from parity with error. Il a été montré que ce problème est équivalent, en termes de difficulté, au SIVP ou Shortest Independent Vector Problem d'une lattice. Pour en revenir au MLWE, nous avons une autre désignation, Learning With Error other Module Lattices. Les modules lattices peuvent être considérées comme des lattices entre celles utilisées dans les définitions du problème LWE et celles utilisées pour le problème R-LWE (Ring LWE). Le Module-LWE offre un compromis entre les deux extrêmes du LWE et du R-LWE. Par la suite, nous nous concentrerons sur le cryptosystème CRYSTALS, un cryptosystème basé sur le MLWE. Notre choix d'étudier ce cryptosystème est basé sur le fait qu'il a été soumis au projet du NIST (National Institute of Standards and Technology) de standardisation des cryptosystèmes post-quantiques. CRYSTALS a pu progresser jusqu'à l'étape 3 du projet. Mais surtout, il est le seul parmi les rares à avoir progressé jusqu'à cette étape finale, à avoir à la fois un cryptosystème de

MADA-ETI

type KEM (Key Encapsulation Mechanism) et un autre de type signature.

Mots clés : *Ordinateur quantique, cryptosystème post-quantique, module LWE, CRYSTALS-Kyber, CRYSTALS-Dilithium, mécanisme d'encapsulation de clés (KEM), signature.*

Abstract

Quantum computers, but still with a low number of qubits, are starting to emerge as we write. It is more crucial than ever to work on new cryptosystems. Cryptosystems that will be based on mathematical problems that are robust enough to withstand the increased computational power that these new types of computers will provide. In the course of our research, we came across the mathematical object that is the Lattice. The latter is one of the promising candidates in terms of fundamental problem for post-quantum cryptosystems. But the mathematical tool on which we will open this article is the MLWE or Module Learning With Error. The LWE (or Learning With Error), in the first instance, is a generalization of the problem Learning from parity with error. It has been shown that this problem is equivalent, in terms of difficulty, to the SIVP or Shortest Independent Vector Problem of a lattice. Coming back to MLWE, we have another designation, Learning With Error other Module Lattices. Module lattices can be considered as lattices between those used in the LWE problem definitions and those used for the R-LWE problem (Ring LWE). The Module-LWE offers a compromise between the two extremes of the LWE and the R-LWE. Thereafter we will focus on the CRYSTALS cryptosystem, a cryptosystem based on the MLWE. Our choice to study this cryptosystem is based on the fact that it was submitted to the NIST (National Institute of Standards and Technology) project of post-quantum cryptosystem standardization. CRYSTALS was able to progress to step 3 of the project. But above all, it is the only one among the few that have progressed to this final stage, to have both a KEM type cryptosystem (Key Encapsulation Mechanism) and another signature type one.

1 Le MLWE

La plupart des schémas cryptographiques basés sur les lattices sont construits sur la difficulté supposée des Short Integer Solution (SIS) et du Leaning With Error (LWE). Leur efficacité peut être améliorée en déplaçant les hypothèses de difficulté vers les problèmes plus compacts que constituent les Ring-SIS et Ring-LWE. Cependant, ce changement d'hypothèses de difficulté s'accompagne d'un affaiblissement de la sécurité : SIS et LWE sont connus pour être au moins aussi difficiles que les problèmes standard (dans le cas le plus défavorable) sur les lattices euclidiennes (version classique des lattices),

alors que Ring-SIS et Ring-LWE ont le même niveau de difficulté uniquement pour leurs restrictions à des classes spéciales d'ideal lattices, correspondant aux ideal de certains rings polynomiaux. [7]

Le présent sous-titre consiste en la définition des problèmes Module-SIS et, surtout, Module-LWE, qui relient respectivement SIS et Ring-SIS, et LWE avec Ring-LWE. Ces problèmes de cas moyen (average case, en anglais pour rappel) sont au moins aussi difficiles que les problèmes de lattices standards limités aux module lattices (ces derniers, les module lattices, à leur tour, servent de juste milieu entre les lattices arbitraires et les ideal lattices). Il est important de noter que les réductions allant du cas le plus défavorable au cas moyen pour les problèmes de module sont fortes, en ce sens qu'il existe des réductions inverses. Cette propriété n'est pas connue pour tenir dans le contexte de Ring-SIS / Ring-LWE : des problèmes d'ideal lattices pourraient se révéler faciles sans affecter la difficulté de Ring-SIS / Ring-LWE.

1.1 Variantes du LWE

1.1.1 LWE

Soit $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ dénotant le segment $[0,1)$ avec l'opération addition modulo 1. Pour une fonction de densité de probabilité χ sur $\mathbb{T}$ et un vecteur $s \in \mathbb{Z}_q^n$, nous posons $A_{s,\chi}$ la distribution sur $\mathbb{Z}_q^n \times \mathbb{T}$ obtenu en choisissant un vecteur $a \in \mathbb{Z}_q^n$ uniformément et d'une manière aléatoire, puis $e \in \mathbb{T}$ par respect à χ, et retournant $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Définition 01 :

La version recherche du problème Learning With Error $\mathrm{SLWE}_{q,\chi}$ est comme suit : soit $s \in \mathbb{Z}_q^n$ un secret ; étant donnés plusieurs échantillons à partir de $A_{s,\chi}$, le but est de trouver s.

La version décision du problème Learning With Error $\mathrm{LWE}_{q,\chi}$ est comme suit : soit $s \in \mathbb{Z}_q^n$ choisi d'une manière uniforme et aléatoire ; le but est de faire la distinction entre plusieurs échantillons indépendants à partir de $A_{s,\chi}$ et le même nombre d'échantillons indépendants à partir de $U(\mathbb{Z}_q^n \times \mathbb{T})$. Où U fait référence à une distribution uniforme, sur l'ensemble $\mathbb{Z}_q^n \times \mathbb{T}$ dans le cas présent donc.

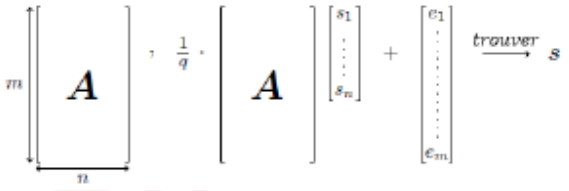


Figure 01 : *Le LWE en termes d'algèbre linéaire.*

Il est également possible d'interpréter le LWE en termes d'algèbre linéaire : Supposons que le nombre d'échantillons requis $\left(a_i, \frac{1}{q}\langle a_i, s\rangle + e_i\right)$ à partir de $A_{s,\chi}$ est m, alors nous considérons la matrice $A \in \mathbb{Z}_q^{m \times n}$ dont les lignes sont les a_i. Et

l'on considèrera le vecteur $e = (e_1, \dots, e_m)^T$. Alors le SLWE est comme nous le montre la **Figure 01**. [3], [4]

Théorème 01 :

Soit $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ de telle sorte que $\alpha q \geq 2\sqrt{n}$. Il existe une réduction quantique de la résolution du $\mathrm{GIVP}^{\eta_\varepsilon}_{\sqrt{8}n/\alpha}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\mathrm{SLWE}_{q,D_\alpha}$ en un temps polynomial avec une probabilité non négligeable.

En assumant que q est premier, $q \leq \mathrm{poly}(n)$, et que χ soit une fonction de densité de probabilité sur $\mathbb{T}$. Alors il existe une réduction en un temps polynomial du $\mathrm{SLWE}_{q,\chi}$ au $\mathrm{LWE}_{q,\chi}$.

Le théorème qui va suivre permet d'outrepasser la condition que q doit être premier.

Théorème 02 :

Soit $\alpha > 0, \varepsilon \in (0, 1/2), m \geq n \geq 1, p \geq 25$ et $q \in [p, 2p)$. Il existe une réduction à temps polynomial du $\mathrm{LWE}_{q,\alpha}$ au $\mathrm{LWE}_{p,\Psi_{\leq\beta}}$ où $\beta = C\alpha\sqrt{n}\sqrt{\log(n/\varepsilon)\log(nm/\varepsilon)}$, pour une constante positive C, et qui perd tout au plus ε en avantage.

1.1.2 R-LWE

Soit ψ une distribution sur $\mathbb{T}_{R^\vee} = K_{\mathbb{R}}/R^\vee$ et $s \in R_q^\vee$. Nous dénotons par $A^{(R)}_{s,\psi}$ la distribution sur $R_q \times \mathbb{T}_{R^\vee}$ obtenu en choisissant $a \in R_q$ d'une manière uniformément aléatoire et $e \in \mathbb{T}_{R^\vee}$ par respect à ψ, et retournant $(a, (a.s)/q + e)$.

Définition 02 :

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Ring Learning With Error $\mathrm{R\text{-}SLWE}_{q,\Psi}$ est comme suit : soit $s \in R_q^\vee$ secret, et $\psi \in \Psi$; étant donné plusieurs échantillons à partir de $A^{(R)}_{s,\psi}$, le but est de trouver s.

Soit Υ une distribution sur une famille de distribution de bruits sur $\mathbb{T}_{R^\vee}$. La version décision du problème Ring Learning With Error $\mathrm{R\text{-}SLWE}_{q,\Upsilon}$ est comme suit : supposons que $s \in R_q^\vee$ soit uniformément aléatoire, et ψ échantillonné à partir de Υ ; le but est de distinguer entre plusieurs échantillons indépendants partir de $A^{(R)}_{s,\psi}$ et le même nombre d'échantillons indépendants à partir de $U(R_q, \mathbb{T}_{R^\vee})$.

Théorème 03 :

Soit $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > \omega(\sqrt{\log n})$. Il existe une réduction quantique de la résolution du $\mathrm{Id\text{-}GIVP}^{\eta_\varepsilon}_{\gamma}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\mathrm{R\text{-}SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec une probabilité non négligeable avec $\gamma = \sqrt{n}.\omega(\sqrt{\log n})/\alpha$.

MADA-ETI

En supposant que q est premier, $q \leq \text{poly}(n)$, et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial de R-SLWE$_{q,\Psi_{\leq\alpha}}$ au R-LWE$_{q,\Upsilon_\alpha}$.

1.1.3 LWE over module

Le M-LWE généralise le LWE et le R-LWE. La variable n et d dénotent respectivement la dimension de R et le rang du module $M \subseteq R^d$. Nous posons $N = nd$ la dimension de la module lattice.

Soit ψ une probabilité de distribution sur $\mathbb{T}_{R^\vee}$ et $s \in \left(R_q^\vee\right)^d$ un vecteur. Nous définissons par $A_{q,s,\psi}^{(M)}$ la distribution sur $\left(R_q\right)^d \times \mathbb{T}_{R^\vee}$ obtenu en choisissant un vecteur $a \in \left(R_q\right)^d$ uniformément aléatoire, et $e \in \mathbb{T}_{R^\vee}$ par respect à ψ, et retournant $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Définition 03 :

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Module Learning With Error M-SLWE$_{q,\Psi}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ un vecteur secret et $\psi \in \Psi$; étant donné arbitrairement plusieurs échantillons à partir de $A_{q,s,\psi}^{(M)}$, le but est de trouver s.

Pour un entier $q \geq 2$ et une distribution Υ sur une famille de distribution sur $K_{\mathbb{R}}$. La version décision du problème Module Learning With Error M-SLWE$_{q,\Upsilon}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ uniformément aléatoire et ψ échantillonné sur Υ ; le but est de distinguer entre arbitrairement plusieurs échantillons aléatoire à partir de $A_{q,s,\psi}^{(M)}$ et le même nombre d'échantillons à partir de $U\left(R_q^d \times \mathbb{T}_{R^\vee}\right)$.

Comme c'était le cas du LWE et du R-LWE, le problème M-LWE peut être interprété en termes d'algèbre linéaire. Il (le M-LWE) consiste à prendre la matrice A du LWE de la forme :

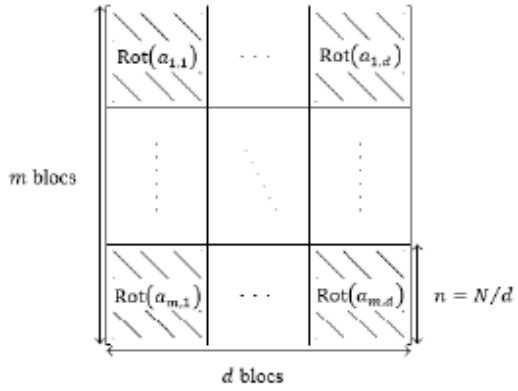


Figure 02 : *Le problème M-LWE peut être interprété en termes d'algèbre linéaire.*

La difficulté du M-LWE est soutenue par les deux théorèmes qui vont suivre.

Théorème 04 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > 2\sqrt{d}.\,\omega\left(\sqrt{\log n}\right)$. Il y a une réduction quantique de la résolution du Mod-GIVP$_\gamma^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du M-

$\mathrm{SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec un avantage non négligeable avec $\gamma = \sqrt{8Nd}.\omega(\sqrt{\log n})/\alpha$.

En assumant que q est premier, $q \leq \mathrm{poly}(N)$ et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial depuis $\mathrm{M\text{-}SLWE}_{q,\Psi_{\leq\alpha}}$ vers $\mathrm{M\text{-}LWE}_{q,\Upsilon_\alpha}$.

Théorème 05 :

Soit $p, q \in \left[2, 2^{N^{O(1)}}\right]$ et $\alpha, \beta \in (0,1)$ de telle sorte que $\beta \geq \alpha.\max\left(1, \frac{q}{p}\right).n^{1/4}N^{1/2}.\omega(\log^2 N)$ et $\alpha q \geq \omega\sqrt{\log(N)/n}$. Il existe une réduction à temps polynomial de $\mathrm{M\text{-}LWE}_{q,\Upsilon_\alpha}$ vers $\mathrm{M\text{-}LWE}_{q,\Upsilon_\beta}$.

$$\mathrm{Mod\text{-}GIVP}^{\eta_\varepsilon}_{\sqrt{8Nd}\cdot\omega(\sqrt{\log n})/\alpha} \xrightarrow{\textbf{Lemme 01}} \mathrm{M\text{-}DGS}^{\eta_\varepsilon}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha} \xrightarrow{\textbf{Lemme 02}} \mathrm{M\text{-}SLWE}_{q,\Psi_{\leq\alpha}}$$

Lemme 01 :

Pour tout $\varepsilon = \varepsilon(N) \leq 1/10$ et tout γ et ϕ de telle sorte que $\gamma \cdot \phi(M) \leq \sqrt{2}\eta_\varepsilon(M)$, il existe une réduction à temps polynomial du $\mathrm{Mod\text{-}GIVP}^{\phi}_{2\sqrt{N}\cdot\gamma}$ au $\mathrm{M\text{-}DGS}^{\phi}_{\gamma}$. [3]

Lemme 02 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et q un entier de telle sorte que $\alpha q \geq 2\sqrt{d} \cdot \omega(\sqrt{\log n})$. En assumant que nous avons accès à un oracle qui résout $\mathrm{M\text{-}SLWE}_{q,\Psi_{\leq\alpha}}$, étant donnés un nombre d'échantillons polynomial. Alors il existe un algorithme quantique à temps polynomial pour $\mathrm{M\text{-}DGS}^{\eta_\varepsilon}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha}$.

1.2 Réduction du Mod-GIVP au M-SLWE

La réduction du Mod-GIVP au M-SLWE utilise le problème intermédiaire qui va suivre, où ϕ dénote une fonction à valeur réelle sur une lattice et γ dépend de la dimension, appelé *Module Discrete Gaussian Sampling Problem* $\mathrm{M\text{-}DGS}^{\phi}_{\gamma}$: étant donné une module lattice M de dimension N, et un nombre $r > \gamma \cdot \phi$, le but est de retourner un échantillon à partir de $D_{M,r}$. La réduction se fait en 2 (deux) étapes :

Lemme 03 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$. Assumons que nous avons accès à un oracle qui résout $\mathrm{M\text{-}SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec une probabilité non négligeable. Alors, il existe un algorithme quantique à temps polynomial qui, étant donnée une module lattice M de dimension N, un nombre $r > 2q\eta_\varepsilon(M)$ et $\mathrm{poly}(N)$ échantillons à partir de $D_{M,r}$, produit un échantillon à partir de $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ avec une probabilité non négligeable.

Pour prouver le **Lemme 03** nous utilisons un problème intermédiaire, le $\mathrm{Mod\text{-}BDD}_\delta$ (*BDD* pour *Bounded Distance Decoding*) : étant donné une module lattice M, $\delta < \lambda_1(M)/2$ et tout point

$y \in \mathbb{R}^n$ de la forme $y = x + e$ pour un certain $x \in M$ et $\|e\|_{2,\infty} \leq \delta$, trouver x. Il est à noter que la norme $\ell_{2,\infty}$ est utilisée ici, parce que cela nous aidera à énoncer le **Lemme 06** plus convenablement.

Un autre problème intermédiaire q-Mod-BDD$_\delta$ sera utilisé : étant donné une module lattice M et un point $y \in \mathbb{R}^n$ dans la distance (par respect à la norme $\ell_{2,\infty}$) δ de M, donner en sortie le coset dans M/qM du vecteur le plus proche y. La démonstration du **Lemme 03** consiste en la séquence de réduction ci-dessous (il est à noter que δ ici prend la valeur $\frac{\alpha q \cdot \omega(\sqrt{\log n})}{\sqrt{2n}r}$) :

Echantillons à partir de $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ $\xrightarrow[\text{(quantique)}]{\textbf{Lemme 04}}$ Mod-BDD$_{M^\vee,\delta}$ $\xrightarrow{\textbf{Lemme 05}}$ q-Mod-BDD$_{M^\vee,\delta}$ $\xrightarrow{\textbf{Lemme 06}}$ M-SLWE$_{q,D_\alpha}$ Echantillon à partir de $D_{M,r}$

Lemme 04 :

Il existe un algorithme quantique efficace qui, étant donnée une module lattice M de dimension N, un nombre $\delta < \lambda_1(M^\vee)\omega(\sqrt{\log n})/(2\sqrt{n})$, et un oracle qui résout Mod-BDD$_\delta$ sur $M^\vee$, donne à la sortie des échantillons à partir de $D_{M,r\sqrt{d}\cdot\omega(\sqrt{\log n})/(\sqrt{2}\delta)}$.

Lemme 05 :

Pour tout $q \geq 2$, il existe une réduction à temps polynomial du Mod-BDD$_\delta$ au q-Mod-BDD$_\delta$.

Lemme 06 :

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$. Soit $M \subseteq R^d$ un R-module, et $r > \sqrt{2}q \cdot \eta_\varepsilon(M)$. Etant donné un accès à un échantillonneur oracle à partir de la distribution $D_{M,r}$, il existe une réduction probabiliste du q-Mod-BDD$_{M^\vee,\frac{\alpha q\cdot\omega(\sqrt{\log n})}{\sqrt{2n}r}}$ au M-SLWE$_{q,\Psi_{\leq\alpha}}$.

Le principe de la réduction est de construire à partir de y, l'entrée de q-Mod-BDD, et à partir de quelques échantillons Gaussiens continus ou discret, les pairs (a, b) distribués selon $A^{(M)}_{q,s,\psi}$, où s va directement dépendre du vecteur x le plus proche de y. Pour produire de tels échantillons (a, b) suivant la distribution désirée, nous combinons les preuves correspondants pour LWE et R-LWE. Puis d'appeler l'oracle du M-SLWE pour retourner s et nous permettre de recouvrir l'information sur x.

Lemme 07 :

Soit $\varepsilon > 0$ et $s = \Theta(x \bmod qM^\vee)$. Il existe $\psi \in \Psi_{\leq\alpha}$ de telle sorte que la distance statistique entre

$A^{(M)}_{q,s,\psi}$ et la distribution de (a, b) est tout au plus 6ε.

Tout ceci conclue la démonstration de la première partie de notre **Théorème 04**.

Pour la seconde partie du théorème (existence d'une réduction du M-SLWE au M-LWE), nous invitons nos lecteurs à consulter la démonstration dans notre bibliographie (paragraphe 4.3 dans [3]).

2 Le cryptosystème CRYSTALS

Le cryptosystème *CRYSTALS* ou *Cryptographic Suite for Algebraic Lattices* encapsule deux types de cryptosystèmes : *Kyber*, un KEM (Key Encapsulation Mechanism, pour rappel) ayant la spécificité d'être classifié sécurisé IND-CCA2 (nous verrons la définition de ce terme un peu plus loin dans notre rédaction), et *Dilithium* un algorithme de signature sécurisé EUF-CMA (même cas que l'IND-CCA2, une définition sera donnée plus loin). Les deux algorithmes sont basés sur les problèmes difficiles relatifs aux modules lattices, ont été par conséquent construits pour résister aux éventuels attaques d'ordinateurs quantiques.

Le CRYSTALS, comme mentionnés dans l'introduction du présent ouvrage, a été soumis au projet NIST de cryptographie post-quantique. Il a pu progresser jusqu'à l'étape 3 dudit projet de standardisation. Mais surtout, le seul parmi le quelques-uns qui ont progresser à cette étape finale, à posséder à la fois un cryptosystème type KEM et signature. [5], [6], [8]

2.1 CRYSTALS-Kyber, cryptosystèmetype KEM

2.1.1 Analyses de performance

Dans cette section, nous examinons les aspects de mise en œuvre de Kyber et rapportons les résultats de performance de deux implémentations : l'implémentation de référence ANSI C (ANSI pour *American National Standards Institute*) et une implémentation optimisée à l'aide d'instructions vectorielles AVX2.

2.1.1.1 Performances de Kyber sur les processeurs Intel Haswell

Le **Tableau 01** présente les résultats de performance d'Intel Haswell de l'implémentation de référence et d'une implémentation optimisée AVX2 de Kyber et de la variante 90s de Kyber ainsi que les tailles des clés et des textes chiffrés. Tous les benchmarks ont été obtenus sur un cœur d'un processeur Intel Core i7-4770K (Haswell) cadencé à 3492 MHz (comme indiqué par /proc/cpuinfo) avec TurboBoost et l'hyperthreading désactivés. La machine utilisée a 32 Go de RAM (pour *Random Access Memory*)et exécute Debian GNU/Linux (GNU de *GNU's Not Unix*) avec la version 4.19.0 du

noyau Linux. Les deux implémentations ont été compilées avec gcc version 8.3.0. Tous les décomptes de cycles indiqués sont la médiane des décomptes de cycles de 10000 exécutions de la fonction respective. Les implémentations ne sont pas optimisées pour l'utilisation de la mémoire, mais en général, Kyber n'a que des besoins en mémoire très modestes.

En addition, *ref* fait référence à l'implémentation de référence C, *AVX2* à l'implémentation utilisant des instructions vectorielles AVX2 ; sk signifie clé secrète, pk pour clé publique et ct pour texte chiffré. Entre parenthèses se trouvent des valeurs approximatives lors de l'inclusion de la génération de clé dans la décapsulation pour éviter d'avoir à stocker des clés secrètes développées. Dans ce scénario, nous stockons uniquement la graine initiale d au tout début de l'algorithme de génération de clé Kyber.CPAPKE.KeyGen(). Les nombres de cycles approximatifs pour ce scénario sont calculés comme la somme des nombres de cycles pour la décapsulation standard et la génération de clé moins le nombre de cycles nécessaires pour générer la matrice **A** de la graine publique ρ. Notez qu'il s'agit d'une estimation très prudente ; les mises en œuvre réelles de l'approche peuvent également économiser, par exemple, l'échantillonnage des 32 octets de caractère aléatoire.

Kyber512					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	1632 (ou 32)	gen :	122684	gen :	33856
pk :	800	enc :	154524	enc :	45200
ct :	768	dec :	187960 (ou ≈ 288912)	dec :	34572 (ou ≈ 59088)
Kyber512 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	1632 (ou 32)	gen :	213156	gen :	21880
pk :	800	enc :	249084	enc :	28592
ct :	768	dec :	277612 (ou ≈ 405268)	dec :	20980 (ou ≈ 38752)
Kyber768					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	2400 (ou 32)	gen :	199408	gen :	52732
pk :	1184	enc :	235260	enc :	67624
ct :	1088	dec :	274900 (ou ≈ 425492)	dec :	53156 (ou ≈ 82220)
Kyber768 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	2400 (ou 32)	gen :	389760	gen :	30460
pk :	1184	enc :	432764	enc :	40140

MADA-ETI

ct :	1088	dec :	473984 (ou ≈ 671864)	dec :	301108 (ou ≈ 52512)
Kyber1024					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	3168 (ou 32)	gen :	307148	gen :	73544
pk :	1568	enc :	346648	enc :	97324
ct :	1568	dec :	396584 (ou ≈ 617848)	dec :	79128 (ou ≈ 115332)
Kyber1024 – années 90					
Tailles (en Octets)		Cycles Hashwell (ref)		Cycles Hashwell (AVX2)	
sk :	3168 (ou 32)	gen :	636380	gen :	43212
pk :	1568	enc :	672644	enc :	56566
ct :	1568	dec :	724144 (ou ≈ 1009448)	dec :	44328 (ou ≈ 71180)

Tableau 01 : *Performances de Kyber sur les processeurs Intel Haswell.*

2.1.1.2 Performance of Kyber on ARM Cortex-M4 CPUs

Le **Tableau 02** rapporte le nombre de cycles et l'utilisation de la RAM d'une implémentation C (propre) et d'une implémentation optimisée (m4) de Kyber sur un ARM Cortex-M4. Tous les benchmarks sont obtenus en utilisant le framework pqm4 sur une carte Discovery STM32F407. Nous ne rapportons pas ici les benchmarks de la version des années 90 de Kyber sur le Cortex-M4, car pqm4 n'inclut pas encore d'implémentation en temps constant d'AES.

Kyber512				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	655595	6020	463068	2844
enc :	865256	8668	561518	2484
dec :	961648	9444	519237	2508
Kyber768				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1087897	10052	756224	3292
enc :	1373744	13212	915676	2980
dec :	1491214	14308	853001	3004
Kyber1024				
	M4 Cycles (propre)	M4 RAM (propre)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1696314	15180	1213303	3804
enc :	2057522	18844	1407769	3492
dec :	2199958	20420	1326409	3516

Tableau 02 : *Performances de Kyber sur les processeurs Intel Haswell.*

2.1.2 Estimation de niveau de sécurité

Dans notre présent paragraphe, nous nous sommes principalement appuyé sur nos bibliographies [1], [10] et [11].

Le **Tableau 03** répertorie les niveaux de sécurité selon la définition de la section 4.A.5 de l'appel à propositions du NIST [11] pour les différents ensembles de paramètres de Kyber. Nos affirmations sont basées sur les estimations de coût des attaques les plus connues contre le problème MLWE sous-jacent à Kyber, comme détaillé dans la sous-section 5.1 de notre bibliographie [1]. Plus précisément, y est listée, la difficulté classique et quantique du noyau-SVP et les utilisons pour dériver les niveaux de sécurité.

	Kyber512	Kyber768	Kyber1024
Niveau de sécurité NIST	1	3	5
Méthodologie noyau-SVP, attaque primitive uniquement			
Attaque lattice dim. d	1003	1424	1885
BKZ-blocksize β (BKZ pour Block Korkine Zolotarev)	403	625	877
Noyau-SVP difficulté classique	118	182	256
Noyau-SVP difficulté quantique	107	165	232
Estimation affinée pour les attaques classiques			
Attaque lattice dim. d	1025	1467	1918
BKZ-blocksize β	413	637	894
Tamisage dimension $\beta' = \beta - d_{4f}$	375	586	829
$\log_2$(gates)	151.5	215.1	287.3
$\log_2$(memory in bits)	93.8	138.5	189.7

Tableau 03 : *Difficulté classique et quantique des différents ensembles de paramètres proposés pour Kyber.*

MADA-ETI

Tous les jeux de paramètres de Kyber ont une certaine probabilité d'échec de décryptage. Ces échecs sont un problème de sécurité (voir Section 5.5 de [1]) et donc les probabilités avec lesquelles ils se produisent doivent être faibles. Mais comme dans le ROM classique, la probabilité d'échec de décryptage est théorique de l'information, nous ne voyons pas la nécessité de la diminuer avec le paramètre de sécurité. En particulier, l'échec de décryptage pour nos ensembles de paramètres de niveau 3 et 5 est inférieur à 2^{-160}, ce qui signifie que si 280 instances de Kyber étaient exécutées toutes les

secondes à partir de maintenant jusqu'à ce que notre soleil devienne une naine blanche, les chances sont toujours fortement favorables à ce qu'il n'y ait jamais d'échec de décryptage. Nous excluons donc ces attaques de nos affirmations concernant les estimations de sécurité du NIST.

L'impact du bruit déterministe causé par $\mathrm{Compress}_q$ *sur Kyber512.* Chaque coefficient de $\mathbf{e_1}$ (et $\mathbf{e_2}$) dans l'algorithme de cryptage $\mathrm{Kyber.CPAPKE}$ se comporte comme une distribution binomiale avec le paramètre $\eta_2 = 2$, avec une variance $\eta_2/2 = 1$. Le paramètre $d_u = 10$ implique que la fonction $\mathrm{Compress}_q$ mappe les éléments modulo q à un ensemble de taille 2^{10} où la différence entre tous les pairs d'éléments de ce dudit ensemble est de 3 ou 4. Cela implique que l'erreur ajoutée par la fonction $\mathrm{Compress}_q$ pour chaque coefficient est soit uniforme sur $\{-1,0,1\}$, $\{-1,0,1,2\}$ ou $\{-2,-1,0,1\}$. Elle a donc une variance au moins aussi grande que la distribution uniforme sur l'ensemble $\{-1,0,1\}$, soit $2/3$. Cela rend la variance totale de chaque coefficient de $\mathbf{e_1}$ plus l'erreur déterministe au moins $1 + 2/3 = 5/3$. Les autres termes secret et bruit ont des distributions binomiales avec le paramètre $\eta_1 = 3$ pour une variance de $\eta_1/2 = 3/2 < 5/3$. En tenant compte des erreurs ajoutées par $\mathrm{Compress}_q$, nous calculons donc la difficulté de Kyber512 en supposant que la variance de chaque coefficient secret/erreur est $3/2$.

L'impact de MAXDEPTH. Les accélérations quantiques les plus connues pour l'algorithme de tamisage, qui sont plus détaillés dans l'analyse des coûts (se référer à la sous-section 5.1.1 de [1]), ne sont que légèrement affectées par la limitation de la profondeur d'un circuit quantique, car elle utilise la recherche de Grover sur des ensembles de petite taille (comparé pour rechercher dans tout l'espace de clés d'AES). Pour que les estimations de fonctionnement de la difficulté noyau-SVP correspondent au coût de la porte quantique de la rupture d'AES aux niveaux de sécurité respectifs, un ordinateur quantique devrait prendre en charge une profondeur maximale de 70 à 80. Lors de la limitation de la profondeur maximale à des valeurs plus petites, ou lors de l'examen d'attaques classiques, les estimations de la difficulté noyau-SVP sont plus petites que le nombre de portes pour les attaques contre AES. Une étude récente sur le coût concret du tamisage suggère que les accélérations quantiques de ces algorithmes sont ténues, indépendamment de la valeur de MAXDEPTH.

MADA-ETI

2.2 *CRYSTALS-Dilithium, the signature type cryptosystem*

2.2.1 Sécurité

La sécurité de notre algorithme de signature peut être prouvée par le biais du ROM, sur la base de la difficulté de deux problèmes [2], [9]. Le premier est le problème standard LWE (sur des anneaux polynomiaux) qui demande de distinguer $(\mathbf{A}, \mathbf{t} := \mathbf{A}\mathbf{s_1} + \mathbf{s_2})$ de $(\mathbf{A}, \mathbf{u})$, où $\mathbf{u}$ est uniformément aléatoire. L'autre problème est ce qu'on a appelé le problème SelfTargetMSIS, qui est le problème de trouver un vecteur $\begin{bmatrix}\mathbf{z}\\ c\\ \mathbf{v}\end{bmatrix}$ avec de petits coefficients et un message μ satisfaisant :

$$\mathrm{H}\left(\mu \| [\mathbf{A}|\boldsymbol{t}|\mathbf{I}] \cdot \begin{bmatrix}\mathbf{z}\\ c\\ \mathbf{v}\end{bmatrix}\right) = c \qquad (01)$$

où $\mathbf{A}$ et t sont uniformément aléatoires et $\mathbf{I}$ est la matrice d'identité. Dans le ROM, on peut obtenir une réduction (non étroite) en utilisant le *lemme d'embranchement* (*forking lemma* en anglais) du problème MSIS standard de trouver un z_0 avec de petits coefficients satisfaisant $\mathbf{A}z_0 = 0$ au SelfTargetMSIS. On peut suivre cette approche exacte pour prouver que Dilithium est sécurisé dans le ROM en fonction de la difficulté du MLWE et SIS.

Niveau de sécurité NIST	2	3	5
Taille de sortie			
taille de la clé publique (bytes)	1312	1952	2592
taille de la signature (bytes)	2420	3293	4595
Difficulté du LWE (Core-SVP, et pour la version améliorée également)			
taille de bloc BKZ b (GSA)	423	624	863
Core-SVP classique	123	182	252
Core-SVP quantum	112	165	229
taille de bloc BKZ b (simulation)	433	624	863
$\log_2$ portes classiques	159	217	285
$\log_2$ mémoires classiques	98	139	187
Difficulté SIS (Core-SVP)			
taille de bloc BKZ b	423 (417)	638 (602)	909 (868)
Core-SVP Classique	123 (121)	186 (176)	265 (253)
Core-SVP quantum	112 (110)	169 (159)	241 (230)

MADA-ETI

Performance (Code de référence non optimisé, Skylake)			
Gen cycle médian	300,751	544,232	819,475
Sign cycle médian	1,081,174	1,713,783	2,383,399
Sign cycle moyen	1,355,434	2,348,703	2,856,803
Verify cycle médian	327,362	522,267	871,609
Performance (AVX2, Skylake)			
Gen cycle médian	124, 031	256, 403	298, 050
Sign cycle médian	259, 172	428, 587	538, 986
Sign cycle moyen	333, 013	529, 106	642, 192
Verify cycle médian	118, 412	179, 424	279, 936
Performance (AVX2+AES, Skylake)			
Gen cycle médian	70, 548	153, 856	153, 936
Sign cycle médian	194, 892	296, 201	344, 578
Sign cycle moyen	251, 144	366, 470	418, 157
Verify cycle médian	72, 633	102, 396	151, 066

Tableau 04 : *Tailles de sortie, sécurité et performances de Dilithium.*

Dans le QROM, où l'adversaire peut interroger H en superposition, la situation est un peu différente. Il a été montré dans que Dilithium est toujours basé sur MLWE et SelfTargetMSIS dans le QROM, même avec une réduction étroite lorsque l'algorithme est déterministe. Mais on ne peut plus utiliser directement le lemme d'embranchement (puisqu'il s'agit d'un type de rembobinage) pour donner une réduction quantique de MSIS à SelfTargetMSIS. Il y a encore de bonnes raisons de croire que le problème SelfTargetMSIS, et donc Dilithium, est sécurisé dans le QROM. Premièrement, il n'y a pas de schémas de signature naturelle construits à partir de protocoles Σ utilisant la transformation Fiat-Shamir qui sont sécurisés dans le ROM mais pas dans le QROM. En outre, il est possible de définir les paramètres de Dilithium (tout en laissant la conception de l'algorithme inchangée) afin que le problème SelfTargetMSIS devienne théoriquement difficile, laissant ainsi cette version de Dilithium sécurisée dans le QROM basé uniquement sur MLWE. Une instanciation de ces paramètres aboutit [2] à un algorithme avec des signatures et des clés publiques qui sont respectivement 2x et 5x plus grandes. Bien que nous ne considérions pas cela comme un bon compromis, l'existence d'un tel système nous

MADA-ETI

donne une confiance supplémentaire dans la sécurité d'une version optimisée de Dilithium.

Récemment, de nouveaux travaux ont réduit encore plus l'écart entre la sécurité dans le ROM et le QROM. Ces travaux [2] ont montré que si le protocole Σ sous-jacent s'effondre et a une solidité particulière, alors sa transformation Fiat-Shamir est une signature sécurisée dans le QROM. La solidité particulière du protocole Dilithium est directement impliquée par la difficulté du MSIS. De plus, lesdits travaux, toujours, énonce un probable effondrement de protocole Σ de Dilithium. Encore dans ces travaux est montré que la propriété d'effondrement a une réduction par rapport à MLWE. La réduction est plutôt non étroite, mais elle donne encore plus d'affirmation qu'il n'y a rien d'incertain dans la construction de Dilithium ou de tout algorithme naturel construit via le cadre Fiat-Shamir dont la sécurité peut être prouvée dans le ROM. À notre avis, il est certain que la distinction entre les signatures sécurisées dans le ROM et QROM sera bientôt traitée de la même manière que la distinction entre les schémas sécurisés dans le modèle standard et le ROM - il y aura des différences théoriques, mais la sécurité dans la pratique sera la même.

Dans le **Tableau 04**, les formules pour les tailles de la clé publique et de la signature sont dans notre bibliographie [2]. Les chiffres entre parenthèses pour la difficulté SIS correspondent à la version fortement infalsifiable (c'est-à-dire qu'il est difficile de trouver une deuxième signature distincte pour une paire message/signature déjà vue) du schéma de signature. Pour la longueur du coefficient correspondant à la version sans parenthèse, nous renvoyons toujours nos lecteurs à notre bibliographie [2] pour plus de détails. En raison de l'échantillonnage de rejet, il y a une différence notable entre les temps de signature médians et moyens, et nous mesurons donc les deux.

2.2.2 Nombre d'itérations

Nous aimerions maintenant calculer la probabilité que l'étape 21, de l'algorithme détaillé de signature de Dilithium, assigne $(\mathbf{z}, \mathbf{h})$ à $\perp$. La probabilité que $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ peut être calculée en considérant chaque coefficient séparément. Pour chaque coefficient σ de $c\mathbf{s}_1$ le coefficient correspondant de $\mathbf{z}$ sera entre $-\gamma_1 + \beta + 1$ et $-\gamma_1 + \beta - 1$ (inclusivement) tant que le coefficient correspondant de $\mathbf{y}_i$ est entre $-\gamma_1 + \beta + 1 - \sigma$ et $-\gamma_1 + \beta - 1 - \sigma$. Le poids de cette fourchette est $2(\gamma_1 - \beta) - 1$ et la coefficient de $\mathbf{y}$ a $2\gamma_1 - 1$ possibilités. Ainsi la probabilité pour que tout coefficient de $\mathbf{y}$ soit dans la bonne fourchette est de :

MADA-ETI

$$\left(\frac{2(\gamma_1-\beta)-1}{2\gamma_1-1}\right)^{256\cdot l} = \left(1-\frac{\beta}{\gamma_1-1/2}\right)^{ln} \approx e^{-256\cdot\beta l/\gamma_1} \quad (02)$$

où nous utilisons le fait que nos valeurs de γ_1 sont plus grandes comparées à $1/2$.

Nous nous attaquons par la suite au calcul de la probabilité que nous avons :

$$\|\mathbf{r}_0\|_\infty = \|\text{LowBits}_q(\mathbf{w}-c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2-\beta$$

Si nous assumons (heuristiquement) que les bits d'ordre inférieur sont uniformément distribués modulo $2\gamma_2$, alors il y a :

$$\left(\frac{2(\gamma_1-\beta)-1}{2\gamma_1-1}\right)^{256\cdot k} \approx e^{-256\cdot\beta k/\gamma_2}$$

probabilité que tous les coefficients soient dans la bonne fourchette (en utilisant le fait que nos valeurs de β sont grandes par rapport à $1/2$). Par conséquent, la probabilité que l'étape 21 réussisse est :

$$\approx e^{-256\cdot\beta(l/\gamma_1+k/\gamma_2)} \quad (03)$$

Il est plus difficile de calculer formellement la probabilité que l'étape 24 aboutisse à un redémarrage. Les paramètres ont été fixés de telle sorte qu'heuristiquement $(\mathbf{z},\mathbf{h}) = \perp$ avec une probabilité comprise entre 1% et 2%. Par conséquent, la grande majorité des répétitions de la boucle seront causées par l'étape 21.

Nous tenons à souligner que le nombre d'itérations prévu est indépendant de la clé secrète $\mathbf{s}_1$, $\mathbf{s}_2$ et donc aucune information à leur sujet ne peut être obtenue par une attaque qui compte les itérations.

Niveau de sécurité NIST	2	3	5
Paramètres			
q [module]	8380417	8380417	8380417
d [bits de $\mathbf{t}$ ignorés]	13	13	13
τ [# de bits 1s dans c]	39	49	60
défi en termes d'entropie $\left\lfloor \log\binom{256}{\tau} + \tau \right\rfloor$	192	225	257
γ_1 [fourchette de coefficient de $\mathbf{y}$]	2^{17}	2^{19}	2^{19}
γ_1 [fourchette d'approximation pour l'ordre inférieur]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k,l) [dimension de $\mathbf{A}$]	(4,4)	(6,5)	(8,7)

η [fourchette de la clé secrète]	2	4	2
β [$\tau \cdot \eta$]	78	196	120
ω [# de bits 1s max dans l'indice **h**]	80	55	75
Itérations (depuis l'équation (03))	4.25	5.1	3.85

Tableau 05 : *Jeux de paramètres de Dilithium.*

3 Conclusion

Cet article nous a permis de nous concentrer sur l'objet mathématique qu'est le MLWE (Module Learning With Error) et son application comme base des cryptosystèmes CRYSTALS-Kyber et CRYSTALS-Dilithium.

Nous avons ouvert notre rédaction sur le développement du MLWE.

La section a mis en exergue qu'il existe une réduction (une équivalence en termes de difficulté ou de dureté donc) du Mod-GIVP (Generalized Independent Vector Problem over Module Lattices) et du M-LWE. Pour être plus précis, la réduction a été détaillée pour la réduction de Mod-GIVP à M-SLWE (version Search du LWE sur les treillis de modules). Le lecteur a ensuite été renvoyé à notre référence [3] pour la deuxième partie de la réduction, qui est M-SLWE vers M-LWE.

Nous nous sommes ensuite concentrés principalement sur le système de cryptage CRYSTALS.

Nous avons commencé par celui de type KEM (Key Encapsulation Mechanism), le CRYSTALS-Kyber.

Nous avons avancé la spécification des deux parties qui le composent : Kyber.CPAPKE et Kyber.CCAKEM. Chacune de ces deux parties est à son tour subdivisée en 03 algorithmes : génération de clé, chiffrement et déchiffrement. Notons au passage le fait que les opérations de multiplication dans lesdits algorithmes, sont effectuées dans le domaine NTT (Number Theoretic Transform). La multiplication dans R_q est réalisée de manière efficace (à moindre coût).

En termes de performance, elle est à son maximum pour CRYSTALS-Kyber lorsqu'il est écrit en tenant compte des avantages des processeurs supportant la technologie AVX2 pour les processeurs Intel Hashwell, et la technologie m4 pour les processeurs ARM.

Vient ensuite le cas de CRYSTALS-Dilithium qui entre dans la catégorie des cryptosystèmes type signature.

Outre la notion de NTT, qui est toujours d'actualité, les opérations de hachage

interviennent de manière plus marquée. Nous citerons entre autres le hachage en boule qui est représenté en termes algorithmiques par la fonction SampleInBall(ρ). ρ représentant une graine.

Enfin, l'algorithme réel de CRYSTALS-Dilithium est donné. Il est composé de 03 sous-parties : génération de clé, signature et vérification.

4 Bibliographie

[1]. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, « *CRYSTALS-Kyber - Algorithm Specifications And Supporting Documentation (version 2.0)* », https://pq-crystals.org/kyber/index.shtml, Mar. 2019;

[2]. S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé, « *CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation* », https://pq-crystals.org/dilithium/index.shtml, Mar. 2021;

[3]. A. Langlois and D. Stehle, « *Worst-Case to Average-Case Reductions for Module Lattices* », https://eprint.iacr.org/2012/090, Aug. 2013;

[4]. O. Regev, « *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography* », https cims.nyu.edu ~regev papers qcrypto.pdf, May 2009;

[5]. « *Easy explanation of IND- security notions?* », https://crypto.stackexchange.com/questions/26689/easy-explanation-of-ind-security-notions, Oct. 2020;

[6]. « *EUF-CMA and SUF-CMA* », https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/, Oct. 2020;

[7]. V. Lyubashevsky, C. Peikert and O. Regev, « *On Ideal Lattices and Learning with Errors Over Rings* », https://www.iacr.org/archive/eurocrypt2010/66320288/66320288.pdf, Nov. 2020;

[8]. S. Diop1, B. O. Sané1 and M. Seck1, « *NTRU-LPR IND-CPA A New Ideal Lattice-based Scheme* », https://eprint.iacr.org/2018/109.pdf, Nov. 2020;

[9]. « *pq-crystals/security-estimates* », https://github.com/pq-crystals/security-estimates, Feb. 2021;

[10]. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, « *CRYSTALS-Kyber a CCA-secure module-lattice-based KEM* », https://eprint.iacr.org/2017/634.pdf, Feb. 2021;

[11]. « *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process* », https://csrc.nist.gov/csrc/media/projects/post-

MADA-ETI

quantum-cryptography/documents/call-for-
proposals-final-dec-2016.pdf, Mar. 2021;

MADA-ETI

A1.4 Application of the MLWE in the cryptosystem CRYSTALS – KYBER as KEM and Dilithium as signature

Vol-7 Issue-5 2021 IJARIIE-ISSN(O)-2395-4396

APPLICATION OF THE MLWE IN THE CRYPTOSYSTEM CRYSTALS – KYBER AS KEM AND DILITHIUM AS SIGNATURE

RAZAFIMAHEFA Fenofahasoavana Ombana[1], RANDRIAMITANTSOA Paul August[2], RANDRIANANDRASANA Marie Emile[3]

[1] PhD Student, TASI, ED-STII, Antananarivo, Madagascar
[2] Thesis Director, TASI, ED-STII, Antananarivo, Madagascar
[3] Thesis Codirector, TASI, ED-STII, Antananarivo, Madagascar

ABSTRACT

Quantum computers, but still with a low number of qubits, are starting to emerge as we write. It is more crucial than ever to work on new cryptosystems. Cryptosystems that will be based on mathematical problems that are robust enough to withstand the increased computational power that these new types of computers will provide. In the course of our research, we came across the mathematical object that is the Lattice. The latter is one of the promising candidates in terms of fundamental problem for post-quantum cryptosystems. But the mathematical tool on which we will open this article is the MLWE or Module Learning With Error. The LWE (or Learning With Error), in the first instance, is a generalization of the problem Learning from parity with error. It has been shown that this problem is equivalent, in terms of difficulty, to the SIVP or Shortest Independent Vector Problem of a lattice. Coming back to MLWE, we have another designation, Learning With Error other Module Lattices. Module lattices can be considered as lattices between those used in the LWE problem definitions and those used for the R-LWE problem (Ring LWE). The Module-LWE offers a compromise between the two extremes of the LWE and the R-LWE. Thereafter we will focus on the CRYSTALS cryptosystem, a cryptosystem based on the MLWE. Our choice to study this cryptosystem is based on the fact that it was submitted to the NIST (National Institute of Standards and Technology) project of post-quantum cryptosystem standardization. CRYSTALS was able to progress to step 3 of the project. But above all, it is the only one among the few that have progressed to this final stage, to have both a KEM type cryptosystem (Key Encapsulation Mechanism) and another signature type one.

Keyword: *Quantum Computer, Post Quantum Cryptosystem, Module LWE, CRYSTALS-Kyber, CRYSTALS-Dilithium, Key Encapsulation Mechanism (KEM), Signature.*

1. The MLWE

Most lattice-based cryptographic schemes are built on the assumed difficulty of Short Integer Solution (SIS) and Leaning With Error (LWE). Their effectiveness can be improved by shifting the difficulty assumptions to the more compact Ring-SIS and Ring-LWE problems. However, this change in difficulty assumptions is accompanied by a weakening of security: SIS and LWE are known to be at least as difficult as standard (worst case) problems on Euclidean lattices (classical version of lattices), while Ring-SIS and Ring-LWE have the same level of difficulty only for their restrictions to special classes of ideal lattices, corresponding to the ideal of some polynomial rings. [7]

The present subtitle of our work consists of the definition of Module-SIS and, especially, Module-LWE problems, which link SIS with Ring-SIS, and LWE with Ring-LWE, respectively. These average case problems are at least as difficult as the standard lattice problems restricted to module lattices (the latter, module lattices, in turn, serve as a middle ground between arbitrary lattices and ideal lattices). It is important to note that the worst case to average case reductions for modulus problems are strong, in the sense that there are inverse reductions. This property is not known to hold in the context of Ring-SIS / Ring-LWE: problems with ideal lattices could be easy without affecting the difficulty of Ring-SIS / Ring-LWE.

1.1. LWE variants

1.1.1. LWE

Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ denote the segment [0,1) with the addition operation modulo 1. For a probability density function χ on $\mathbb{T}$ and a vector $s \in \mathbb{Z}_q^n$, we posit $A_{s,\chi}$ the distribution on $\mathbb{Z}_q^n \times \mathbb{T}$ obtained by choosing a vector $a \in \mathbb{Z}_q^n$ uniformly and in a random fashion, then $e \in \mathbb{T}$ by respect to χ, and returning $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Definition 01:

The search version of the Learning With Error problem $\mathrm{SLWE}_{q,\chi}$ is as follows: let $s \in \mathbb{Z}_q^n$ be a secret; given several samples from $A_{s,\chi}$, the goal is to find s.

The decision version of the Learning With Error problem $\mathrm{LWE}_{q,\chi}$ is as follows: let $s \in \mathbb{Z}_q^n$ be chosen in a uniformly random fashion; the goal is to distinguish between several independent samples from $A_{s,\chi}$ and the same number of independent samples from $U(\mathbb{Z}_q^n \times \mathbb{T})$. Where U refers to a uniform distribution, on the set $\mathbb{Z}_q^n \times \mathbb{T}$ in this case therefore.

It is also possible to interpret the LWE in terms of linear algebra: Suppose the number of samples required $\left(a_i, \frac{1}{q}\langle a_i, s\rangle + e_i\right)$ from $A_{s,\chi}$ is m, then we consider the matrix $A \in \mathbb{Z}_q^{m \times n}$ whose rows are the a_i. And we will consider the vector $e = (e_1, \ldots, e_m)^T$. Then the SLWE is as follows: [3], [4]

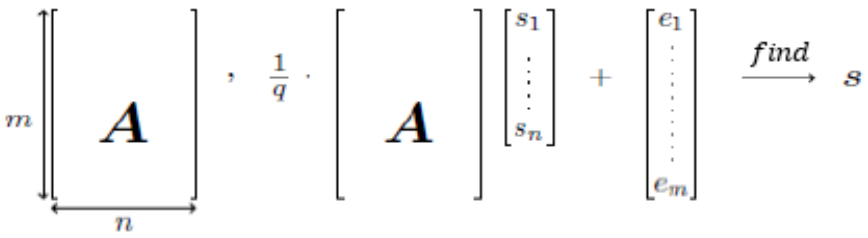


Figure 01: The LWE in terms of linear algebra.

Theorem 01:

Let $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ and $q \geq 2$ such that $\alpha q \geq 2\sqrt{n}$. There is a quantum reduction from solving $\mathrm{GIVP}^{\eta_\varepsilon}_{\sqrt{8}n/\alpha}$ in polynomial time (in the worst case, with high probability) to solving $\mathrm{SLWE}_{q,D_\alpha}$ in polynomial time with nontrivial probability.

Assuming q is prime, $q \leq \mathrm{poly}(n)$, and χ is a probability density function on $\mathbb{T}$. Then there exists a reduction in polynomial time from $\mathrm{SLWE}_{q,\chi}$ to $\mathrm{LWE}_{q,\chi}$.

The following theorem overrides the condition that q must be prime.

Theorem 02:

Let $\alpha > 0$, $\varepsilon \in (0, 1/2)$, $m \geq n \geq 1$, $p \geq 25$ and $q \in [p, 2p)$. There exists a polynomial-time reduction from $\mathrm{LWE}_{q,\alpha}$ to $\mathrm{LWE}_{p,\Psi_{\leq\beta}}$ where $\beta = C\alpha\sqrt{n}\sqrt{\log(n/\varepsilon)\log(nm/\varepsilon)}$, for a positive constant C, and which loses at most ε in advantage.

1.1.2. R-LWE

Let ψ be a distribution on $\mathbb{T}_{R^\vee} = K_{\mathbb{R}}/R^\vee$ and $s \in R_q^\vee$. We denote by $A_{s,\psi}^{(R)}$ the distribution over $R_q \times \mathbb{T}_{R^\vee}$ obtained by choosing $a \in R_q$ in a uniformly random manner and $e \in \mathbb{T}_{R^\vee}$ with respect to ψ, and returning $(a, (a.s)/q + e)$.

Definition 02:

Let $q \geq 2$ and Ψ be a family of distributions on $\mathbb{T}_{R^\vee}$. The search version of the Ring Learning With Error problem R-$\mathrm{SLWE}_{q,\Psi}$ is as follows: let $s \in R_q^\vee$ be secret, and $\psi \in \Psi$; given several samples from $A_{s,\psi}^{(R)}$, the goal is to find s.

Let Υ be a distribution over a family of noise distributions on $\mathbb{T}_{R^\vee}$. The decision version of the Ring Learning With Error problem R-$\mathrm{SLWE}_{q,\Upsilon}$ is as follows: suppose $s \in R_q^\vee$ is uniformly random, and ψ sampled from Υ ; the goal is to distinguish between multiple independent samples from $A_{s,\psi}^{(R)}$ and the same number of independent samples from $U(R_q, \mathbb{T}_{R^\vee})$.

Theorem 03:

Let $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ and $q \geq 2$ of a known factorization such that $\alpha q > \omega(\sqrt{\log n})$. There is a quantum reduction from solving the Id-$\mathrm{GIVP}_\gamma^{\eta_\varepsilon}$ in polynomial time (in the worst case, with high probability) to solving the R-$\mathrm{SLWE}_{q,\Psi_{\leq\alpha}}$ in polynomial time with nontrivial probability with $\gamma = \sqrt{n}.\omega(\sqrt{\log n})/\alpha$.

Assuming q is prime, $q \leq \mathrm{poly}(n)$, and $q = 1 \bmod v$. Then there exists a polynomial time reduction from R-$\mathrm{SLWE}_{q,\Psi_{\leq\alpha}}$ to R-$\mathrm{LWE}_{q,\Upsilon_\alpha}$.

1.1.3. LWE over modules

The M-LWE generalizes the LWE and the R-LWE. The variable n and d denote the dimension of R and the rank of the $M \subseteq R^d$ module, respectively. We pose $N = nd$ the dimension of the lattice modulus.

Let ψ be a probability distribution on $\mathbb{T}_{R^\vee}$ and $s \in (R_q^\vee)^d$ a vector. We define by $A_{q,s,\psi}^{(M)}$ the distribution on $(R_q)^d \times \mathbb{T}_{R^\vee}$ obtained by choosing a vector $a \in (R_q)^d$ uniformly random, and $e \in \mathbb{T}_{R^\vee}$ by respect to ψ, and returning $\left(a, \frac{1}{q}\langle a, s\rangle + e\right)$.

Definition 03:

Let $q \geq 2$ and Ψ be a family of distributions on $\mathbb{T}_{R^\vee}$. The search version of the Module Learning With Error problem M-$\mathrm{SLWE}_{q,\Psi}$ is as follows: Let $s \in (R_q^\vee)^d$ be a secret vector and $\psi \in \Psi$; given arbitrarily many samples from $A_{q,s,\psi}^{(M)}$, the goal is to find s.
For an integer $q \geq 2$ and a distribution Υ over a family of distributions on $K_{\mathbb{R}}$. The decision version of the Module Learning With Error problem M-$\mathrm{SLWE}_{q,\Upsilon}$ is as follows: Let $s \in (R_q^\vee)^d$ be uniformly random and ψ sampled over Υ; the goal is to distinguish between arbitrarily many random samples from $A_{q,s,\psi}^{(M)}$ and the same number of samples from $U(R_q^d \times \mathbb{T}_{R^\vee})$.

As was the case with the LWE and R-LWE, the M-LWE problem can be interpreted in terms of linear algebra. It (the M-LWE) consists of taking the LWE matrix A of the form:

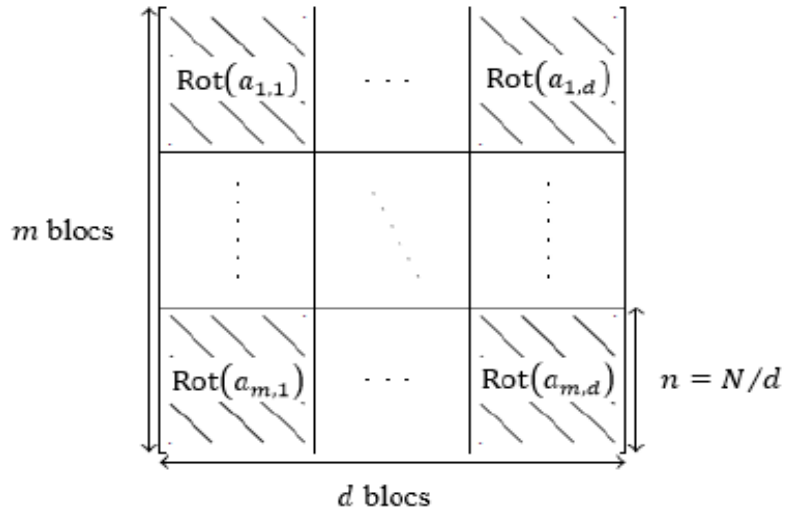


Figure 02: The M-LWE problem can be interpreted in terms of linear algebra.

The difficulty of M-LWE is supported by the two theorems that follow.

Theorem 04:

Let $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ and $q \geq 2$ of a known factorization such that $\alpha q > 2\sqrt{d}.\omega(\sqrt{\log n})$. There is a quantum reduction from solving Mod-GIVP$_\gamma^{\eta_\varepsilon}$ in polynomial time (in the worst case, with high probability) to solving M-SLWE$_{q,\Psi_{\leq\alpha}}$ in polynomial time with a nontrivial advantage with $\gamma = \sqrt{8Nd}.\omega(\sqrt{\log n})/\alpha$.

Assuming q is prime, $q \leq \text{poly}(N)$ and $q = 1 \bmod v$. Then there exists a polynomial time reduction from M-SLWE$_{q,\Psi_{\leq\alpha}}$ to M-LWE$_{q,\Upsilon_\alpha}$.

When $n = N$ and $d = 1$, Theorem 04 is equivalent to Theorem 03. When $n = 1$ and $d = N$, it (Theorem 04, again) is equivalent to Theorem 01.

Theorem 05:

Let $p, q \in \left[2, 2^{N^{O(1)}}\right]$ and $\alpha, \beta \in (0,1)$ such that $\beta \geq \alpha.\max\left(1, \frac{q}{p}\right).n^{1/4}N^{1/2}.\omega(\log^2 N)$ and $\alpha q \geq \omega\sqrt{\log(N)/n}$. There is a polynomial time reduction from M-LWE$_{q,\Upsilon_\alpha}$ to M-LWE$_{q,\Upsilon_\beta}$.

1.2. Reduction of Mod-GIVP to M-SLWE

The reduction from Mod-GIVP to M-SLWE uses the following intermediate problem, where ϕ denotes a real-valued function on a lattice and γ depends on the dimension, called the *Module Discrete Gaussian Sampling Problem* M-DGS$_\gamma^\phi$: given a module lattice M of dimension N, and a number $r > \gamma \cdot \phi$, the goal is to return a sample from $D_{M,r}$. The reduction is done in 2 (two) steps:

$$\text{Mod-GIVP}^{\eta_\varepsilon}_{\sqrt{8Nd}\cdot\omega(\sqrt{\log n})/\alpha} \xrightarrow{\textbf{Lemma 01}} \text{M-DGS}^{\eta_\varepsilon}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha} \xrightarrow{\textbf{Lemma 02}} \text{M-SLWE}_{q,\Psi_{\leq\alpha}}$$

Lemma 01:

For any $\varepsilon = \varepsilon(N) \leq 1/10$ and any γ and ϕ such that $\gamma \cdot \phi(M) \leq \sqrt{2}\eta_\varepsilon(M)$, there exists a polynomial-time reduction of the Mod-GIVP$^{\phi}_{2\sqrt{N}\cdot\gamma}$ to M-DGS$^{\phi}_{\gamma}$. [3]

Lemma 02:

Let $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ and q be an integer such that $\alpha q \geq 2\sqrt{d} \cdot \omega(\sqrt{\log n})$. Assuming we have access to an oracle that solves M-SLWE$_{q,\Psi_{\leq\alpha}}$, given a polynomial number of samples. Then there exists a polynomial-time quantum algorithm for M-DGS$^{\eta_\varepsilon}_{\sqrt{2d}\cdot\omega(\sqrt{\log n})/\alpha}$.

Lemma 03:

Let $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ and $q \geq 2$. Assume that we have access to an oracle that solves M-SLWE$_{q,\Psi_{\leq\alpha}}$ in polynomial time with nontrivial probability. Then there exists a polynomial-time quantum algorithm that, given an N -dimensional lattice modulus M, a number $r > 2q\eta_\varepsilon(M)$ and poly(N) samples from $D_{M,r}$, produces a sample from $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ with nontrivial probability.

To prove **Lemma 03** we use an intermediate problem, Mod-BDD$_\delta$ (BDD for *Bounded Distance Decoding*): given a lattice modulus M, $\delta < \lambda_1(M)/2$ and any point $y \in \mathbb{R}^n$ of the form $y = x + e$ for some $x \in M$ and $\|e\|_{2,\infty} \leq \delta$, find x. Note that the norm $\ell_{2,\infty}$ is used here, because it will help us state **Lemma 06** more appropriately.

Another intermediate problem q-Mod-BDD$_\delta$ will be used: given a lattice modulus M and a point $y \in \mathbb{R}^n$ in the distance (by respect to the norm $\ell_{2,\infty}$) δ from M, give as output the coset in M/qM of the nearest vector y. The proof of **Lemma 03** consists of the reduction sequence below (note that δ here takes the value $\frac{\alpha q\cdot\omega(\sqrt{\log n})}{\sqrt{2n}r}$).

Echantillons à partir de $D_{M,\frac{r\sqrt{d}\cdot\omega(\sqrt{\log n})}{\alpha q}}$ $\xrightarrow[\text{(quantique)}]{\textbf{Lemma 04}}$ Mod-BDD$_{M^\vee,\delta}$ $\xrightarrow{\textbf{Lemma 05}}$ q-Mod-BDD$_{M^\vee,\delta}$ $\xrightarrow{\textbf{Lemma 06}}$ M-SLWE$_{q,D_\alpha}$ Echantillon à partir de $D_{M,r}$

Lemma 04:

There exists an efficient quantum algorithm that, given an N-dimensional lattice modulus M, a number $\delta < \lambda_1(M^\vee)\omega(\sqrt{\log n})/(2\sqrt{n})$, and an oracle that solves Mod-BDD$_\delta$ on $M^\vee$, yields at the output samples from $D_{M,r\sqrt{d}\cdot\omega(\sqrt{\log n})/(\sqrt{2}\delta)}$.

Lemma 05:

For any $q \geq 2$, there exists a polynomial-time reduction from Mod-BDD$_\delta$ to q-Mod-BDD$_\delta$.

Lemma 06:
Let $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ and $q \geq 2$. Let $M \subseteq R^d$ be an R-module, and $r > \sqrt{2}q \cdot \eta_\varepsilon(M)$. Given an access to an oracle sampler from the distribution $D_{M,r}$, there is a probabilistic reduction from q-Mod-BDD$_{M^\vee,\frac{\alpha q\cdot\omega(\sqrt{\log n})}{\sqrt{2n}r}}$ to M-SLWE$_{q,\Psi_{\leq\alpha}}$.

The principle of the reduction is to construct from y, the input of q-Mod-BDD, and from some continuous or discrete Gaussian samples, the (a,b) peers distributed according to $A^{(M)}_{q,s,\psi}$, where s will directly depend on the x vector closest to y. To produce such samples (a,b) according to the desired distribution, we combine the

corresponding proofs for LWE and R-LWE. Then call the M-SLWE oracle to return s and allow us to recover the information about x.

Lemma 07:

Let $\varepsilon > 0$ and $s = \Theta(x \bmod qM^{\vee})$. There exists $\psi \in \Psi_{\leq\alpha}$ such that the statistical distance between $A_{q,s,\psi}^{(M)}$ and the distribution of (a, b) is at most 6ε.

All this concludes the statement of the first part of our Theorem 05.

For the second part of the theorem (existence of a reduction from M-SLWE to M-LWE), we invite our readers to consult the proof in our bibliography (paragraph 4.3 in [3]).

2. The cryptosystem CRYSTALS

The CRYSTALS or Cryptographic Suite for Algebraic Lattices encapsulates two types of cryptosystems: Kyber, a KEM (Key Encapsulation Mechanism, to recall) having the specificity of being classified as IND-CCA2 secure, and Dilithium a EUF-CMA secure signature algorithm. Both algorithms are based on difficult problems related to lattice modules, and were therefore built to resist possible attacks from quantum computers.

CRYSTALS, as mentioned in the introduction to this book, was submitted to the NIST Post-Quantum Cryptography Project. It was able to progress to stage 3 of the said standardization project. But above all, it is the only one among the few that have progressed to this final stage, to have both a KEM and a signature type cryptosystem. [5], [6], [8]

2.1. CRYSTALS-Kyber, the KEM type cryptosystem

2.1.1. Performance Analysis

In this section, we discuss the implementation aspects of Kyber and report performance results for two implementations: the *ANSI C* (ANSI for *American National Standards Institute*) reference implementation and an optimized implementation using *AVX2* vector instructions.

2.1.1.1. Kyber performance on Intel Haswell processors

Table 01 shows the Intel Haswell performance results of the reference implementation and an optimized Kyber AVX2 implementation and Kyber variant 90s as well as the key and ciphertext sizes. All benchmarks were obtained on a core of an Intel Core i7-4770K (Haswell) processor clocked at 3492 MHz (as indicated by /proc/cpuinfo) with TurboBoost and hyperthreading disabled. The machine used has 32 GB RAM (for *Random Access Memory*) and runs Debian GNU/Linux (GNU from *GNU's Not Unix*) with version 4.19.0 of the Linux kernel. Both implementations were compiled with gcc version 8.3.0. All cycle counts shown are the median of the cycle counts of 10000 executions of the respective function. The implementations are not optimized for memory usage, but in general, Kyber has very modest memory requirements.

Table 01 : Performance of Kyber on Intel Haswell processors

Kyber512					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	1632 (ou 32)	gen :	122684	gen :	33856
pk :	800	enc :	154524	enc :	45200
ct :	768	dec :	187960 (ou ≈ 288912)	dec :	34572 (ou ≈ 59088)
Kyber512 – 90s					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	1632 (ou 32)	gen :	213156	gen :	21880

pk :	800	enc :	249084	enc :	28592
ct :	768	dec :	277612 (ou ≈ 405268)	dec :	20980 (ou ≈ 38752)
Kyber768					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	2400 (ou 32)	gen :	199408	gen :	52732
pk :	1184	enc :	235260	enc :	67624
ct :	1088	dec :	274900 (ou ≈ 425492)	dec :	53156 (ou ≈ 82220)
Kyber768 – 90s					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	2400 (ou 32)	gen :	389760	gen :	30460
pk :	1184	enc :	432764	enc :	40140
ct :	1088	dec :	473984 (ou ≈ 671864)	dec :	301108 (ou ≈ 52512)
Kyber1024					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	3168 (ou 32)	gen :	307148	gen :	73544
pk :	1568	enc :	346648	enc :	97324
ct :	1568	dec :	396584 (ou ≈ 617848)	dec :	79128 (ou ≈ 115332)
Kyber1024 – 90s					
Sizes (in Bytes)		Hashwell Cycles (ref)		Hashwell Cycles (AVX2)	
sk :	3168 (ou 32)	gen :	636380	gen :	43212
pk :	1568	enc :	672644	enc :	56566
ct :	1568	dec :	724144 (ou ≈ 1009448)	dec :	44328 (ou ≈ 71180)

In addition, *ref* refers to the C reference implementation, AVX2 to the implementation using AVX2 vector instructions; *sk* stands for secret key, *pk* for public key and *ct* for ciphertext. In parentheses are approximate values when including key generation in decapsulation to avoid having to store expanded secret keys. In this scenario, we store only the initial seed d at the very beginning of the key generation algorithm Kyber. CPAPKE. KeyGen(). The approximate cycle counts for this scenario are calculated as the sum of the cycle counts for standard decapsulation and key generation minus the number of cycles required to generate the public seed ρ matrix **A**. Note that this is a very conservative estimate; actual implementations of the approach may also save, for example, sampling the 32 bytes of randomness.

2.1.1.2. Performances de Kyber sur les processeurs ARM Cortex-M4

Table 02 reports the cycle count and RAM usage of a C implementation (clean) and an optimized implementation (m4) of Kyber on an ARM Cortex-M4. All benchmarks are obtained using the pqm4 framework on a Discovery STM32F407 board. We do not report here benchmarks for the 90s version of Kyber on the Cortex-M4, as pqm4 does not yet include a constant-time implementation of AES.

Table 02: Performance of Kyber on ARM Cortex-M4 processors

Kyber512				
	M4 Cycles (clean)	M4 RAM (clean)	M4 Cycles (m4)	M4 RAM (m4)
gen :	655595	6020	463068	2844
enc :	865256	8668	561518	2484
dec :	961648	9444	519237	2508
Kyber768				
	M4 Cycles (clean)	M4 RAM (clean)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1087897	10052	756224	3292

enc :	1373744	13212	915676	2980
dec :	1491214	14308	853001	3004
Kyber1024				
	M4 Cycles (clean)	M4 RAM (clean)	M4 Cycles (m4)	M4 RAM (m4)
gen :	1696314	15180	1213303	3804
enc :	2057522	18844	1407769	3492
dec :	2199958	20420	1326409	3516

2.1.2. Security level estimation

In this paragraph, we have mainly relied on our bibliographies [1], [10] and [11].

Table 03 lists the security levels as defined in Section 4.A.5 of the NIST call for proposals [12] for the different parameter sets of Kyber. Our assertions are based on the cost estimates of the most well-known attacks against the MLWE problem underlying Kyber, as detailed in subsection 5.1 of our bibliography [1]. More precisely, the classical and quantum difficulty of the kernel-SVP are listed and used to derive the security levels.

All Kyber parameter sets have a certain probability of decryption failure. These failures are a security problem (see Section 5.5 of [1]) and thus the probabilities with which they occur must be small. But since in classical ROM the probability of decryption failure is information theoretic, we see no need to decrease it with the security parameter. In particular, the decryption failure for our level 3 and 5 parameter sets is less than 2^{-160}, which means that if 280 instances of Kyber were executed every second from now until our sun becomes a white dwarf, the odds are still strongly in favor of there never being a decryption failure. We therefore exclude these attacks from our claims about NIST security estimates.

The impact of deterministic noise caused by Compress_q *on Kyber512.* Each coefficient of $\mathbf{e_1}$ (and $\mathbf{e_2}$) in the Kyber. CPAPKE encryption algorithm behaves like a binomial distribution with parameter $\eta_2 = 2$, with variance $\eta_2/2 = 1$. The parameter $d_u = 10$ implies that the Compress_q function maps the elements modulo q to a set of size 2^{10} where the difference between all the even numbers of elements in said set is 3 or 4. This implies that the error added by the Compress_q function for each coefficient is either uniform over$\{-1,0,1\}$, $\{-1,0,1,2\}$ or $\{-2,-1,0,1\}$. It therefore has a variance at least as large as the uniform distribution on the set $\{-1,0,1\}$, or $2/3$. This makes the total variance of each coefficient of $\mathbf{e_1}$ plus the deterministic error at least $1 + 2/3 = 5/3$. The remaining secret and noise terms have binomial distributions with parameter $\eta_1 = 3$ for a variance of $\eta_1/2 = 3/2 < 5/3$. Taking into account the errors added by Compress_q, we therefore compute the Kyber512 difficulty by assuming that the variance of each secret/noise coefficient is $3/2$.

Table 03: Classical and quantum difficulty of the different parameter sets proposed for Kyber.

	Kyber512	Kyber768	Kyber1024
NIST security Level	1	3	5

Core-SVP methodology, Primal attack only			
Lattice attack dim. d	1003	1424	1885
BKZ-blocksize β (BKZ for Block Korkine Zolotarev)	403	625	877
Core-SVP classical hardness	118	182	256
Core-SVP quantum hardness	107	165	232

Refined estimate for classical attacks using			
Lattice attack dim. d	1025	1467	1918
BKZ-blocksize β	413	637	894

Sieving dimension $\beta' = \beta - d_{4f}$	375	586	829
$\log_2$(gates)	151.5	215.1	287.3
$\log_2$(memory in bits)	93.8	138.5	189.7

MAXDEPTH impact. The best-known quantum accelerations for the sifting algorithm, which are more detailed in the cost analysis (refer to subsection 5.1.1 of [1]), are only slightly affected by the depth limitation of a quantum circuit, as it uses Grover search on small sets (compared to search the entire AES key space). For the operating estimates of the core-SVP difficulty to match the quantum gate cost of breaking AES at the respective security levels, a quantum computer would have to support a maximum depth of 70 to 80. When limiting the maximum depth to smaller values, or when considering classical attacks, the estimates of the kernel-SVP difficulty are smaller than the number of gates for attacks against AES. A recent study of the concrete cost of sieving suggests that the quantum accelerations of these algorithms are tenuous, regardless of the value of MAXDEPTH.

2.2. CRYSTALS-Dilithium, the signature type cryptosystem

2.2.1. Security

The security of our signature algorithm can be proved through ROM, based on the difficulty of two problems [2], [9]. The first is the standard LWE problem (on polynomial rings) which requires distinguishing $(\mathbf{A}, \mathbf{t} := \mathbf{As_1} + \mathbf{s_2})$ from $(\mathbf{A}, \mathbf{u})$, where $\mathbf{u}$ is uniformly random. The other problem is what has been called the SelfTargetMSIS problem, which is the problem of finding a vector $\begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}$ with small coefficients and a satisfying message μ :

$$\mathrm{H}\left(\mu || [\mathbf{A}|\boldsymbol{t}|\mathbf{I}] \cdot \begin{bmatrix} \mathbf{z} \\ c \\ \mathbf{v} \end{bmatrix}\right) = c \qquad (01)$$

where $\mathbf{A}$ and t are uniformly random and $\mathbf{I}$ is the identity matrix. In ROM, one can obtain a (non-narrow) reduction using the forking lemma of the standard MSIS problem to find a z_0 with small coefficients satisfying $\mathbf{A}z_0 = 0$ at SelfTargetMSIS. This exact approach can be followed to prove that Dilithium is secure in ROM depending on the difficulty of MLWE and SIS.

In QROM, where the adversary can query H in overlay, the situation is somewhat different. It has been shown in Dilithium that it is still based on MLWE and SelfTargetMSIS in QROM, even with a narrow reduction when the algorithm is deterministic. But one can no longer use the forking lemma directly (since it is a type of rewinding) to give a quantum reduction from MSIS to SelfTargetMSIS. There are still good reasons to believe that the SelfTargetMSIS problem, and thus Dilithium, is secure in QROM. First, there are no natural signature schemes built from Σ protocols using the Fiat-Shamir transformation that are secure in ROM but not in QROM. Furthermore, it is possible to set the parameters of Dilithium (while leaving the algorithm design unchanged) so that the SelfTargetMSIS problem becomes theoretically difficult, thus leaving this version of Dilithium secure in the MLWE-based QROM only. An instantiation of these parameters results [2] in an algorithm with signatures and public keys that are 2x and 5x larger respectively. Although we do not consider this a good compromise, the existence of such a system gives us additional confidence in the security of an optimised version of Dilithium.

Recently, new work has narrowed the gap between security in ROM and QROM even further. This work [2] showed that if the underlying Σ protocol collapses and has a particular strength, then its Fiat-Shamir transformation is a secure signature in the QROM. The particular soundness of the Dilithium protocol is directly implied by the difficulty of MSIS. Moreover, the said work, again, states a probable protocol collapse Σ of Dilithium. Again in this work is shown that the collapse property has a reduction with respect to MLWE. The reduction is rather non-narrow, but it gives even more affirmation that there is nothing uncertain in the construction of Dilithium or any natural algorithm built via the Fiat-Shamir framework that can be proven safe in ROM. In our view, it is certain that the distinction between secure signatures in ROM and QROM will soon be treated in the same way as the distinction

between secure schemes in the standard model and ROM - there will be theoretical differences, but the security in practice will be the same.

In **Table 04**, the formulas for the public key and signature sizes are in our bibliography [2]. The numbers in parentheses for the SIS difficulty correspond to the strongly unforgeable (i.e., it is difficult to find a second distinct signature for a previously seen message/signature pair) version of the signature scheme. For the length of the coefficient corresponding to the parenthesis-free version, we always refer our readers to our bibliography [2] for more details. Due to the rejection sampling, there is a noticeable difference between the median and average signature times, so we measure both.

Table 04: Output sizes, safety and performance of Dilithium.

NIST Security Level	2	3	5
Output Size			
public key size (bytes)	1312	1952	2592
Signature size (bytes)	2420	3293	4595
LWE Hardness (Core-SVP and refined)			
BKZ block size b (GSA)	423	624	863
Classical Core-SVP	123	182	252
Quantum Core-SVP	112	165	229
BKZ block size b (simulation)	433	624	863
$\log_2$ Classical Gates	159	217	285
$\log_2$ Classical Memory	98	139	187
SIS Hardness (Core-SVP)			
BKZ block size b	423 (417)	638 (602)	909 (868)
Core-SVP Classique	123 (121)	186 (176)	265 (253)
Core-SVP quantum	112 (110)	169 (159)	241 (230)
Performance (Unoptimized Reference Code, Skylake)			
Gen median cycles	300,751	544,232	819,475
Sign median cycles	1,081,174	1,713,783	2,383,399
Sign average cycles	1,355,434	2,348,703	2,856,803
Verify median cycles	327,362	522,267	871,609
Performance (AVX2, Skylake)			
Gen median cycles	124,031	256,403	298,050
Sign median cycles	259,172	428,587	538,986
Sign average cycles	333,013	529,106	642,192
Verify median cycles	118,412	179,424	279,936
Performance (AVX2+AES, Skylake)			
Gen median cycles	70,548	153,856	153,936
Sign median cycles	194,892	296,201	344,578
Sign average cycles	251,144	366,470	418,157
Verify median cycles	72,633	102,396	151,066

2.2.2. Number of iterations

We would now like to calculate the probability that step 21 assigns $(\mathbf{z}, \mathbf{h})$ to $\perp$. The probability that $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ can be calculated by considering each coefficient separately. For each coefficient σ of $c\mathbf{s}_1$ the corresponding coefficient of $\mathbf{z}$ will be between $-\gamma_1 + \beta + 1$ and $-\gamma_1 + \beta - 1$ (inclusive) as long as the corresponding coefficient of $\mathbf{y}_i$ is between $-\gamma_1 + \beta + 1 - \sigma$ and $-\gamma_1 + \beta - 1 - \sigma$. The weight of this range is $2(\gamma_1 - \beta) - 1$ and the coefficient of $\mathbf{y}$ has $2\gamma_1 - 1$ possibilities. Thus the probability that any coefficient of $\mathbf{y}$ is in the correct range is :

$$\left(\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1 - 1}\right)^{256 \cdot l} = \left(1 - \frac{\beta}{\gamma_1 - 1/2}\right)^{ln} \approx e^{-256 \cdot \beta l / \gamma_1} \quad (02)$$

where we use the fact that our γ_1 values are larger compared to $1/2$.

We then tackle the calculation of the probability that we have:

$$\|\mathbf{r}_0\|_\infty = \left\|\text{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\right\|_\infty < \gamma_2 - \beta$$

If we assume (heuristically) that the lower order bits are uniformly distributed modulo $2\gamma_2$, then there are:

$$\left(\frac{2(\gamma_1 - \beta) - 1}{2\gamma_1 - 1}\right)^{256 \cdot k} \approx e^{-256 \cdot \beta k/\gamma_2}$$

probability that all coefficients are in the correct range (using the fact that our β values are large compared to $1/2$). Therefore, the probability that step 21 will succeed is:

$$\approx e^{-256 \cdot \beta (l/\gamma_1 + k/\gamma_2)} \quad (03)$$

Table 05: Parameter of Dilithium.

NIST Security Level	2	3	5
Parameters			
q [modulus]	8380417	8380417	8380417
d [dropped bits from $\mathbf{t}$]	13	13	13
τ [# of 1's in c]	39	49	60
challenge entropy $\left[\log\binom{256}{\tau} + \tau\right]$	192	225	257
γ_1 [$\mathbf{y}$ coefficient range]	2^{17}	2^{19}	2^{19}
γ_1 [low-order rounding range]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k, l) [dimensions of $\mathbf{A}$]	(4,4)	(6,5)	(8,7)
η [secret key range]	2	4	2
β [$\tau \cdot \eta$]	78	196	120
ω [max. # of 1's in the hint $\mathbf{h}$]	80	55	75
Répétitions (from the equation (08))	4.25	5.1	3.85

It is more difficult to formally calculate the probability that step 24 will result in a restart. The parameters were set so that heuristically $(\mathbf{z}, \mathbf{h}) = \perp$ with a probability between 1% and 2%. Therefore, the vast majority of loop repetitions will be caused by step 21.

We would like to emphasize that the expected number of iterations is independent of the secret key $\mathbf{s}_1$, $\mathbf{s}_2$ and thus no information about them can be obtained by an attack that counts the iterations.

3. Conclusions

This paper has allowed us to focus on the mathematic object that is the MLWE (Module Learning With Error) and its application as basis of the CRYSTALS-Kyber and Dilithium cryptosystems.

We have opened our writing on the development of the MLWE.

The section has put in exergue that there is a reduction (an equivalence in terms of difficulty or hardness then) from the Mod-GIVP (Generalized Independent Vector Problem over Module Lattices) and the M-LWE. To be more precise, the reduction was detailed for the reduction from Mod-GIVP to M-SLWE (Search version of the LWE over Module Lattices). The reader was then sent to our reference [3] for the second part of the reduction, which is M-SLWE to M-LWE.

We did then focus mainly on the cryptosystem CRYSTALS.

We started with the KEM (Key Encapsulation Mechanism) type one, the CRYSTALS-Kyber.

We advanced the specification of the two parts that compose it: Kyber.CPAPKE and Kyber.CCAKEM. Each of these two parts in their turn is subdivided into 03 algorithms: key generation, encryption, and decryption. Let us note in passing the fact that the operations of multiplication in the said algorithms, are carried out in the domain NTT (Number Theoretic Transform). The multiplication in R_q is performed in an efficient way (at a lower cost).

In terms of performance, it is at its peak for CRYSTALS-Kyber when it is written taking into account the advantages of processors supporting AVX2 technology for Intel Hashwell processors, and m4 technology for ARM processors.

Then came the case of CRYSTALS-Dilithium which falls in the category of signature cryptosystem.

In addition to the notion of NTT, which is still relevant, hash operations are also involved in a more marked way. We will list among others the ball hash which is represented in algorithmic terms in the function SampleInBall(ρ). ρ representing a seed.

Finally, the actual algorithm of CRYSTALS-Dilithium is given. It consists of 03 subparts: key generation, signature and verification.

4. References

[1]. R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Kyber - Algorithm Specifications And Supporting Documentation (version 2.0) »*, https://pq-crystals.org/kyber/index.shtml, Mar. 2019;

[2]. S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation »*, https://pq-crystals.org/dilithium/index.shtml, Mar. 2021;

[3]. A. Langlois and D. Stehle, *« Worst-Case to Average-Case Reductions for Module Lattices »*, https://eprint.iacr.org/2012/090, Aug. 2013;

[4]. O. Regev, *« On Lattices, Learning with Errors, Random Linear Codes, and Cryptography »*, https cims.nyu.edu ~regev papers qcrypto.pdf, May 2009;

[5]. *« Easy explanation of IND- security notions? »*, https://crypto.stackexchange.com/questions/26689/easy-explanation-of-ind-security-notions, Oct. 2020;

[6]. *« EUF-CMA and SUF-CMA »*, https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/, Oct. 2020;

[7]. V. Lyubashevsky, C. Peikert and O. Regev, *« On Ideal Lattices and Learning with Errors Over Rings »*, https://www.iacr.org/archive/eurocrypt2010/66320288/66320288.pdf, Nov. 2020;

[8]. S. Diop, B. O. Sané, M. Seck, *« NTRU-LPR IND-CPA A New Ideal Lattice-based Scheme »*, https://eprint.iacr.org/2018/109.pdf, Nov. 2020;

[9]. *« pq-crystals/security-estimates »*, https://github.com/pq-crystals/security-estimates, Feb. 2021;

[10]. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Kyber a CCA-secure module-lattice-based KEM »*, https://eprint.iacr.org/2017/634.pdf, Feb. 2021;

[11]. *« FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions »*, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, Feb. 2021;

[12]. « *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process* », https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf, Mar. 2021;

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND INNOVATIVE IDEAS IN EDUCATION

★★★

CERTIFICATE

of

PUBLICATION

The Board of International Journal of Advance Research and Innovative Ideas in Education is hereby Awarding this Certificate to

OMBANA FENOFAHASOAVANA RAZAFIMAHEFA

In Recognition of the Publication of the Paper Entitled

APPLICATION OF THE MLWE IN THE CRYPTOSYSTEM CRYSTALS – KYBER AS KEM AND DILITHIUM AS SIGNATURE

Published in E-Journal

Volume-7 Issue-5 2021

Paper Id : 15464
ISSN(O) : 2395-4396

IJARIIE

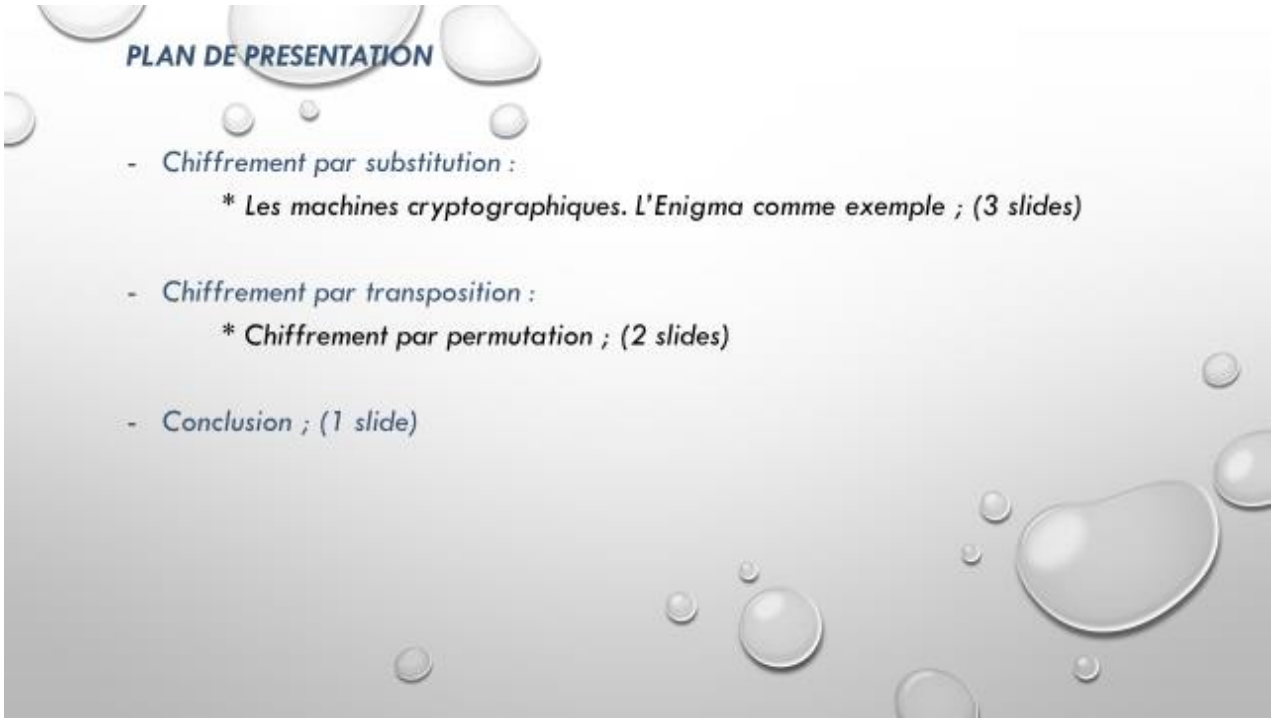
www.ijariie.com

Editor In Chief

Annexe 2 Communications scientifiques au sein de l'Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'Innovation

A2.1 Communication scientifique année D1, semestre 11





CHIFFREMENT PAR SUBSTITUTION > *Les machines cryptographique. L'Enigma comme exemple (1 sur 3)*

- *Il y a eu 2 (deux) principaux types de machines cryptographiques : ceux de type pinwheel (roues codeuses) et ceux faisant intervenir les rotors.*

- *Les machines cryptographiques de type pinwheel fonctionnent d'une manière uniquement mécanique, si le 2nd de type rotor fonctionne à la fois d'une manière mécanique et électrique.*

- *L'image à droite est celle de l'Enigma, la machine cryptographique à rotor que nous allons présenter.*

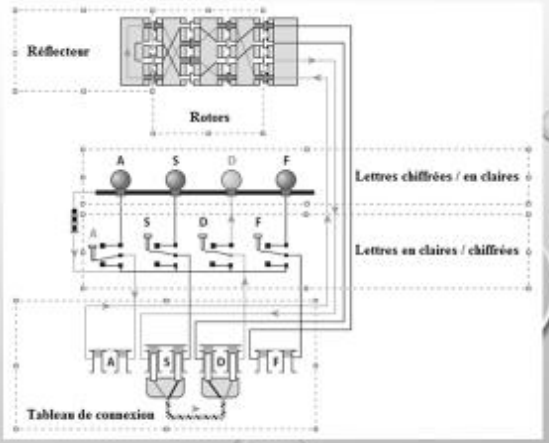


CHIFFREMENT PAR SUBSTITUTION > *Les machines cryptographique. L'Enigma comme exemple (2 sur 3)*

- *L'Enigma possède 3 rotors, qui effectue chacune un chiffre de substitution mono-alphabétique fixe (non modifiable), propre à chacun des rotors.*

- *Un Réflecteur qui effectue une substitution d'un premier groupe de 13 lettres vers un second groupe de 13 lettres.*

- *Un tableau de connexion qui permet de définir manuellement une substitution mono-alphabétique additionnelle à appliquer*



CHIFFREMENT PAR SUBSTITUTION > *Les machines cryptographique. L'Enigma comme exemple (3 sur 3)*

- *La formule du chiffrement effectué par une machine Enigma est représentée comme suit :*

$$E \equiv TRMLUL^{-1}M^{-1}R^{-1}T^{-1}$$

Où :

E représente le chiffrement (Encryption) de la machine Enigma dans sa globalité ;
T et T^{-1} représentent le chiffre du tableau de connexion (Table of connection ou plugboard) et son inverse ;
R et R^{-1} représentent le chiffre du rotor de droite (R pour Right) et son inverse ;
M et M^{-1} représentent le chiffre du rotor du milieu (M pour Middle) et son inverse ;
L et L^{-1} représentent le chiffre du rotor de gauche (L pour Left) et son inverse ;
U représente le chiffre symétrique du réflecteur.

CHIFFREMENT PAR TRANSPOSITION> *Chiffrement par permutation* ***(1 sur 2)***

- *Le chiffre par permutation consiste à assigner une autre position à toutes les lettres en claires données du message. Pour praticité, le message est subdivisé en un nombre donné de lettres, puis la permutation est effectuée sur chaque groupe de lettres. Encore pour praticité, la longueur de subdivision du message en claire ainsi que l'ordre de permutation proviennent d'un mot que l'on pourrait désigner comme clé du chiffre.*

- *L'exemple donné plus bas représente un chiffre de permutation utilisant le mot clé TALE.*

	4132	4132	4132	4132	4132	4132	4132	4132	4132	4132
Mot clé :	tale	tale	tale	tale	tale	tale	tale	tale	tale	tale
Message en clair :	theb	attl	eand	thes	word	thep	aper	andt	hepe	nllu
Message chiffré :	HBET	TLTA	ADNE	HSET	ODRW	HPET	PREA	NTDA	EEPH	LULN

CHIFFREMENT PAR TRANSPOSITION> Chiffrement par permutation (2 sur 2)

- *Pour la modélisation mathématique, la matrice de chiffrement est représentée sous la forme d'une matrice à 2 lignes et l colonnes, l étant la longueur du mot clé utilisé. La première ligne représente un nombre croissant allant de 1 au nombre de lettres du mot clé (ici 4). La seconde, la succession de chiffre dérivé du mot clé, dans notre cas TALE :*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$$

- *La matrice de déchiffrement, échanger la première et la seconde ligne de la matrice. Puis trier suivant la première ligne :*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$$

CONCLUSION (1 sur 1)

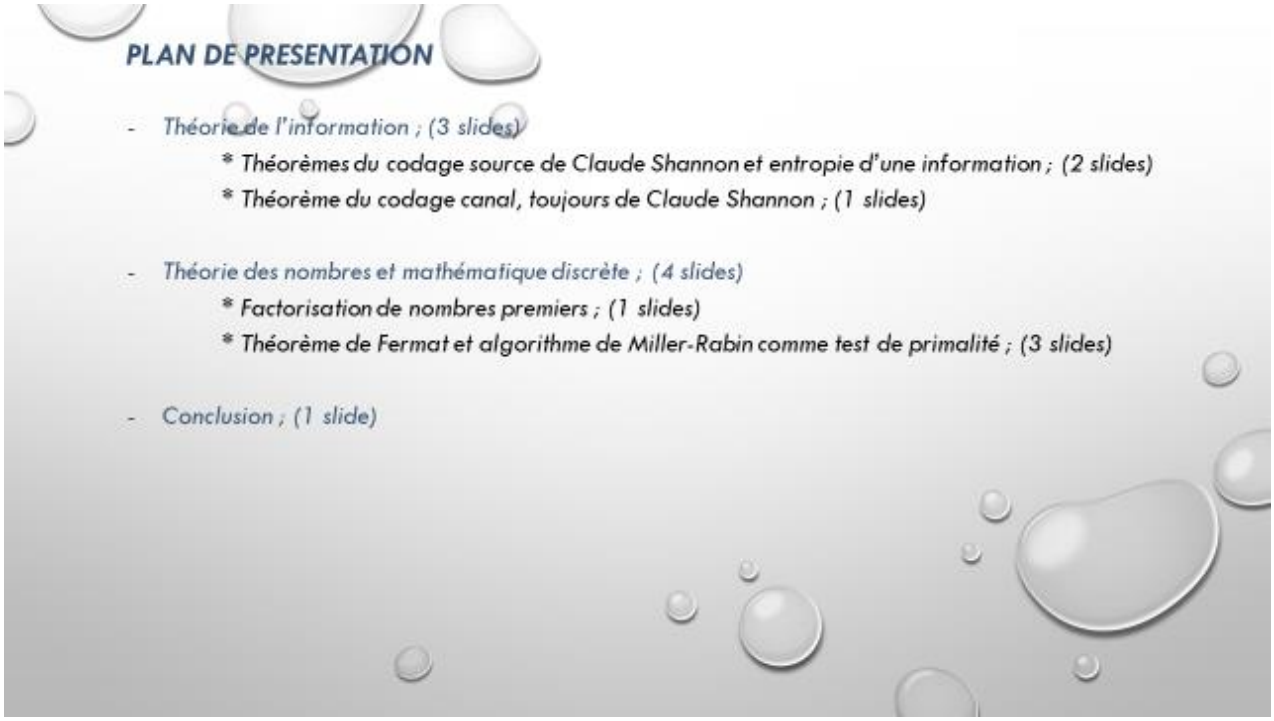
- *Dans cette présentation, nous avons pu évoquer les deux principaux types de méthodes cryptographiques, ce durant l'ère pré-informatique : le chiffrement par substitution et le chiffrement par transposition ;*

- *Nous en avons donné à chacun un exemple : l'énigma pour le premier, et le chiffre par permutation pour le second ;*

- *MERCI.*

A2.2 Communication scientifique année D1, semestre 12





THEORIE DE L'INFORMATION > *Théorèmes du codage source de Claude Shannon et entropie d'une information (1 sur 2)*

- *Le théorème du codage source* stipule qu'il est impossible de compresser une chaine de variables aléatoires indépendantes, identiquement distribuées (i.i.d.) – la longueur de la chaîne tendant donc vers l'infini – de telle sorte que le taux du code (la longueur moyenne des codes des variables) soit inférieur à l'entropie de la source, sans avoir la certitude que l'on perde de l'information. L'on peut cependant avoir une compression avec un taux de code proche de ladite entropie.

- *L'entropie d'une information* se définit comme étant la mesure de la quantité d'information contenue dans un message. Cette définition équivaut au fait que l'entropie est également la mesure de l'incertain dans un message. Elle est donnée par la formule qui suit :

THEORIE DE L'INFORMATION > *Théorèmes du codage source de Claude Shannon et entropie d'une information (2 sur 2)*

$$H(X) = -\sum_{k=1}^{K} p_k \log p_k$$

Où :

X dénote une variable aléatoire i.i.d. et $H(X)$ l'entropie de X ;

K le nombre de lettres possibles de l'alphabet dans lequel sont tirés les lettres du message composant X ;

p_k la probabilité d'apparition de chaque lettre a_k de l'alphabet ;

THEORIE DE L'INFORMATION > *Théorème du codage canal, toujours de Claude Shannon (1 sur 1)*

- *Le théorème du codage canal statue qu'il est possible de transmettre des données numériques sur un canal, même bruité, presque sans erreur à un débit maximum calculable. La limite ou capacité de Shannon d'un canal est le débit maximal de transfert d'information sur ce canal pour un certain niveau de bruit donné.*

- *La formulation mathématique du théorème est attribuée par la littérature au binôme Shannon-Hartley et est donnée par ce qui suit :*

$$C = B \log\left(1 + \frac{S}{N}\right)$$

Où :
C est la capacité du canal en bits par seconde (bits/s) ;
B est la bande passante du canal en hertz (Hz) ;
S/N est le rapport signal sur bruit (SNR ou Signal to Noise Ratio en anglais) ;

THEORIE DES NOMBRES ET MATHEMATIQUE DISCRETE > *Factorisation de nombres premiers (1 sur 1)*

- *Un nombre premier se définit comme étant un nombre positif ayant exactement deux diviseurs : 1 et lui-même. 1 n'est pas considéré comme étant un nombre premier.*

- *Tout nombre entier $a > 1$ peut être écrit d'une manière unique sous la forme:*

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_K^{a_K}$$

Où $p_1 < p_2 < \cdots < p_K$ sont des nombres premiers et $a_1, a_2, \ldots, a_K$ sont des nombres entiers positifs.

Exemple : $3600 = 24^4 \times 32^2 \times 52^2$; $11011 = 7 \times 11^2 \times 13$;

- *Soit en d'autres termes si P étant l'ensemble regroupant tous les nombres premiers p_k. L'on aura :*

$$a = \prod_{p_k \in P} p_k^{a_k},\ a_k \geq 0,$$

THEORIE DES NOMBRES ET MATHEMATIQUE DISCRETE > *Théorème de Fermat et algorithme de Miller-Rabin comme test de primalité (1 sur 3)*

- *Le théorème de Fermat statue que, si p est premier et que a est un nombre entier non divisible par p, l'on a :*

$$a^{p-1} \equiv 1 \pmod{p}$$

- *L'algorithme de Miller-Rabin est basé sur le fait que si n est premier, alors soit le premier élément de la liste $\left(a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}, a^{2^k q}\right) \bmod n$ est égal à 1, soit un élément de la liste est egal à $n-1$; sinon n est un nombre composite.*

- *D'un autre côté, si un nombre n vérifie une des deux conditions précédemment énumérées, cela ne signifie pas forcément que n est premier. Mais plutôt que n pourrait être premier avec une probabilité de ¼. Un usage répété t fois de l'algorithme permet de réduire ce taux d'erreur a $(¼)^t$.*

THEORIE DES NOMBRES ET MATHEMATIQUE DISCRETE > *Théorème de Fermat et algorithme de Miller-Rabin comme test de primalité (2 sur 3)*

- *Dénommons $TEST(n)$ l'algorithme de Miller-Rabin, les étapes correspondant sont comme suit :*

$TEST(n)$:

* *Trouver les entiers k et q, avec $k > 0$ et q impair tel que $n - 1 = 2^k q$;*
* *Sélectionner un nombre aléatoire a, tel que $1 < a < n - 1$;*
* *Si $a^q \bmod n = 1$ alors retourner « n pourrait être premier » ;*
* *Pour $j = 0$ à $k - 1$ faire si $a^{2^j q} \bmod n = n - 1$ alors retourner « n pourrait être premier » ;*
* *Retourner « n est composite » ;*

Exemple : Essayons de voir si 29 est premier. L'on a $29 - 1 = 28 = 2^2 \times 7 = 2^k q$.

THEORIE DES NOMBRES ET MATHEMATIQUE DISCRETE > *Théorème de Fermat et algorithme de Miller-Rabin comme test de primalité (3 sur 3)*

Premièrement, essayons $a = 10$. *L'on a* $10^7 \bmod 29 = 17$. *Ce qui n'est ni* 1 *ni* 28. *On continue avec* $(10^7)^2 \bmod 29 = 28$. *Ce qui nous emmène au résultat que* 29 *pourrait être premier.*

En essayant encore avec $a = 2$, *l'on trouve* $2^7 \bmod 29 = 12$ *et* $(2^7)^2 \bmod 29 = 28$. *Ce qui nous emmène, encore une fois, au résultat que* 29 *pourrait être premier. Si nous réitérons le calcul pour tout* $1 < a < 28$, *nous obtiendrons toujours le résultat comme quoi n pourrait être premier.*

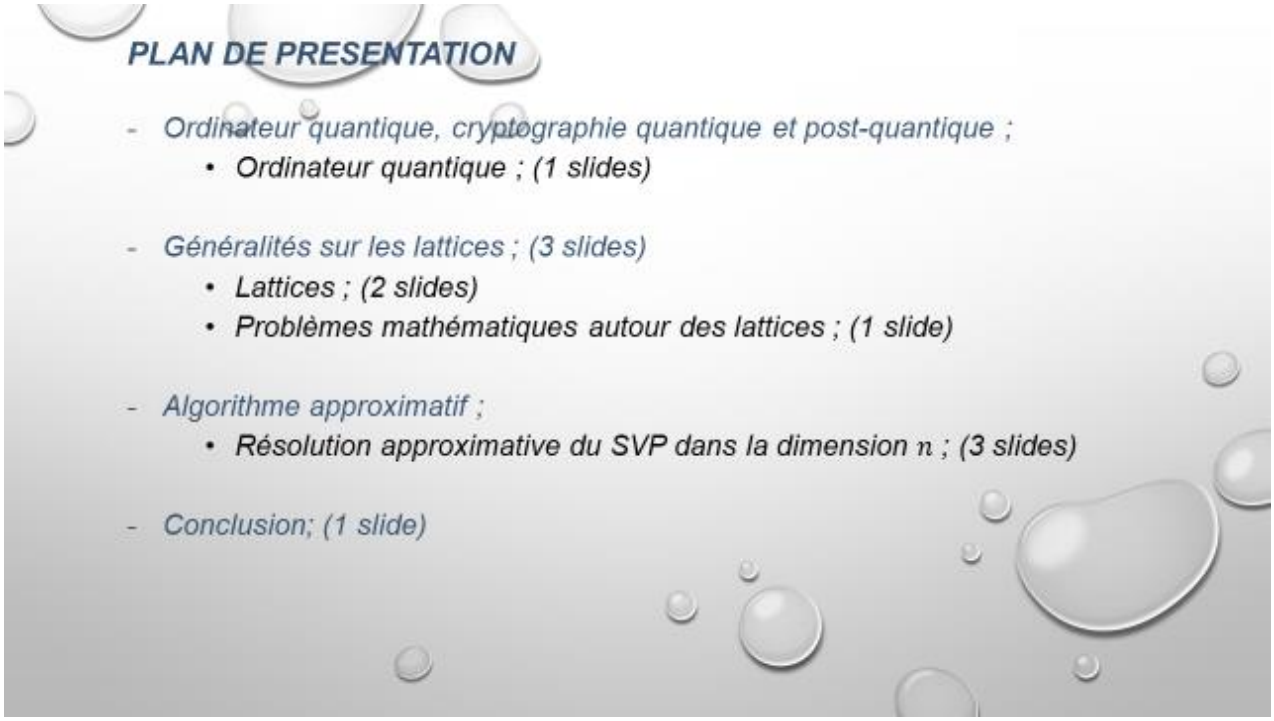
CONCLUSION (1 sur 1)

- *Cette présentation nous a permis de donner un apperçu sur des notions mathématiques indispensables à la cryptographie. Il s'est divisé en deux grandes parties. Celle relative à la théorie de l'information dans un premier temps. La seconde partie en est la théorie des nombres et la mathématique discrète.*

- *MERCI.*

A2.3 Communication scientifique année D2, semestre 13





ORDINATEUR QUANTIQUE, CRYPTOGRAPHIE QUANTIQUE ET POST-QUANTIQUE > *Ordinateur quantique (1 sur 1)*

- *Les ordinateurs quantiques utilisent la physique quantique pour calculer d'une manière différente ce que les ordinateurs classiques ne peuvent faire. Contrairement aux ordinateurs classiques, qui opèrent sur des bits qui sont soit 0 soit 1, les ordinateurs quantiques sont basés sur des bits quantiques (ou qubits) qui peut être simultanément 0 et 1, ce phénomène est connu sous le nom de superposition.*

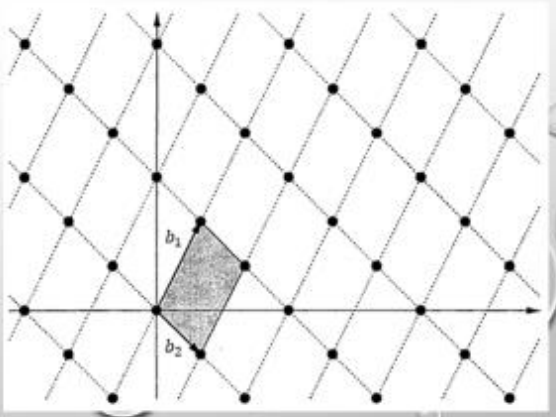
- *Un simple qubit Ψ est constitué de deux amplitudes, α et β, qui ne sont autres que des nombres complexe de la forme $a + bi$, ou $i^2 = -1$. Nous pouvons exprimer l'état du qubit par $\alpha\,|0\rangle + \beta\,|1\rangle$ où le "$|\,\rangle$" sont utilisé pour dénoter les vecteurs dans un état quantique.*

$$\Psi = \alpha\,|0\rangle + \beta\,|1\rangle,\ |\alpha|^2 + |\beta|^2 = 1$$

GÉNÉRALITÉS SUR LES LATTICES > *Lattices (1 sur 2)*

- *La figure à droite représente Une Lattice dans $\mathbb{R}^2$ où $\boldsymbol{b_1}$ et $\boldsymbol{b_2}$ en sont les vecteurs de bases.*

- *Le déterminant d'une lattice $\Lambda = \mathcal{L}(B)$, noté $\det(\Lambda)$, est le volume de dimension n du parallélépipède $\mathcal{P}(B)$ sur lequel s'étendent les vecteurs de bases (se référer à la zone ombré de le figure à droite.*



GÉNÉRALITÉS SUR LES LATTICES > Lattices (2 sur 2)

- Une méthode pour calculer le déterminant d'une lattice passe par l'algorithme d'orthogonalisation de Gram-Schmidt. Pour toute séquence de vecteurs $b_1, \dots, b_n$, l'on définit les vecteurs orthogonalisés $b_1^*, \dots, b_n^*$ par :

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \text{ où } \mu_{i,j} = \langle b_i, b_j^* \rangle / \langle b_j^*, b_j^* \rangle$$

Où $\langle x, y \rangle = \sum_{i=1}^{m} x_i y_i$ est le produit scalaire dans $\mathbb{R}^m$. Pour tout i, b_i^* est la projection de b_i orthogonale à $b_1, \dots, b_{i-1}$.

- Le déterminant de la lattice équivaut au produit de la longueur des vecteurs orthogonaux :

$$\det(\Lambda) = \prod_{i=1}^{n} \|b_i^*\|$$

GÉNÉRALITÉS SUR LES LATTICES > Problèmes mathématiques autour des lattices (1 sur 1)

- Shortest Vector Problem ou SVP : Etant donné une base de lattice $B \in \mathbb{Z}^{m \times n}$ le Shortest Vector Problem ou SVP revient à trouver un vecteur lattice non nul Bx avec $x \in \mathbb{Z}^n \setminus \{0\}$ de telle sorte que $\|Bx\| < \|By\|$ pour tout $y \in \mathbb{Z}^n \setminus \{0\}$. Nos bibliographies s'accordent sur le fait qu'aucun algorithme s'exécutant en un temps polynomial n'existe pour résoudre le SVP.

- Closest Vector Problem ou CVP : Soit toujours une base de lattice $B \in \mathbb{Z}^{m \times n}$, en addition de laquelle l'on mentionne un vecteur cible t, le Closest Vector Problem ou CVP consiste à trouver un vecteur lattice Bx qui se rapproche le plus du vecteur cible $t \in \mathbb{Z}^m$. C'est-à-dire, trouver un vecteur à éléments entiers ($x \in \mathbb{Z}^n$) de telle sorte que $\|Bx - t\| \leq \|By - t\|$ pour tout $y \in \mathbb{Z}^n$.

ALGORITHME APPROXIMATIF > Résolution approximative du SVP dans la dimension n (1 sur 3)

- *Bases réduites. Une base $B = [b_1, \dots, b_n] \in \mathbb{R}^{m \times n}$ est réduite en LLL (LLL-reduced en anglais, et LLL pour les initiales Lenstra–Lenstra–Lovász) selon le paramètre δ (réduite en δLLL en quelque sorte) si :*

(1) $|\mu_{i,j}| \leq \frac{1}{2}$ pour tout $i > j$, où $\mu_{i,j}$ sont les coefficients de Gram-Schmidt ;

(2) pour tout pair de vecteurs consécutifs b_i, b_{i+1} :

$$\delta \|\pi_i(b_i)\|^2 \leq \|\pi_i(b_{i+1})\|^2$$

Où $\pi_i(x)$ est la composante de x orthogonal aux vecteurs $b_1, \dots, b_{i-1}$. En particulier, les vecteurs orthogonalisés de Gram-Schmidt peuvent être exprimés comme $b_i^* = \pi_i(b_i)$.

ALGORITHME APPROXIMATIF > Résolution approximative du SVP dans la dimension n (2 sur 3)

- *Lemme 3.01 : Si $B = [b_1, \dots, b_n] \in \mathbb{R}^{m \times n}$ est une base réduite en δLLL avec $\delta \in]1/4, 1[$ alors, $\|b_1\| \leq \left(2/\sqrt{4\delta - 1}\right)^{n-1} \lambda_1$. En particulier si $\delta = 1/4 + 3/4^{(n/n-1)}$ alors $\|b_1\| \leq \left(2/\sqrt{3}\right)^n \lambda_1$.*

- *L'algorithme de réduction de base LLL :*

(1) Etape de réduction : $b_2 \coloneqq b_2 - cb_1$, où :

$$c = \left\lfloor \frac{\langle b_2, b_1 \rangle}{\langle b_1, b_1 \rangle} \right\rceil$$

avec $\lfloor x \rceil = \left\lfloor x + \frac{1}{2} \right\rfloor$ équivaut à l'arrondissement de la valeur $x + \frac{1}{2}$ à l'entier directement inférieur ou égal à elle-même (round half up en anglais).

ALGORITHME APPROXIMATIF > *Résolution approximative du SVP dans la dimension n (3 sur 3)*

- *L'algorithme de réduction de base LLL* (suite) :

(1) Etape d'échange : si $\|b_1\| > \|b_2\|$ échanger $b_1 \Leftrightarrow b_2$;

(2) Si b_1 et b_2 n'est pas réduite, répéter ;

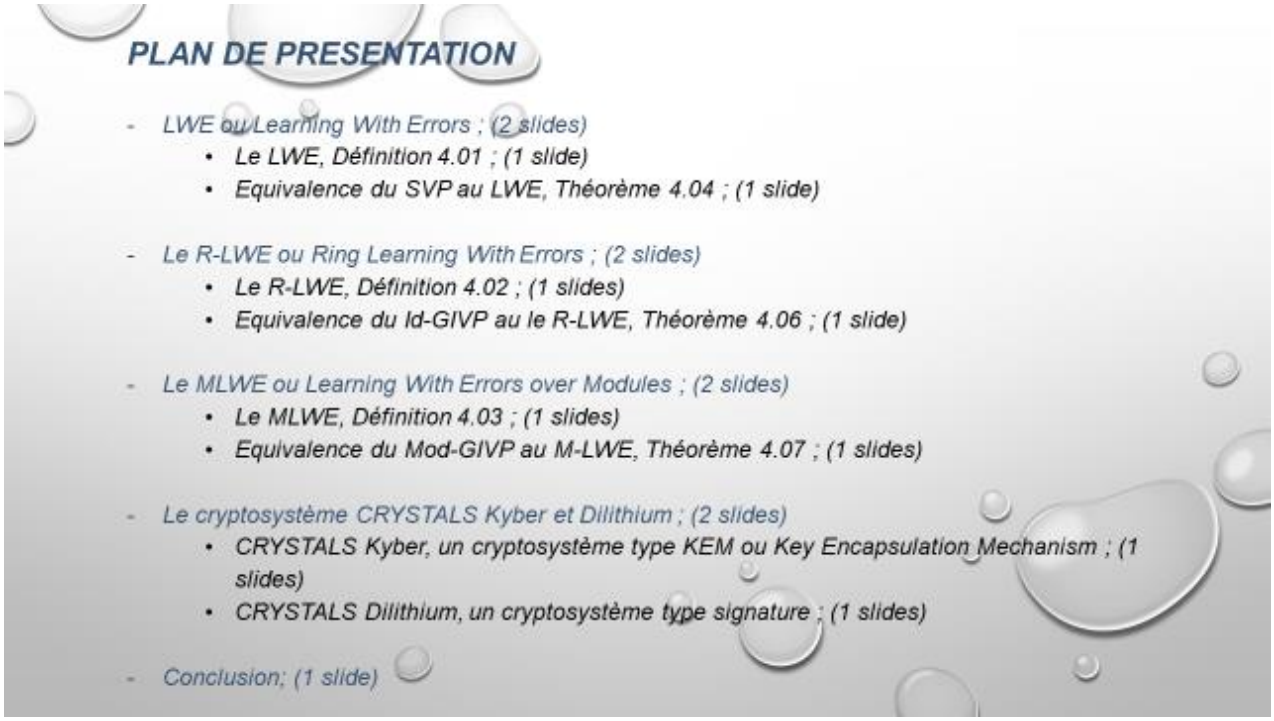
Ainsi si l'algorithme LLL se termine, la sortie en est une base réduite en LLL. Le fait que l'algorithme se termine toujours est démontré dans notre ouvrage. En addition, l'algorithme s'exécute en un temps polynomial en n.

CONCLUSION (1 sur 1)

- *Ce chapitre nous a permis de s'étaler sur deux sujets importants :*
    - *D'une part, la menace qu'apporte les ordinateurs quantiques aux cryptosystèmes largement utilisés en ce moment.*
    - *D'autre part, nous avons développé les basiques de la lattice, outil mathématique le plus promettant sur lequel se base bon nombre de candidats aux cryptosystèmes post-quantiques, ainsi que deux problèmes mathématiques fondamentaux qui lui sont relatifs : le SVP et le CVP.*
- *MERCI.*

A2.4 Communication scientifique année D2, semestre 14





LWE OU LEARNING WITH ERRORS > Le LWE, Définition 4.01

- *Définition 4.01 :*

La version recherche du problème Learning With Errors $\mathrm{SLWE}_{q,\chi}$ *est comme suit : soit* $s \in \mathbb{Z}_q^n$ *un secret ; étant donnés plusieurs échantillons à partir de* $A_{s,\chi}$*, le but est de trouver* s*.*

La version décision du problème Learning With Errors $\mathrm{LWE}_{q,\chi}$ *est comme suit : soit* $s \in \mathbb{Z}_q^n$ *choisi d'une manière uniforme et aléatoire ; le but est de faire la distinction entre plusieurs échantillons indépendants à partir de* $A_{s,\chi}$ *et le même nombre d'échantillons indépendants à partir de* $U(\mathbb{Z}_q^n \times \mathbb{T})$*. Où* U *fait référence à une distribution uniforme, sur l'ensemble* $\mathbb{Z}_q^n \times \mathbb{T}$ *dans le cas présent donc.*

LWE OU LEARNING WITH ERRORS > Equivalence entre le SVP et le LWE, Théorème 4.04

- *Théorème 4.04 :*

Soit $\varepsilon(n) = n^{-\omega(1)}$*,* $\alpha \in (0,1)$ *et* $q \geq 2$ *de telle sorte que* $\alpha q \geq 2\sqrt{n}$*. Il existe une réduction quantique de la résolution du* $\mathrm{GIVP}^{\eta_\varepsilon}_{\sqrt{8}n/\alpha}$ *en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du* $\mathrm{SLWE}_{q,D_\alpha}$ *en un temps polynomial avec une probabilité non négligeable.*

En assumant que q *est premier,* $q \leq \mathrm{poly}(n)$*, et que* χ *soit une fonction de densité de probabilité sur* $\mathbb{T}$*. Alors il existe une réduction en un temps polynomial du* $\mathrm{SLWE}_{q,\chi}$ *au* $\mathrm{LWE}_{q,\chi}$*.*

LE R-LWE OU RING LEARNING WITH ERRORS > Le R-LWE, Définition 4.02

- *Définition 4.02 :*

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Ring Learning With Errors $\text{R-SLWE}_{q,\Psi}$ est comme suit : soit $s \in R_q^\vee$ secret, et $\psi \in \Psi$; étant donné plusieurs échantillons à partir de $A_{s,\psi}^{(R)}$, le but est de trouver s.

Soit Υ une distribution sur une famille de distribution de bruits sur $\mathbb{T}_{R^\vee}$. La version décision du problème Ring Learning With Errors $\text{R-LWE}_{q,\Upsilon}$ est comme suit : supposons que $s \in R_q^\vee$ soit uniformément aléatoire, et ψ échantillonné à partir de Υ ; le but est de distinguer entre plusieurs échantillons indépendants partir de $A_{s,\psi}^{(R)}$ et le même nombre d'échantillons indépendants à partir de $U(R_q, \mathbb{T}_{R^\vee})$.

LE R-LWE OU RING LEARNING WITH ERRORS > Equivalence entre le Id-GIVP et le R-LWE, Théorème 4.06

- *Théorème 4.06 :*

Soit $\varepsilon(n) = n^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > \omega(\sqrt{\log n})$. Il existe une réduction quantique de la résolution du $\text{Id-GIVP}_\gamma^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\text{R-SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec une probabilité non négligeable avec $\gamma = \sqrt{n}.\omega(\sqrt{\log n})/\alpha$.

En supposant que q est premier, $q \leq \text{poly}(n)$, et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial de $\text{R-SLWE}_{q,\Psi_{\leq\alpha}}$ au $\text{R-LWE}_{q,\Upsilon_\alpha}$.

LE MLWE OU LEARNING WITH ERRORS OVER MODULES > Le MLWE, Définition 4.03

- *Définition 4.03 :*

Soit $q \geq 2$ et Ψ une famille de distributions sur $\mathbb{T}_{R^\vee}$. La version recherche du problème Module Learning With Errors $\text{M-SLWE}_{q,\Psi}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ un vecteur secret et $\psi \in \Psi$; étant donné arbitrairement plusieurs échantillons à partir de $A_{q,s,\psi}^{(M)}$, le but est de trouver s.

Pour un entier $q \geq 2$ et une distribution Υ sur une famille de distribution sur $K_{\mathbb{R}}$. La version décision du problème Module Learning With Errors $\text{M-LWE}_{q,\Upsilon}$ est comme suit : Soit $s \in \left(R_q^\vee\right)^d$ uniformément aléatoire et ψ échantillonné sur Υ ; le but est de distinguer entre arbitrairement plusieurs échantillons aléatoire à partir de $A_{q,s,\psi}^{(M)}$ et le même nombre d'échantillons à partir de $U\left(R_q^d \times \mathbb{T}_{R^\vee}\right)$.

LE MLWE OU LEARNING WITH ERRORS OVER MODULES > Equivalence du Mod-GIVP au M-LWE, Théorème 4.07

- *Théorème 4.07 :*

Soit $\varepsilon(N) = N^{-\omega(1)}$, $\alpha \in (0,1)$ et $q \geq 2$ d'une factorisation connue de telle sorte que $\alpha q > 2\sqrt{d}.\omega\left(\sqrt{\log n}\right)$. Il y a une réduction quantique de la résolution du $\text{Mod-GIVP}_\gamma^{\eta_\varepsilon}$ en un temps polynomial (dans le cas le plus défavorable, avec une probabilité élevée) vers la résolution du $\text{M-SLWE}_{q,\Psi_{\leq\alpha}}$ en un temps polynomial avec un avantage non négligeable avec $\gamma = \sqrt{8Nd}.\omega\left(\sqrt{\log n}\right)/\alpha$.

En assumant que q est premier, $q \leq \text{poly}(N)$ et que $q = 1 \bmod v$. Alors il existe une réduction à temps polynomial depuis $\text{M-SLWE}_{q,\Psi_{\leq\alpha}}$ vers $\text{M-LWE}_{q,\Upsilon_\alpha}$.

LE CRYPTOSYSTÈME CRYSTALS KYBER ET DILITHIUM > CRYSTALS Kyber, un cryptosystème type KEM ou Key Encapsulation Mechanism

- *CRYSTALS Kyber (ou Kyber tout court pour simplifier) est un cryptosystème type KEM (Key Encapsulation Mecanism) sécurisé IND-CCA2. La sécurité de Kyber est basée sur la difficulté de la résolution du problème Learning With Errors other Module lattices ou MLWE.*

- *La construction de Kyber est subdivisée en une approche en deux étapes : dans un premier temps, l'on fait appel à un algorithme de cryptage à clé publique, sécurisé IND-CPA, pour agir sur un message de longueur fixe égal à 32 bytes. Cette première étape se nomme Kyber.CPAPKE. Une version légèrement modifiée de la transformation de Fujisaki–Okamoto sera utilisée par la suite pour construire le KEM sécurisé IND-CCA2. Cette seconde étape du cryptosystème Kyber est dénommé par Kyber.CCAKEM.*

LE CRYPTOSYSTÈME CRYSTALS KYBER ET DILITHIUM > CRYSTALS Dilithium, un cryptosystème type signature

- *Le cryptosystème de signature numérique Dilithium, dont la sécurité est basée sur la difficulté du SVP dans les lattices, a été conçu en tenant compte des critères suivants :*

    - *Facile à implémenter d'une manière sécurisée ;*
    - *Adopter des paramètres efficacité qui dure dans le temps ;*
    - *Minimisez la taille de la clé publique et aussi celle de la signature ;*
    - *Cryptosystème modulaire, à niveau de sécurité facilement modifiable ;*

CONCLUSION (1 sur 1)

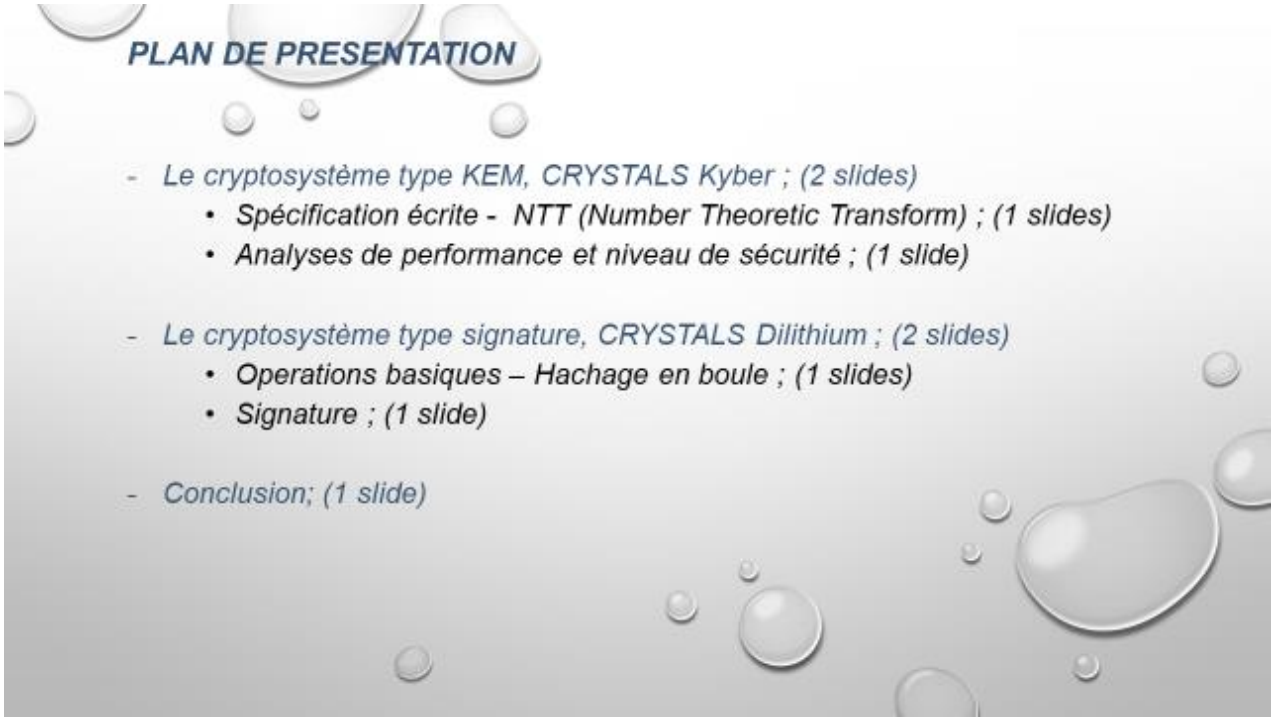
- *Dans le présent chapitre, nous avons pu introduire une nouvelle famille de technologie mathématique : le LWE, le R-LWE et le MLWE.*

- *Nous avons par la suite terminé le présent chapitre par un aperçu du cryptosystème CRYSTALS basé sur l'outil mathématique MLWE. CRYSTALS est composé de Kyber, un cryptosystème type KEM (Key Encapsulation Mechanism), et de Dilithium, un cryptosystème type signature.*

- *MERCI.*

A2.5 Communication scientifique année D3, semestre 15





LE CRYPTOSYSTÈME TYPE KEM, CRYSTALS KYBER >

Spécification écrite - NTT (Number Theoretic Transform)

- *Multiplication NTT et ordre de bit renversé : une manière très efficace d'effectuer une multiplication dans R_q est via le tant nommé Transformée Théorique des Nombres NTT (pour Number Theoritic Transform).*

- *Nous définissons* $\mathrm{NTT}: R_q \longrightarrow R_q$ *est la bijection qui mappe* $f \in R_q$ *aux polynômes* $\hat{f} \in R_q$ *:*

$$\mathrm{NTT}(f) = \hat{f} = \hat{f}_0 + \hat{f}_1 X + \cdots + \hat{f}_{255} X^{255}$$

avec :

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j}\, \zeta^{(2br_7(i)+1)j} \text{ et } \hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1}\, \zeta^{(2br_7(i)+1)j}$$

LE CRYPTOSYSTÈME TYPE KEM, CRYSTALS KYBER >

Analyses de performance et niveau de sécurité

- *En termes de performance, elle est à son apogée pour CRYSTALS Kyber lorsque ce dernier est écrite en prenant en compte les avantages des processeurs supportant la technologie AVX2 pour les processeurs type Intel Hashwell, et celui de la technologie m4 pour les processeurs type ARM.*

- *Pour parler de la sécurité, celle de Kyber est basée sur la robustesse du problème sous-jacent qu'est le MLWE. Deux type de réductions ont été avancées dans le présent chapitre pour supporter ce fait : une réduction étroite par respect au ROM (Random Oracle Model) pour la partie* $\mathrm{Kyber.CPAPKE}$ *et une réduction non-étroite par respect au QROM (Quantique Random Oracle Model) pour celle* $\mathrm{Kyber.CCAKEM}$.

LE CRYPTOSYSTÈME TYPE SIGNATURE, CRYSTALS DILITHIUM

> *Operations basiques – Hachage en boule*

- *En addition à la notion de NTT qui est toujours d'actualité, des opérations de hachage sont également impliquées d'une manière plus marquée. On énumèrera en autres le hachage en boule qui est représenté en termes algorithmique en la fonction* $\mathrm{SampleInBall}(\rho)$. ρ *représentant un germe.*

Algorithme 5.11 : $\mathrm{SampleInBall}(\rho)$

1 : Initialiser $\mathbf{c} = c_0 c_1 \dots c_{255} = 00 \dots 0$
2 : pour $i \coloneqq 256 - \tau$ à 255
3 : $\quad j \leftarrow \{0,1,\dots,i\}$
4 : $\quad s \leftarrow \{0,1\}$
5 : $\quad c_i \coloneqq c_j$
6 : $\quad c_j \coloneqq (-1)^s$
7 : retourner $\mathbf{c}$

LE CRYPTOSYSTÈME TYPE SIGNATURE, CRYSTALS DILITHIUM

> *Signature*

- *L'algorithme proprement dit de CRYSTALS Dilithium est donné. Il consiste en 03 sous parties : la génération de la clé, la signature et la vérification.*

- *Ci-dessous la partie correspondant à la vérification pour illustration :*

$\mathrm{Verify}(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, \tilde{c}))$

27 : $\mathbf{A} \in R_q^{k \times l} \coloneqq \mathrm{ExpandA}(\rho)$ — $\mathbf{A}$ est générée puis sauvegardée dans la représentation NTT en tant que $\widehat{\mathbf{A}}$
28 : $\mu \in \{0,1\}^{384} \coloneqq \mathrm{CRH}(\mathrm{CRH}(\rho || \mathbf{t}_1) || M))$
29 : $c \coloneqq \mathrm{SampleInBall}(\tilde{c})$
30 : $\mathbf{w}_1' \coloneqq \mathrm{UseHint}_q(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$ — Calculer $\mathrm{NTT}^{-1}(\widehat{\mathbf{A}} \cdot \mathrm{NTT}(z) - \mathrm{NTT}(c) \cdot \mathrm{NTT}(\mathbf{t}_1 \cdot 2^d))$
31 : retourner $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ et $[\![\tilde{c} = \mathrm{H}(\mu || \mathbf{w}_1')]\!]$ et $[\![\#\text{de 1s dans } \mathbf{h} \text{ est} \leq \omega]\!]$

CONCLUSION (1 sur 1)

- *Ce chapitre nous a permis mettre en exergue deux points importants :*
    - *Nous avons étalé d'une manière claire et nette les algorithmes définissant les cryptosystèmes CRYSTALS Kyber et Dilithium. Cryptosystèmes qui rappelons-le se basent sur l'objet mathématique MLWE ou Module Leaning With Error. Ce dernier, à son tour, possède une équivalence en termes de difficulté aux problèmes généraux relatifs à l'objet mathématique Lattice.*
    - *Des jeux de paramètres ont été avancés pour s'aligner sur le niveau de sécurité détaillé dans le cahier des charges concocté par le NIST (National Institute of Standards and Technology). Des détails additionnels sur la performance et la sécurité de l'algorithme, entre autres, sont donnés. Pour le cryptosystème type KEM (Key Encapsulation Mechanism) surtout.*
- *MERCI.*

REFERENCES

[1.01] J. Holden, *« The mathematics of secrets: cryptography from Caesar ciphers to digital encryption »*, Princeton University Press, New Jersey, Feb. 2017.

[1.02] W. Stallings, *« Cryptography and network security, principles and practice, seventh edition »*, Pearson Education Limited, England, 2017.

[1.03] C. Easttom, *« Modern cryptography, applied mathematics for encryption and information security »*, McGraw-Hill Education, New York, 2016.

[1.04] F. Carter, *« The Turing bombe »*, http://www.rutherfordjournal.org/article030108.html, Jun. 2018.

[2.01] K. Bostrom, « *Shannon source coding theorem* », Institut fur physik - Universitat Potsdam, Germany, Apr. 2015.

[2.02] P. Milne *« Channel Capacity »*, http://www.dip.ee.uct.ac.za/~nicolls/lectures/eee482f/04_chancap.pdf , 2003;

[2.03] « Entropy (Information Theory) », https://en.wikipedia.org/wiki/Entropy_(information_theory), 2019;

[2.04] A. Webster and S. Tavares, *« Chapter On The Design Of S-Boxes »*, Electrical Engineering - Queen's University, Kingston Ont. Canada, 1986;

[2.05] K. Conrad, *« Miller-Rabin Test »*, https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf, 2019;

[3.01] J. P. Aumasson, *« Serious Cryptography, A Practical Introduction to Modern Encryption »*, No Starch Press, 2017;

[3.02] « Simon's problem », https://en.wikipedia.org/wiki/Simon%27s_problem, March 2020;

[3.03] « Revealing XOR patterns I », https://www.cs.cmu.edu/~odonnell/quantum18/lecture11, March 2020;

[3.04] P. W. Shor, « Polynomial-time algorithms for prime factorization and discrete logarithms

on a quantum computer », Siam J. Comput., Oct. 1997, Oct. 1997;

[3.05] « Timeline of quantum computing », https://en.wikipedia.org/wiki/Timeline_of_quantum_computing, Apr. 2020;

[3.06] « Post-Quantum Cryptography Standardization », https://en.wikipedia.org/wiki/Post-Quantum_Cryptography_Standardization, Mar. 2020;

[3.07] « McEliece cryptosystem », https://en.wikipedia.org/wiki/McEliece_cryptosystem, Mar. 2020;

[3.08] « Multivariate cryptography », https://en.wikipedia.org/wiki/Multivariate_cryptography, May 2020;

[3.09] « NP-completeness », https://en.wikipedia.org/wiki/NP-completeness, May 2020;

[3.10] J. Ding and A. Petzoldt, « Current State of Multivariate Cryptography », https://www.researchgate.net/publication/319170467_Current_State_of_Multivariate_Cryptography, May 2020;

[3.11] X. Wang, « Mathematical Foundations of Public Key Cryptography », Science Press, 2016;

[3.12] D. Micciancio and S. Goldwasser, « Complexity of Lattice Problems: A Cryptographic Perspective », Springer, 2002;

[3.13] D. Micciancio, « Lattice Algorithms and Applications », https://cseweb.ucsd.edu/classes/wi10/cse206a/lec1.pdf, 2010.pdf

[4.01] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Kyber - Algorithm Specifications And Supporting Documentation (version 2.0) »*, https://pq-crystals.org/kyber/index.shtml, Mar. 2019;

[4.02] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe5, G. Seiler and D. Stehlé, *« CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation »*, https://pq-crystals.org/dilithium/, Mar. 2019;

[4.03] O. Regev, *« On Lattices, Learning with Errors, Random Linear Codes, and Cryptography »*, https://cims.nyu.edu/~regev/papers/qcrypto.pdf, May 2009;

[4.04] V. Lyubashevskyy, C. Peikertz and O. Regev, *« On Ideal Lattices and Learning with Errors Over Rings »*, https://eprint.iacr.org/2012/230, Jun 2013;

[4.05] A. Langlois and D. Stehle, *« Worst-Case to Average-Case Reductions for Module Lattices »*, https://eprint.iacr.org/2012/090, Aug. 2013;

[4.06] *« Post-Quantum Cryptography Standardization »*, https://en.wikipedia.org/wiki/Post-Quantum_Cryptography_Standardization, Sep. 2020;

[4.07] *« Big O notation »* , https://en.wikipedia.org/wiki/Big_O_notation, Sep. 2020;

[4.08] *« Big O tilde notation »*, https://www.johndcook.com/blog/2019/01/17/big-o-tilde-notation/, Sep. 2020;

[4.09] C. Gentry and C. Peikert, *« How to Use a Short Basis - Trapdoors for Hard Lattices and New Cryptographic Constructions »*, https://eprint.iacr.org/2007/432.pdf, Aug 2008;

[4.10] *« Easy explanation of IND- security notions? »*, https://crypto.stackexchange.com/questions/26689/easy-explanation-of-ind-security-notions, Oct. 2020;

[4.11] *« EUF-CMA and SUF-CMA »*, https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/, Oct. 2020;

[4.12] V. Lyubashevsky, C. Peikert and O. Regev, *« On Ideal Lattices and Learning with Errors Over Rings »*, https://www.iacr.org/archive/eurocrypt2010/66320288/66320288.pdf, Nov. 2020;

[4.13] S. Diop1, B. O. Sané1, M. Seck1, *« NTRU-LPR IND-CPA A New Ideal Lattice-based Scheme »*, https://eprint.iacr.org/2018/109.pdf, Nov. 2020;

[5.01] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Kyber - Algorithm Specifications And Supporting Documentation (version 3.0) »*, https://pq-crystals.org/kyber/resources.shtml,

Feb. 2021 ;

[5.02] *« In-place algorithm »*, https://en.wikipedia.org/wiki/In-place_algorithm, Feb. 2021 ;

[5.03] *« pq-crystals/security-estimates »*, https://github.com/pq-crystals/security-estimates, Feb. 2021 ;

[5.04] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seilery and D. Stehléz, *« CRYSTALS – Kyber a CCA-secure module-lattice-based KEM »*, https://eprint.iacr.org/2017/634.pdf, Feb. 2021 ;

[5.05] *« FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions »*, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, Feb. 2021 ;

[5.06] T. Saito, K. Xagawa, T. Yamakawa, *« Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model »*, https://eprint.iacr.org/2017/1005.pdf, Mar. 2021 ;

[5.07] *« Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process »*, https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf, Mar. 2021 ;

[5.08] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé, *« CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation »*, https://pq-crystals.org/dilithium/index.shtml, Mar. 2021 ;

FICHE DE RENSEIGNEMENT



Nom : RAZAFIMAHEFA

Prénoms : Fenofahasoavana Ombana

Adresse de l'auteur : Parcelle 23/44 Dépôt Analakininina, C/13 Betainomby

Téléphone : +261 32 37 846 82

E-mail : ombana.razafimahefa@gmail.com

Titre du mémoire :

CONTRIBUTION À LA MODÉLISATION ET À L'ÉVALUATION DE LA PERFORMANCE DES CRYPTOSYSTÈMES POST QUANTIQUES À BASE DE MLWE - CAS DE CRYSTALS KYBER ET DILITHIUM

Directeur scientifique : Mr RANDRIAMITANTSOA Paul Auguste, Professeur Titulaire

Téléphone : +261 34 10 342 58

E-mail : rpauguste@gmail.com

Codirecteur scientifique : RANDRIANANDRASANA Marie Emile, Maître de Conférences

Téléphone : +261 34 40 015 81

E-mail : emile3marie@gmail.com

Enseignante en Langue Française (Correction en orthographe et grammaire) :

Mme RAKOTOARINIVO Vololonarivo Herimalala

Téléphone : +261 34 01 131 71

E-mail : voarand@yahoo.fr

Nombre de pages : 341

Nombre de tableaux : 32

Nombre de figures : 40

RESUMES ET MOTS CLES

Les ordinateurs quantiques, mais toujours avec un faible nombre de qubits, commencent à émerger au moment où nous écrivons. Il est plus crucial que jamais de travailler sur de nouveaux cryptosystèmes. Des cryptosystèmes qui seront basés sur des problèmes mathématiques suffisamment robustes pour résister à la puissance de calcul accrue que ces nouveaux types d'ordinateurs fourniront. Au cours de nos recherches, nous sommes tombés sur l'objet mathématique qu'est le Module LWE. Le MLWE offre un compromis entre les deux extrêmes du LWE et du Ring-LWE. Par la suite, nous nous concentrerons sur le cryptosystème CRYSTALS, un cryptosystème basé sur le MLWE. Notre choix d'étudier ce cryptosystème est basé sur le fait qu'il a été soumis au projet du NIST (National Institute of Standards and Technology) de standardisation des cryptosystèmes post-quantiques. CRYSTALS a pu progresser jusqu'à l'étape 3 du projet. Mais surtout, il est le seul parmi les rares à avoir progressé jusqu'à cette étape finale, à avoir à la fois un cryptosystème de type KEM (Key Encapsulation Mechanism) et un autre de type signature.

Mots clés : Cryptosystème post-quantique, Module-LWE, CRYSTALS-Kyber et Dilithium, KEM, signature.

ABSTRACT AND KEYWORDS

Quantum computers, but still with a low number of qubits, are starting to emerge as we write. It is more crucial than ever to work on new cryptosystems. Cryptosystems that will be based on mathematical problems that are robust enough to withstand the increased computational power that these new types of computers will provide. In the course of our research, we came across the mathematical object that is the Module-LWE The MLWE offers a compromise between the two extremes of the LWE and the Ring-LWE. Thereafter we will focus on the CRYSTALS cryptosystem, a cryptosystem based on the MLWE. Our choice to study this cryptosystem is based on the fact that it was submitted to the NIST (National Institute of Standards and Technology) project of post-quantum cryptosystem standardization. CRYSTALS was able to progress to step 3 of the project. But above all, it is the only one among the few that have progressed to this final stage, to have both a KEM type cryptosystem (Key Encapsulation Mechanism) and another signature type one.

Keywords: Post Quantum Cryptosystem, Module-LWE, CRYSTALS-Kyber and Dilithium, KEM, Signature.