



UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT ELECTRONIQUE



MEMOIRE DE FIN D'ETUDES

En vue de l'obtention du diplôme

DOMAINE : Science de L'ingénieur

GRADE : Master

MENTION : Electronique

Parcours à visée de recherche : Télécommunication, Automatique, Signal et Image (TASI)

« POST-QUANTUM CRYPTO SPECIFIE PAR L'ORGANISATION NIST »

Présenté par : **RAKOTONDRAMANANA Radiarisainana Sitraka**

Soutenu le 14 décembre 2016 devant la Commission d'Examen composée de :

Président :

Monsieur RANDRIAMITANTSOA Paul Auguste, Professeur Titulaire

Examineurs :

Monsieur RAKOTOMIRAHLO Soloniaina, Professeur

Monsieur RATSIMBA Mamy, Maître de conférences

Monsieur RANDRIAMAROSON Rivo Mahandrisoa, Maître de conférences

Directeur de mémoire :

Monsieur RANDRIAMITANTSOA Andry Auguste, Maître de conférences

N° d'ordre :

Année Universitaire 2014-2015

REMERCIEMENTS

Avant toute chose, je rends grâce au Seigneur, qui m'a donné l'énergie, la santé et la volonté nécessaires, sans quoi je n'aurais pas pu achever ce mémoire de fin d'études.

Je tiens à remercier sincèrement Monsieur ANDRIANAHARISON Yvon, Professeur Titulaire, Directeur de l'Ecole Supérieure Polytechnique d'Antananarivo, Responsable du domaine science de l'ingénieur.

Ma gratitude et ma reconnaissance les plus sincères vont à Monsieur RANDRIAMITANTSOA Andry Auguste, Maître de conférences, qui, en tant que Directeur de ce mémoire, s'est toujours montré à l'écoute et très disponible tout au long de sa réalisation.

Un vif remerciement à Monsieur ANDRIAMANANTSOA Guy Danielson, Maître de conférence, Chef de Département de la mention Electronique au sein de l'E.S.P.A.

J'exprime également ma gratitude aux membres de jury, qui ont voulu examiner ce travail :

- Monsieur RAKOTOMIRAHLO Soloniaina, Professeur
- Monsieur RATSIMBA Mamy, Maître de conférences
- Monsieur RANDRIAMAROSON Rivo Mahandrisoa, Maître de conférences

Ce travail de mémoire n'aurait pu être mené de façon efficace et rigoureuse en parallèle à ma formation académique sans l'aide des différents enseignants et personnels administratifs de l'Ecole, à qui j'adresse toute ma gratitude.

Enfin, je n'oublie pas mes parents pour leur amour inconditionnel, leur contribution, leur soutien et leur patience. J'adresse mes plus sincères remerciements à tous mes proches et mes amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

TABLE DES MATIERES

Table des matières

REMERCIEMENTS.....	i
TABLE DES MATIERES	ii
LISTE DES ABREVIATIONS.....	vii
INTRODUCTION GENERALE	1
CHAPITRE 1 GENERALITE SUR LA CRYPTOGRAPHIE	2
1.1 Notion et Définitions	2
1.2 L'objectif de la cryptographie	3
1.2.1 Les types d'attaques (actives – passifs)	3
1.2.2 Les postulats C.A.I.N de sécurité associés à la cryptographie	4
1.2.3 Domaines d'application de la cryptographie	6
1.3 Les différents types de cryptographie ou les systèmes de chiffrements	6
1.3.1 Historique.....	6
1.3.2 Définitions d'un système cryptographique	7
1.3.3 Le chiffrement classique.....	8
1.4 Les chiffrements modernes	16
1.4.1 Algorithme à clé secrète.....	16
1.4.2 Algorithme à clé publique.....	17
1.5 Les générateurs des nombres aléatoires.....	17
1.6 Les fonctions de hachage	17

1.6.1 Généralités	17
1.6.2 Construction de fonction de Hachage.....	19
1.6.3 Exemples de fonctions de Hachage.....	21
1.7 Signature numérique	22
1.7.2 Exemples de signature à clé asymétrique	24
1.7.3 Scellement ou signature à clé symétrique	24
1.8 Cas pratique des signatures :	25
1.9 Conclusion	25
CHAPITRE 2 ETUDE APPROFONDIE SUR LA CRYPTOGRAPHIE MODERNE	26
2.1 Cryptographie symétrique	26
2.1.2 Les chiffrements à flot.....	27
2.1.3 Les chiffrements par blocs.....	28
2.2 Le mode de chiffrement.....	30
2.2.1 ECB	30
2.2.2 CBC	32
2.2.3 CFB.....	35
2.2.4 OFB.....	36
2.3 Chiffrement à clé publique.....	38
2.3.2 RSA.....	39
2.3.3 Cryptosystème d'El Gamal	40
2.3.4 Cryptosystème PGP.....	41

2.3.5 Cryptosystème RABIN	42
2.3.6 Cryptosystème basé sur les codes d'erreurs	43
2.3.7 Cryptosystème Menezes-Vanstone	45
2.4 Conclusion	46
CHAPITRE 3 ORDINATEUR QUANTIQUE ET PROBLEME CALCULATOIRE	47
3.1 La loi de Moore – Génération d'ordinateur futur	47
3.2 Les C-Bits et Q-Bits.....	48
3.3 Espace d'Hilbert.....	50
3.4 Les circuits quantiques	53
3.4.1 Hadamard Gate.....	53
3.4.2 Pauli Gate.....	54
3.4.3 Square root of Not Gate.....	54
3.4.4 Phase shift Gate	54
3.4.5 SWAP Gate.....	54
3.4.6 Square root of swap Gate.....	55
3.4.7 Controlled Gate	55
3.4.8 Remarques.....	56
3.5 Algorithmes quantiques.....	56
3.5.1 QFT.....	56
3.5.2 Algorithmes de recherche quantique de Grover	57
3.5.3 Algorithme de Schor – Factorisation quantique	61

3.6 Initiation à la création d'un ordinateur quantique	64
3.6.1 Concept de base.....	64
3.6.2 Evolution actuelle des ordinateurs quantiques.....	65
3.6.3 Loi Rose.....	65
3.7 Les problèmes calculatoires touchés par les ordinateurs quantiques	66
3.8 Les différents calculs combinatoires post-quantiques	68
3.8.1 Latticed based cryptography.....	68
3.8.2 Multivariate cryptographie	69
3.8.3 SIDH (Supersingular Isogeny Diffie Hellman).....	71
3.8.4 Code based et Hash based cryptography.....	73
3.9 Conclusion	73
CHAPITRE 4 IMPLEMENTATION D'UN ALGORITHME POST-QUANTIQUE	74
4.1 Etude approfondie de Mc-Eliece Cryptosystème	74
4.1.1 Généralités sur les codes correcteurs d'erreurs.....	74
4.1.2 Goppa Code.....	77
4.1.3 Première version de Mc-Eliece par Goppa Code.....	78
4.1.4 QC-MDPC code	80
4.1.5 Cryptosystème McEliece à base de graphe	81
4.2 Cryptographie Salsa20-Poly1305.....	85
4.2.1 PKDF2.....	85
4.2.2 Salsa20	85

4.2.3 Poly1305	89
4.3 Etude pratique d'un cryptosystème quantique	89
4.4 Conclusion	96
CONCLUSION GENERALE	97
ANNEXE	98
A1 Algorithme de décodage de Patterson	98
A2 Certaines classes de complexité algorithmique	99
BIBLIOGRAPHIE	100
FICHE DE RENSEIGNEMENTS	106

LISTE DES ABREVIATIONS

AES	Advanced Encryption Standard
BCH	Bose-Chaudhuri-Hocquenghem
BLISS	Base Level Integrated Support System
CAIN	Confidentialité Authenticité Intégrité et Non-répudation
CBC	Cipher Block Chaining
C-Bit	Classic-Bit
CFB	Cipher FeedBack
CRHF	Collision Resistant Hash Function
CTACK	CipherText Auto Key
CTR	CounTeR
CTS	Cipher Text Stealing
CVP	closest vector problem
DES	Data Encryption Standard
DH	Diffie Hellman
DHP	Diffie Hellman Problem
DK	Derived Key
DPA	Difference Power Analysis
DSA	Digital Signature Algorithm
DSS	Data Security Standard
ECB	Electronic CodeBook
ECDH	Elliptic Curve Diffie Hellman
EEA	Extend Euclidian Algorithm
GDHP	Generalized Diffie Hellman Problem
GDLP	Generalized Discret Logarithm Problem
GRS	Generalized Reed Solomon
HFE	Hidden Field Equation
HMAC	Hash-Based Message Authentication Code
HTTPS	HyperText Transfer Protocol Secure
IDEA	International Data Encryption Algorithm
IFP	Integer Factorization Problem
IOT	Internet Of Thing

IV	Initial Value
KAK	Key Auto Key
LDP	Logarithm Discret Problem
LDPC	Low-Density Parity Check
LFSR	Linear Feedback Shift Register
LWE	Learning with Errors
MAC	Message Authentication Code
MD	Message Digest
MDC	Manipulation Detection Code
MECS	McEliece Cryptosystem Security
MI	Matsumoto-Imai
MQ-PKC	Multivariate Quadratic Public Key Cryptosystem
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTRU	Number Theorists aRe Us / Number Theory Research Unit
OFB	Output Feedback Mode
OTP	One Time Padding
OTS	One Time Signature
P/NP	Polynomial/Non Polynomial
PGCD	Plus Grand Commun Diviseur
PGP	Pretty Good Privacy
PKDF2	Password Key of Derivation Function 2
PRF	PseudoRandom Function
PTM	Probabilistic Turing Machine
QC-MDPC	Quasi-Cyclic Moderate Density Parity-Check
QRP	Quadratic Residuosity Problem
QuBit	Quantum Bit
RC	Rivest's Cipher
RFC	Request For Comment
RIPE	RACE Integrity Primitives Evaluation
RM-Code	Reed Muller Code
RSA	Rivest, Shamir, & Adleman
RSAP	

RS-Code	Reed-Solomon COde
SCPA	Slide Channel Power Analysis
SHA	Secure Hash Algorithm
SIDH	Supersingular Isogeny Diffie Hellman
SIVP	Shortest Independant Vector Problem
SPA	Simple Power Analysis
SQROOT	Square root
SSH	Secure Shell
STS	Stepwise Triangle System
SVP	Shortest Vector Problem
TLS/SSL	Secure Socket Layer/Transport Layer Security
TMTO	Time Memory Traded Off
UOV	Unbalanced Oil and Vinegar
USA	United States of America
WPA2	Wi-Fi Protected Access 2
XTS	XEX-based Tweaked-codebook with Stealing

INTRODUCTION GENERALE

La présence de l'insécurité informatique et la deuxième guerre mondiale ont engendré l'évolution de la cryptographie actuelle. Les cryptanalystes deviennent de plus en plus intelligents et utilisent des méthodes perfectionnées telles que : Cryptanalyse analytique, attaque par implémentation ou attaque par des ordinateurs quantiques. La nouvelle technologie comme la 5G doit répondre à ces critères. En 2020, la 5G Iot sera mise en place et c'est la fin de IP-V4. Ces algorithmes de chiffrement sont appelés H2020 (Horizon 2020). Ils travaillent avec des protocoles à faibles poids appelés également Light-Weight Security. Comme les ordinateurs quantiques peuvent résoudre rapidement certains problèmes mathématiques de complexité non-polynomiale, alors à partir de 2006, l'organisation de normalisation des algorithmes de chiffrement appelée NIST organise chaque année la sélection des algorithmes résistants à des ordinateurs post-quantiques résistants aux attaques quantiques. Le dead-ligne de cette sélection sera l'année 2017, et les conférences de NIST pour cette année ont été effectués ce février 2016. De ce fait, il faut également proposer un algorithme de chiffrement répondant au critère spécifié par NIST. La plupart des anciens problèmes combinatoires pour sécuriser l'information jusqu'à maintenant va être délaissée et d'autres problèmes combinatoires vont être créés pour éviter ces attaques quantiques. Pour éclaircir ce contexte, ce mémoire s'intitule : « *Post-Quantum crypto spécifié par l'organisation NIST* ». L'Objectif de ce mémoire est d'élaborer une étude approfondie sur la cryptographie post-quantique et puis de fixer un algorithme post-quantique H2020 qui sera utilisé par le futur réseau 5G tout en respectant la norme dictée par NIST. Nous allons diviser ce travail en 3 grandes parties. Dans la première partie, nous allons parler de la généralité sur la cryptographie. Dans la seconde partie nous allons entamer sur l'étude approfondie de la cryptographie moderne. La troisième partie sera consacrée sur l'ordinateur quantique et les problèmes calculatoires. Pour la dernière partie, nous allons finaliser sur l'implémentation d'un algorithme post-quantique.

CHAPITRE 1 GENERALITE SUR LA CRYPTOGRAPHIE

1.1 Notion et Définitions

L'étude de la cryptographie nécessite quelques définitions et notions de bases pour éviter à des confusions de vocabulaires [1][2][3] :

- La *confidentialité* ou *le secret* : Elle assure qu'une information secrète ne peut être accédée par des personnes non autorisées.
- Le *code* : c'est un système de symbole pour pouvoir représenter une information (mais le code ne signifie pas un secret).
- La *cryptographie* : C'est une des disciplines de la cryptologie s'attachant à protéger des messages (assurant confidentialité/ou authenticité) en s'aidant souvent de secrets ou clés. Le mot cryptographie découle des mots grecs "**Krypto**" ou "**Cruptos**" qui veut dire "cacher ou dissimuler" (un secret) et "**Graphein**" qui veut dire "écrire " (une écriture ou un document). C'est la science des codes secrets permettant de rendre des informations confidentielles à travers un canal de transmission peu sûr. Par rapport à la théorie du codage qui consiste à protéger l'information face aux bruits non intentionnels dans le canal de transmission, la cryptographie vise à protéger les informations vis-à-vis de personnes malintentionnées.
- *Crypter* ou *Chiffrer* : C'est une opération qui consiste à rendre confidentiel une information.
- Un *cryptosystème* : C'est l'ensemble d'un système cryptographique permettant de crypter ou de décrypter une information.
- Le *texte clair* ou *PlainText* : C'est la donnée en entrée qui va être chiffrée afin de la rendre confidentielle
- Le *texte chiffré* ou *cryptogramme* ou *CipherText* : C'est un texte crypté par un cryptosystème.
- Le *cryptage* ou *chiffrement* : C'est le processus cryptographique qui permet de transformer un texte clair en cryptogramme.
- Le *déchiffrement* : C'est le processus cryptographique inverse qui permet de transformer un cryptogramme en texte clair. C'est une action exécutée par une personne autorisée.

- Une *clé de chiffrement / déchiffrement* : C'est une information secrète utilisée pendant une opération de cryptage / décryptage pour crypter / décrypter le texte clair / chiffré.
- La *cryptanalyse* : C'est une science permettant d'analyser la sécurité d'un cryptosystème. Souvent elle est utilisée par des parties non-autorisées pour casser ou trouver une faille dans un cryptosystème.
- La *cryptologie* : C'est une science qui étudie la cryptographie et la cryptanalyse.
- *Algorithme* : C'est une description non-ambigüe d'une méthode de résolution.
- *Protocole* : C'est une description non-ambigüe d'une suite d'interactions entre plusieurs participants.

1.2 L'objectif de la cryptographie

1.2.1 Les types d'attaques (actives – passifs)

Le but de la cryptographie est de fournir des moyens pour lutter contre les attaques passives, consistant à prendre frauduleusement connaissance de données transmises ou stockées, et contre les attaques actives, consistant à modifier frauduleusement leur origine ou leur contenu. Ainsi, L'objectif fondamental de la cryptographie est de permettre à deux personnes de communiquer à travers d'un canal peu sûr (téléphone, réseau informatique ou autre) sans qu'un opposant puisse comprendre ce qui est échangé. [3][4]

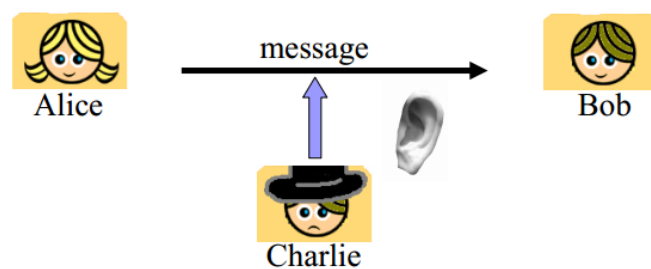


Figure 1.01 : Représentations d'un attaque passive dans le réseau

Les attaques actives nécessitent une intervention dans la ligne de communication. Quelques exemples de ses attaques les plus connues peuvent être citées comme suit :

- Impersonnification : modification de l'identité de l'émetteur ou récepteur
- Altération des données : modification des contenus

- Destruction des messages
- Retardement de la transmission
- Répudiation des messages : L'émetteur nie d'avoir envoyé le message

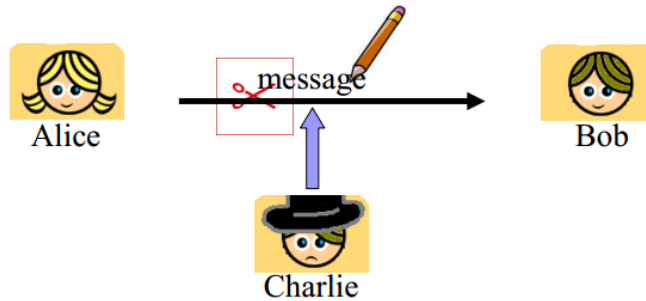


Figure 1.02 : *Représentation d'une attaque active*

1.2.2 Les postulats C.A.I.N de sécurité associés à la cryptographie

Le postulat C.A.I.N (Confidentialité Authentification Intégrité et Non répudiation ou encore Confidentiality Authentication Integrity and No-repudiation) est l'un des services que la cryptographie doit mettre en valeur [2][5].

- La Confidentialité : Il s'agit de rendre l'information inintelligible à tous les opposants tant lors de sa conservation qu'au cours de son transfert par un canal de communication. La confidentialité peut se représenter également par un stockage de données sécurisé.

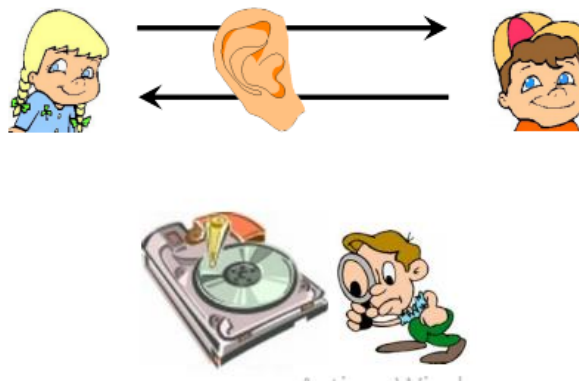


Figure 1.03 : *Le problème de confidentialité avec un stockage sécurisé*

- L'Authentification : Le contrôle d'accès et le contrôle d'identification.
 - Il s'agit d'authentifier les utilisateurs de façon à limiter l'accès aux données, serveurs et ressources aux seules personnes autorisées (un mot de passe pour un disque dur, par exemple).
 - L'identification : Le contrôle d'identification consiste à s'assurer que le destinataire est bien celui qui prétend être (authentification des partenaires) et d'obtenir une garantie que l'expéditeur a bien signé l'acte (authentification de l'origine des informations). Dans un réseau, la principale question consiste à assurer la provenance de message pour que personne ne doive contrefaire l'origine de donnée.

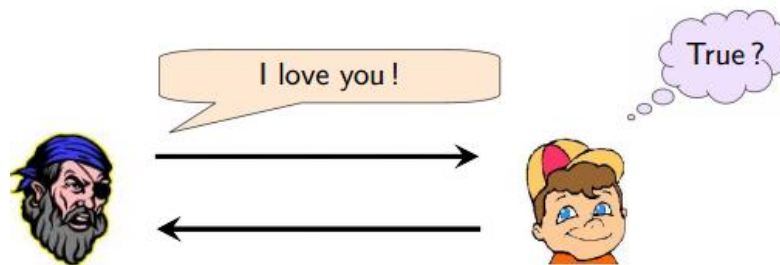


Figure 1.04 : *Le problème de l'authentification dans un réseau*

- L'Intégrité des données : Le contrôle d'intégrité d'une donnée consiste à vérifier que cette donnée n'a pas été altérée, frauduleusement ou accidentellement. Le contenu doit arriver à la destination de façon intact et sans modification.

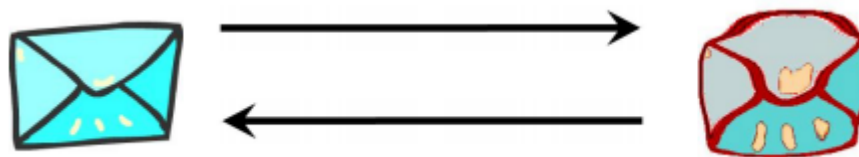


Figure 1.05 : *Le problème de de l'intégrité dans un réseau*

- Le Non répudiation : Il s'agit de garantir l'authenticité de l'acte. L'expéditeur ne peut nier le dépôt d'information, le récepteur ne peut nier la remise d'information, ni l'un ni l'autre

ne peut nier le contenu de cette information. Une signature ou un certificat permet à une personne de signer un document sans qu'elle puisse renier.



Figure 1.06 : *Non répudiation en utilisant des signatures numériques*

1.2.3 Domaines d'application de la cryptographie

La science du secret est aussi un domaine modulable selon les objectifs à atteindre, d'ailleurs pour Bruce Schneier [6], « *Il existe deux types de cryptographie dans le monde : la cryptographie qui empêche votre petite sœur de lire vos fichiers, et la cryptographie qui empêche les principaux gouvernements de lire vos fichiers* ».

1.3 Les différents types de cryptographie ou les systèmes de chiffrements

1.3.1 Historique

L'histoire de la cryptographie existe déjà depuis longtemps. Son utilisation se rapporte en Egypte il y a 4000 ans. Toutefois, pendant des siècles, les méthodes utilisées étaient restées souvent très primitives. De plus, sa mise en œuvre était limitée aux besoins de l'armée et de la diplomatie. La seconde guerre mondiale a joué un rôle très important dans l'amélioration des méthodes de chiffrement et de cryptanalyse, ainsi qu'une profonde influence sur le cours de celle-ci. C'est la prolifération actuelle des systèmes de communication qui a fait sortir la cryptographie du domaine militaire. De plus, elle a diversifié la demande et provoqué le développement de nouvelles techniques cryptographiques. Elle est à l'origine d'un développement rapide depuis les dernières décennies, qui ne semble pas s'essouffler aujourd'hui, bien au contraire. Beaucoup de systèmes de chiffrement ont été imaginés pour se protéger contre la curiosité et la malveillance des ennemis

depuis des siècles. On peut classer ces systèmes en trois grandes classes représentées sur la Figure 1.07 [7][8].

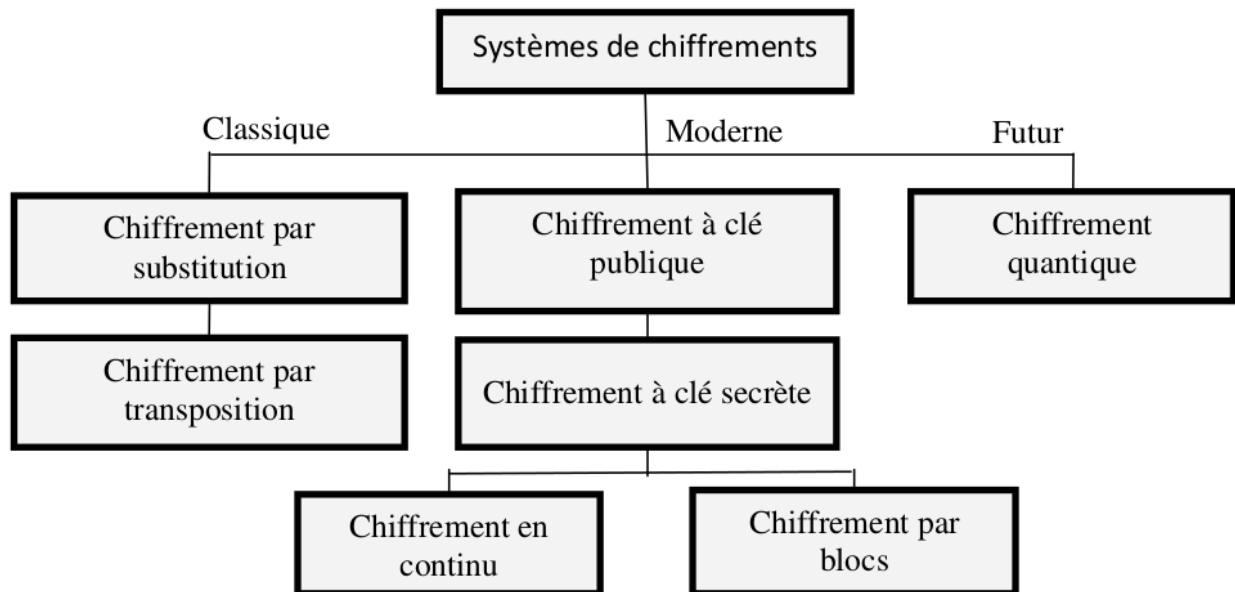


Figure 1.07 : *Les principales techniques de chiffrement*

1.3.2 Définitions d'un système cryptographique

Un système cryptographique est un quintuple $(P, C, K, \mathcal{E}, D)$ satisfaisant les conditions suivantes [3][7] :

1. P est un ensemble fini de bloc de **textes clairs** possibles
2. C est un ensemble fini de bloc de **textes chiffrés** possibles
3. K est un ensemble fini de **clés** possibles
4. Pour tout $K \in K$, il y a **une règle de chiffrement**, $e_K \in \mathcal{E}$ et une règle de déchiffrement correspondante $d_K \in D$. Chaque $e_K: P \rightarrow C$ et $d_K: C \rightarrow P$ sont des fonctions telles que $d_K(e_K(x)) = x$ pour tout texte clair $x \in P$.

La principale propriété est la quatrième. Elle précise que si un texte clair x est chiffré en utilisant e_K , et si le texte chiffré y obtenu est déchiffré en utilisant d_K , on retrouve le texte clair x original.

1.3.3 Le chiffrement classique

De l'antiquité à la renaissance, l'art de la cryptographie mise en œuvre, pour cacher la substance d'un texte, une combinaison plus ou moins élaborée de substitutions et de permutations [9] [10]. Dès l'origine, la cryptographie a été utilisée à des fins diplomatiques puis militaires. Historiquement, la substitution est le premier type de chiffrement utilisé. C'est un chiffrement dans lequel chaque caractère du texte en clair est remplacé par un autre caractère dans le texte chiffré.

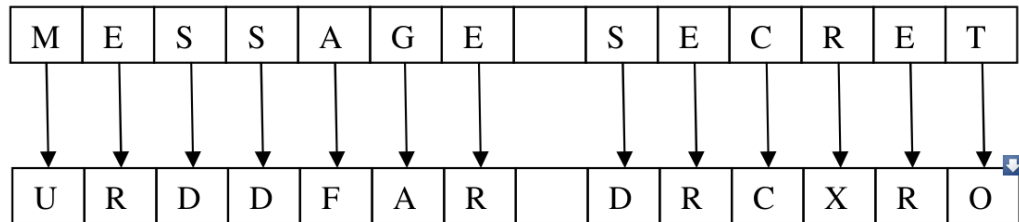


Figure 1.08 : *Principe de la substitution*

Un chiffrement par transposition est un chiffrement dans lequel les caractères du texte en clair demeurent inchangés mais dont les positions respectives sont modifiées.

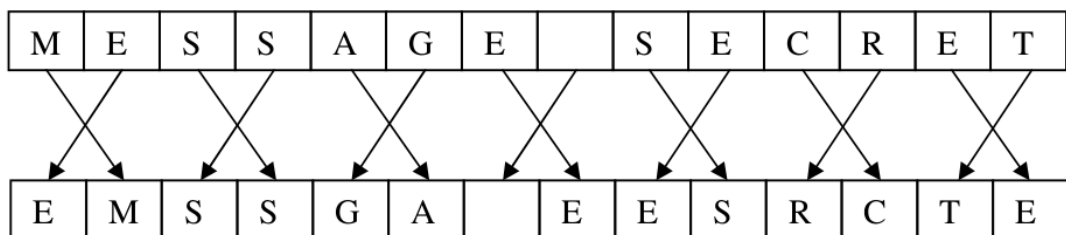


Figure 1.09 : *Principe de la transposition*

1.3.3.2 Les techniques traditionnelles :

- En Egypte, 2000 ans avant notre ère un scribe avait gravé des hiéroglyphes transformés sur la pierre tombale de Khumhotep II pour rendre inintelligible la description de sa vie.
- Jules César dans la Guerre des Gaules décrit un procédé de substitution bien connu aujourd'hui : Il consiste à un chiffrement par décalage d'un nombre de rangs convenu la lettre « claire » dans l'alphabet usuel. Si la clé est 3, a est remplacé par d et b par e ...

Alphabet	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
Substitution	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

Tableau 1.01: *Tableau représentatif de la cryptographie de Jules César*

- A Sparte, au IV^{ème} siècle avant notre ère, les communications entre les chefs des armées et les commandants étaient chiffrées à l'aide d'une Scytale, un bâton sur lequel on enroule une lanière en spires jointives.



Figure 1.10 : *Scytale*

- En Crète, un disque de Phaïstos, datant de 1700 avant notre ère, comportant un texte chiffré sur les deux faces a été retrouvé. Ce texte n'est encore à ce jour décrypté de manière sûre.

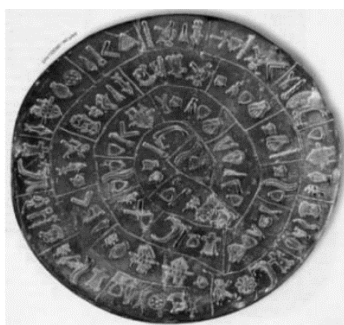


Figure 1.11 : *Disque de Phaïstos*

1.3.3.3 Les techniques avancées

Les techniques reposent toujours sur la substitution et les permutations mais utilisent des bases mathématiques plus robustes. La plupart de ses cryptographies est inspirée de celle de la technique traditionnelle [11] [12]

a. Définition du modulo et modulus

Si a , b et m des entiers, et si $m > 0$, on écrit $a \equiv b \pmod{m}$ ou $a \equiv b [m]$ se lit a congru b modulo m . L'entier m est appelé **modulus**.

Supposons que l'on divise a et b par m . On obtient des quotients et des restes, ceux-ci étant compris entre 0 et $m-1$. Précisément, on a :

$$a = q_1m + r_1 \text{ et } b = q_2m + r_2 \text{ avec } 0 \leq r_1 \leq m-1 \text{ et } 0 \leq r_2 \leq m-1$$

Ainsi, il est facile de voir $a \equiv b \pmod{m}$ ou $a \equiv b [m]$ si et seulement si $r_1 = r_2$.

On représentera par $a \bmod m$ (sans parenthèse) le reste dans la division de a par m . On a donc $a \equiv b \pmod{m}$ si, et seulement, si $a \bmod m = b \bmod m$. Si on remplace a par $a \bmod m$, on dit que l'on *réduit* a modulo m .

Dans plusieurs langages de programmation, $a \bmod m$ est défini comme l'entier congru à a et compris entre $-m+1$ et $m-1$ et de même signe que a . Il est cependant plus commode de définir le reste comme étant toujours positifs.

b. Chiffrement par décalage :

C'est un chiffrement basé par l'arithmétique modulaire. Par définition, on a un système cryptographique $(P, C, K, \mathcal{E}, D)$, Soit $P = C = \mathbb{Z}_{26}$. On définit alors

$$\forall (x, y) \in \mathbb{Z}_{26}; e_K(x) = x + K \bmod(26) \quad (1.01)$$

$$d_K(y) = y - K \bmod(26) \quad (1.02)$$

Pour la clé particulière $K = 3$; on dit que le système cryptographique est appelé **Chiffrement de César**, car il est utilisé par Jules César lors de la guerre de Gaules.

c. Chiffrement par substitution

Par définition, on a un système cryptographique $(P, C, K, \mathcal{E}, D)$, Soit $P = C = \mathbb{Z}_{26}$ et K est l'ensemble des permutations sur l'ensemble de 26 nombres 0,1,2,...,25. Pour chaque permutation $\pi \in K$ on définit

$$\forall (x, y) \in \mathbb{Z}_{26}; e_\pi(x) = \pi(x) \quad (1.03)$$

$$d_\pi(y) = \pi^{-1}(y) \text{ avec } \pi^{-1}: \text{permutation réciproque de } \pi \quad (1.04)$$

La clé de déchiffrement est simplement la permutation des 26 alphabets. Pour une recherche exhaustive, le nombre de clef est donc $26!$ qui est supérieur à $4,6 * 10^{26}$. Une recherche exhaustive est donc un peu trop grande même pour un ordinateur.

d. Chiffrement affine

Un **chiffrement par décalage** n'est en fait qu'un cas particulier du **chiffrement par substitution** qui n'utilise que 26 cas des $26!$ permutations possibles. Un autre cas de chiffrement par substitution est le **chiffrement affine**. Dans ce procédé, on limite la fonction de chiffrement par la forme [12] :

$$\text{Soit } P = C = \mathbb{Z}_{26} \text{ et } \forall (x, a, b) \in \mathbb{Z}_{26}; e_K(x) = ax + b \text{ mod}(26) \quad (1.06)$$

Ces fonctions sont appelés *fonctions affines*, d'où le nom du procédé. On note que l'on retrouve le chiffrement par décalage en prenant $a = 1$.

Pour que la fonction de déchiffrement soit possible, il est nécessaire que la fonction affine soit injective. Autrement dit :

$$\forall y \in \mathbb{Z}_{26}; ax + b \equiv y \text{ mod}(26) \quad (1.07)$$

doit avoir une solution x , l'équation est équivalent à :

$$\forall y \in \mathbb{Z}_{26}; ax \equiv y - b \text{ mod}(26) \quad (1.08)$$

Pour que cette fonction soit injective, il est nécessaire que **pgcd(a,26) = 1** ; Pour pouvoir déchiffrer le message, il faut connaître une opération mathématique du nom de l'inverse en utilisant la table de multiplication de \mathbb{Z}_{26} . On donne sa définition par :

Soit $a \in \mathbb{Z}_m$; l'inverse de a est l'élément a^{-1} tel que $a \cdot a^{-1} \equiv 1 \text{ mod}(m)$

La définition générale d'un système cryptographique affine est donc :

Soit $P = C = \mathbb{Z}_{26}$ et soit $K = \{(a, b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} : \text{pgcd}(a, 26) = 1\}$

soit $K = (a, b) \in K$, on définit :

$$\forall (x, y) \in \mathbb{Z}_{26}; e_K(y) = ax + b \text{ mod}(26) \quad (1.09)$$

$$\text{et } \forall (x, y) \in \mathbb{Z}_{26}; d_K(y) = a^{-1}y - b \text{ mod}(26) \quad (1.10)$$

Le nombre de clefs de chiffrement affine est donné par $m \cdot \phi(m)$ avec $\phi(m)$ est la fonction indicatrice d'Euler.

Soit la décomposition en élément simple de m définit par $m = \prod_{i=1}^n p_i^{e_i}$

avec les p_i sont des nombre premiers et $e_i > 0$ et $1 \leq i \leq n$.

$$\phi(m) = \prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1}) \quad (1.11)$$

Dans notre cas on utilise le modulus 26 c'est-à-dire $m = 26$.

e. Chiffrement de Vigenère

Dans le cas de chiffrement par décalage et du chiffrement par substitution, dès que la clef est fixée, chaque caractère alphabétique est transformé en un unique caractère alphabétique. Pour cette raison, ce procédé est appelé mono-alphabétique. Blaise Vigenère [11], a inventé un chiffrement qui n'est pas mono-alphabétique définit par :

Soit $m > 0$; $P = K = C = (\mathbb{Z}_{26})^m$ Pour toute clé $K = (k_1, k_2, \dots, k_m)$. On définit :

$$e_K(x_1, x_2, \dots, x_m) = (x_1 + k_1, x_2 + k_2, \dots, x_m + k_m) \quad (1.12)$$

$$\text{et } d_K(y_1, y_2, \dots, y_m) = (y_1 - k_1, y_2 - k_2, \dots, y_m - k_m) \quad (1.13)$$

Pour faciliter le chiffrement Vigenère, on liste tous les décalages des alphabets dans un carré de Vigenère.

Clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	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
2	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
3	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
4	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
5	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
6	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
7	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
8	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
9	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
10	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
11	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
12	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
13	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
14	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
15	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
16	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
17	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
18	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
19	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
20	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
21	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
22	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
23	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
24	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
25	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
26	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 1.12 : Carré de Vigenère

Imaginons que nous allons chiffrer « COULEUR » par la clé « TRIAGE ». En commençant par la ligne 19 qui contient la lettre T jusqu'à avoir la colonne de l'alphabet C, on voit que V remplace C et F remplace O ... On remarque toute suite que le chiffrement Vigenère est poly-alphabétique. Le cas possible de clé montre considérablement de 26^m possibilités qui est déjà impossible à casser si le pirate utilise une recherche exhaustive.

f. Chiffrement de Hill

Le chiffrement de Hill est un chiffrement poly-alphabétique inventé par Lester S. Hill en 1929[12]. Soit m un entier strictement positif, et soit $P = C = (\mathbb{Z}_{26})^m$. L'idée est de transformer m caractères d'un bloc du texte en clair en m caractères d'un bloc du texte chiffré par des combinaisons linéaires. En général, on prend une matrice $m \times m$ pour clef K . Soit $k_{i,j}$ le coefficient de i -ème ligne et j -ème colonne de la matrice K , on écrit $K = (k_{i,j})$. Pour $x = (x_1, x_2, \dots, x_m) \in P$ et $K \in K$, On calcul $y = e_K(x) = (y_1, y_2, \dots, y_m)$ ainsi par la formule suivante :

$$(y_1, y_2, \dots, y_m) = (x_1, x_2, \dots, x_m) \begin{pmatrix} k_{1,1} & \dots & k_{m,1} \\ \vdots & \ddots & \vdots \\ k_{m,1} & \dots & k_{m,m} \end{pmatrix} \quad (1.14)$$

Soit $y = x \cdot K$

On dit que le texte chiffré est obtenu par transformation linéaire. Pour voir comment le procédé de déchiffrement fonctionne, on utilise donc la propriété de la matrice inverse.

$$x = y \cdot K^{-1} \quad (1.15)$$

On définit les opérateurs matriciels suivants :

- Si $A = (a_{i,j})$ une matrice $l \times m$ et $B = (b_{j,k})$ une matrice $m \times n$ on définit le produit matriciel $AB = (C_{i,k})$ par la formule :

$$C_{i,k} = \sum_{j=1}^m a_{i,j} \times b_{j,k} \quad (1.16)$$

- Le déterminant d'une matrice $m \times m$, $A = (a_{i,j})$ se calcule de cette manière :

$$\det(A) = a_{1,1} \times a_{2,2} - a_{1,2} \times a_{2,1}; \text{ pour } m = 2 \quad (1.17)$$

On note $A_{i,j}$ la matrice obtenue en supprimant le i -ème ligne et j -ème colonne de la matrice A . En prenant t quelconque du ligne et z quelconque de la colonne, on a :

$$\det(A) = \sum_{i=1}^m (-1)^{i+z} \cdot A_{i,z} = \sum_{j=1}^m (-1)^{t+j} \cdot A_{t,j} \quad (1.18)$$

- La comatrice notée $K^* = (k_{i,j}^*)$ est une matrice $m \times m$ dont le coefficient sera

$$k_{i,j}^* = \sum (-1)^{i+j} \cdot K_{i,j} \quad (1.19)$$

- La matrice inverse est définie par $K^{-1} = (\det(K))^{-1} \times K^*$

La définition générale du chiffrement de Hill est donc comme suit :

Soit m un entier strictement positif et $P = C = (\mathbb{Z}_{26})^m$ et

$K = \{\text{matrices } m \times m \text{ inversible dans } \mathbb{Z}_{26}\}$, pour toute clef K , on définit :

$$e_K(x) = x \cdot K \text{ et } d_K(x) = y \cdot K^{-1} \quad (1.20)$$

g. Chiffrement par permutation

Tous systèmes cryptographiques précédents reposent sur la substitution : tous caractères de textes claires sont remplacés par un autre dans un texte chiffré. L'idée de chiffrement par permutation consiste à réordonner le texte clair selon Giovanni Porta en 1563 pour la première fois. Donald E. Knuth fut des études approfondies concernant la permutation dans ses livres de 3 éditions intitulant « The art of Computer programming » [13] [14] [15]

Soit un entier strictement positif m . Soit $P = C = \{0, 1, \dots, 25\}^{25}$

et soit K l'ensemble de permutation de $\{1, \dots, m\}$.

Pour toute clef π (pour toute permutation) on définit :

$$e_\pi(x_1, x_2, \dots, x_m) = (x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(m)}) \text{ et } d_\pi(y_1, y_2, \dots, y_m) = (y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)}, \dots, y_{\pi^{-1}(m)}) \quad (1.21)$$

Supposons que l'on a un texte clair : shesellsseashellsbytheseashore

Et prenons $m = 6$; la permutation $\pi = \begin{pmatrix} 123456 \\ 351652 \end{pmatrix}$ on aura $\pi^{-1} = \begin{pmatrix} 123456 \\ 351652 \end{pmatrix}$

On découpe le texte en groupe de 6 lettres : shesel lsseas hellsb ythese ashore
chaque bloc est réordonné suivant la permutation π .

eeshlsh salses lshble hsyeeet hraeos

Le texte chiffré est donc : eeshlshsalseslshblehsyeeethraeos

On voit bien **que le chiffrement par permutation** est un cas particulier **du chiffrement de Hill**. A toute permutation π donnée de l'ensemble $\{1,2,\dots,m\}$, on peut associer une matrice $m \times m$ $K_\pi = (K_{i,j})$ suivant la formule :

$$K_{i,j} = \begin{cases} 1 & \text{si } i = \pi(j) \\ 0 & \text{sinon} \end{cases} \quad (1.22)$$

h. Chiffrement de chaine

Dans les systèmes précédents, les éléments du texte clair sont chiffrés de la même manière à partir du clef K . En fait, la chaine du texte chiffré y est obtenue ainsi [11] [12] :

$$y = y_1 y_2 \dots = e_K(x_1) e_K(x_2) \dots \quad (1.23)$$

Les procédés de ce type sont appelés chiffrement par bloc. Une autre approche consiste à utiliser le chiffrement à chaine, l'idée de base consiste à engendrer une séquence de clefs $z = z_1 z_2 \dots$ et à l'utiliser pour chiffrer la chaine $x = x_1 x_2 \dots$ suivant la règle $y = y_1 y_2 \dots = e_{z_1}(x_1) e_{z_2}(x_2) \dots$

Un chiffrement en chaine fonctionne ainsi. Supposons que l'on ait une clé $K \in K$ et un texte clair $x_1 x_2 \dots$. On utilise une fonction f_i pour créer z_i (le i -ème séquence de la clefs) à l'aide de clef K et les $i-1$ premiers caractères du texte clair. Généralement, on écrit f_i par récurrence.

$$z_i = f_i(K, x_1, x_2, \dots, x_{i-1}) \quad (1.24)$$

Le déchiffrement se fait aussi pas à pas en commençant par :

$$z_1, x_1, z_2, x_2 \dots$$

Nous pouvons donner la définition générale d'un système cryptographique à chaine :

Un **chiffrement à chaine** est un tuple $(P, C, K, \ell, F, \mathcal{E}, D)$

- P est un ensemble fini de bloc de **textes clairs** possibles
- C est un ensemble fini de bloc de **textes chiffrés** possibles
- K est un ensemble fini de **clés** possibles
- ℓ est un ensemble fini **alphabet de séquence** ou **keystream alphabet**.
- $F = (f_1, f_2, \dots)$ est un **générateur de séquence** ou **keystream generator**

$$f_i: K \times P^{i-1} \rightarrow \ell$$

- Pour tout $z \in \ell$, il y a **une règle de chiffrement**, $e_z \in \mathcal{E}$ et une règle de déchiffrement correspondante $d_z \in \mathcal{D}$. Chaque $e_z: P \rightarrow C$ et $d_z: C \rightarrow P$ sont des fonctions telles que $d_z(e_z(x)) = x$ pour tout texte clair $x \in P$.
- **Un chiffrement par bloc** est un cas particulier du **chiffrement en chaîne** en prenant la valeur de clef comme constant : $z_i = K ; \forall i \geq 1$
- Un chiffrement est dit **périodique** de période d , si, et seulement si, $z_{i+d} = z_i$
- **Un chiffrement de Vigenère** de clef $K = (k_1, k_2, \dots, k_m)$ est un chiffrement à chaîne **périodique de période m** .
- Un chiffrement est dit **synchrone** si la séquence de la clef est **dépendant du texte clair** dans le cas contraire, il est **dit asynchrone**.

1.4 Les chiffrements modernes

1.4.1 Algorithme à clé secrète

Les algorithmes à clé secrète sont des algorithmes où la clé de chiffrement peut être calculée à partir de la clé de déchiffrement ou vice versa. Dans la plupart des cas, la clé de chiffrement et la clé de déchiffrement sont identiques. Pour de tels algorithmes, l'émetteur et le destinataire doivent se mettre d'accord sur une clé à utiliser avant d'échanger des messages. Cette clé doit être gardée secrète. La sécurité d'un algorithme à clé secrète repose ainsi sur la clé secrète [5] [12] [16]. Les machines cryptographiques marquent le début de l'arrivée de chiffrement à clé secrète. La clé secrète était la configuration du machine. Ce sont des machines cryptographiques utilisés pendant la deuxième guerre mondiale : l'une de plus célèbre est Enigma en 1926 :

- La machine Enigma possède 3 rotors, chaque rotor contient les 26 lettres de l'alphabet. L'ordre de grandeur de la combinatoire est donc de 26^3 , c'est à dire 17576 positions initiales des rotors.
- La combinatoire est encore augmentée par le fait que les rotors sont permutable
- Vers la fin de 1938 les 3 rotors sont choisis parmi 5, ce qui fait passer les possibilités d'arrangement des rotors de 6 à 60.

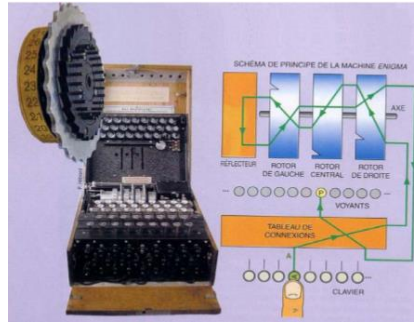


Figure 1.13 : Enigma

L'armée française s'équipe au milieu des années 1930, de matériel suédois : La machine C36 et la machine B211 qui resteront en service environ 20 ans. Des machines ont été également inventées pour faire de cryptanalyse et ce sont le père des ordinateurs actuels comme : Machine de Turing, BOMBA, MAGIC, ULTRA ...

1.4.2 Algorithme à clé publique

Les algorithmes à clé publique sont conçus de telle manière que la clé de chiffrement soit différente de la clé de déchiffrement. De plus, la clé de déchiffrement ne peut pas être calculée à partir de la clé de chiffrement. De tels algorithmes sont appelés algorithmes « à clé publique » parce que la clé de chiffrement peut être rendue publique : n'importe qui peut l'utiliser pour chiffrer un message mais seul celui qui possède la clé de déchiffrement peut déchiffrer le message chiffré résultant. Dans de tels systèmes, la clé de chiffrement est appelée clé publique, et la clé de déchiffrement est appelée clé privée ou clé secrète. Parfois, les messages seront chiffrés avec la clé privée et déchiffrés avec la clé publique ; une telle technique est utilisée pour les signatures numériques. [17] [18]

1.5 Les générateurs des nombres aléatoires

1.6 Les fonctions de hachage

1.6.1 Généralités

D'après le livre [1] [3] [12] [19], une fonction de hachage fonctionne comme une empreinte digitale, elle ne permet pas de reconstituer les autres caractéristiques d'un individu, mais permet d'identifier celui-ci. Les fonctions de hachage sont très utilisées en cryptographie puisqu'elles permettent de diminuer la quantité d'information à crypter, de mettre au point des protocoles de signature

électronique et d'authentification des messages, et de vérifier l'intégrité d'un document. Le principal critère de robustesse d'une fonction de hachage est sa résistance à des collisions. Une fonction de « hachage » consiste donc à ne signer qu'un « résumé numérique » du message. Ce résumé s'appelle « condensat » et est produit par ce qu'on appelle algorithme de condensation comme MD5 ou SHA.

Formellement, une fonction de hachage $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ est une application qui transforme une chaîne de taille quelconque en une chaîne de taille fixe n , comme illustrée sur la figure 1.14

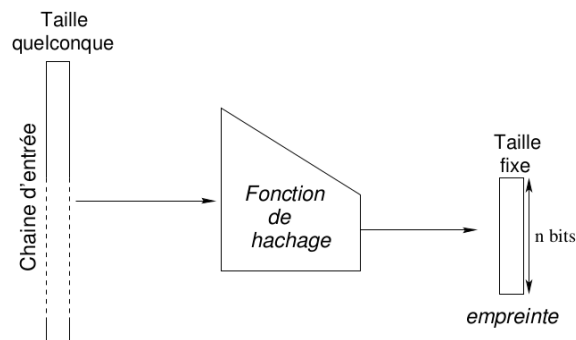


Figure 1.14 : *Le principe de fonction de hachage*

Le hachage doit tenir compte de ces différentes vulnérabilités : le premier préimage, le second préimage, et la collision. Dans la mesure où l'entrée d'une fonction de hachage peut avoir une taille quelconque n , les collisions sont inévitables. Si y est de tel que $y = H(x)$, alors x est appelé *antécédent* ou *préimage* de y et y est une *empreinte* de x . Une contrainte de base dans l'élaboration d'une fonction de hachage est l'efficacité, en tenant compte de l'algorithme de compression et le temps de calcul et aussi les différentes vulnérabilités :

- Résistance du préimage : étant donné y , on ne peut pas trouver en temps raisonnable x tel que $y = H(x)$.
- Résistance au second préimage : étant donné x , on ne peut pas trouver en temps raisonnable x' tel que $x' \neq x$ alors $H(x') = H(x)$
- Résistance à la collision : on ne peut pas trouver en temps raisonnable x' et x tel que $H(x) = H(x')$

Une fonction de *hachage unique* est une fonction qui vérifie la condition du premier préimage et second préimage.

Les fonctions basiques de la fonction de hachage sont :

– les codes de détection de manipulation (MDC pour Manipulation Detection Code) qui permettent de s’assurer de l’intégrité d’un message, à l’instar des bits de parité. On distingue encore OWHFs One Way Hash Function, et CRHF Collision Resistant Hash Function.

– les codes d’authentification de messages (MAC pour Message Authentication Code) qui gèrent à la fois l’intégrité et l’authentification de la source d’une donnée. On dit qu’une fonction de hachage est « cassée » lorsqu’il existe un algorithme permettant de trouver des collisions pour cette fonction dont la complexité est meilleure que l’attaque de Yuval. Une collision apparaît lorsque deux messages différents possèdent le même condensat, et la même empreinte. L’attaque de Yuval ou encore attaque anniversaire désigne la recherche de collisions par force brute. La complexité d’une telle attaque est $O(2^{\frac{n}{2}})$ avec n est la taille d’empreinte.

1.6.2 Construction de fonction de Hachage

- Construction de fonction de hachage de Merkle-Damgård. L’une des constructions les plus connues de fonction de hachage fait intervenir une fonction de compression [1] [12] [19].

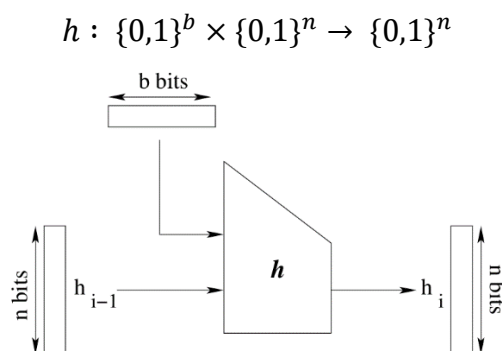


Figure 1.15 : *Fonction de compression de Merkle-Damgård.*

Le message M est découpé en blocs de b bits M_1, \dots, M_k (il faut éventuellement rajouter des bits fictifs afin que la taille de M divise exactement b).

On itère la fonction de compression h selon le modèle présenté dans la figure 1.16

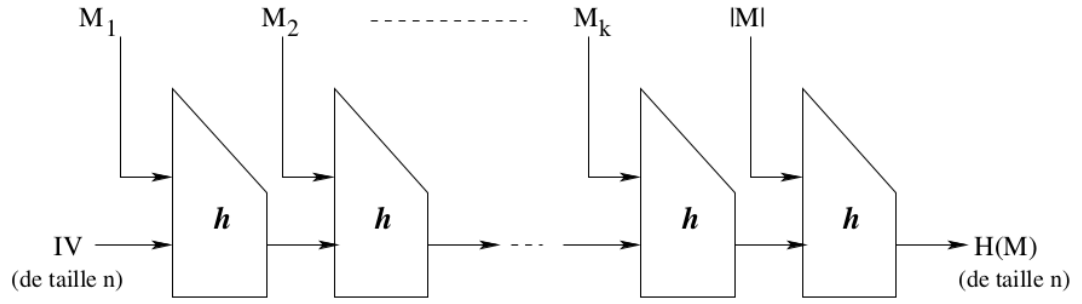


Figure 1.16 : *Construction de Merkle-Damgård*

- Construction de Davies-Meyer : C'est une fonction de compression h résistants aux collisions définit par [1] [12] :

$$h_i = E_{M_i}(h_{i-1}) \oplus h_{i-1} \quad (1.24)$$

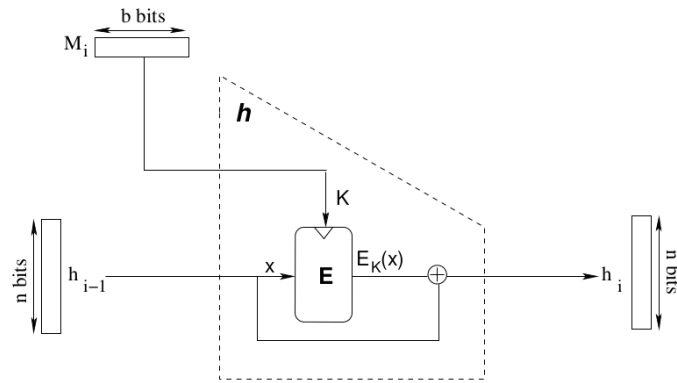


Figure 1.17 : *Construction de Davie-Meyer*

Cette construction est une cryptographie à clé secrète par bloc. Mais une attaque de la résistance à la préimage fut conçue en 1999 par Drew Dean qui exploitait la définition de point fixe sur cette construction. Aussi les fonctions de compression utilisant cette construction sont moins robustes.

- Construction de Miyaguchi-Preneel : C'est une amélioration de la construction précédente. La fonction g est une fonction d'adaptation à la taille de clef de la fonction de chiffrement E [1] [19].

$$h_i = E_g(h_{i-1}) \oplus h_{i-1} \oplus M_i \quad (1.25)$$

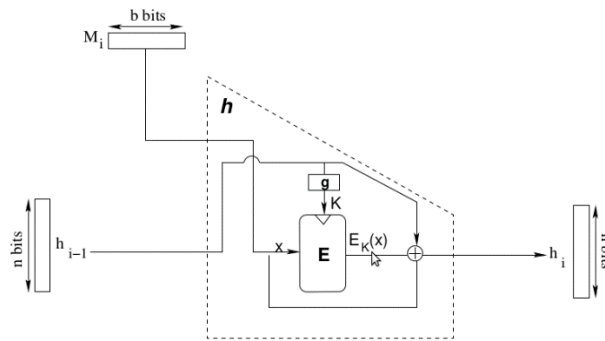


Figure 1.18 : *Construction de Miyaguchi-Preneel*

1.6.3 Exemples de fonctions de Hachage

1.6.3.1 MD5

MD5 message digest 5 a été proposé par Rivest en 1991. La taille d’empreinte vaut 128 bits. De notoriété, MD5 n’est plus considéré comme sûr car il a été cassé depuis longtemps.

1.6.3.2 SHA-1

Publié sous forme de norme en 1995 par le NIST, SHA-1 produit une empreinte de 160 bits. SHA-1 a déjà été cassé, même si l’attaque reste difficile. Néanmoins, il est encore très populaire et est encore largement utilisé.

1.6.3.3 SHA-256

C’est l’amélioration plus robuste de SHA-1 publié en 2000. En effet, la taille d’empreinte est plus longue. A l’heure actuelle, il n’y a pas encore d’attaque efficace connue contre le SHA-256.

1.6.3.4 SHA-512

La longueur d’empreinte est de 512 bits.

1.6.3.5 Whirlpool

Longueur d’empreinte 512 bits ; elle est encore considérée comme sûre car il n’y a pas encore d’attaque efficace contre elle.

1.6.3.6 RIPEMD-128

RIPE-MD pour RIPE Message Digest. RIPE signifie : RACE Integrity Primitives Evaluation. Cette fonction de hachage fournit une empreinte de 128 bits. Il existe une version renforcée avec une empreinte conséquente RIPEMD-160.

Fonction	Empreinte	Attaque anniversaire	Résistance aux collisions	Complexité de l'attaque
MD5	128 bits	$O(2^{64})$	Cassée	$O(2^{64})$
SHA-1	160 bits	$O(2^{80})$	Cassée	$O(2^{64})$
HAVAL	256 bits	$O(2^{128})$	Cassée	$O(2^{64})$
SHA-256	256 bits	$O(2^{128})$	Sûre	
Whirlpool	512 bits	$O(2^{256})$	Sûre	

Tableau 1.02: *Tableau comparatif des fonctions de hachage*

Le paradoxe est que plus l'empreinte est longue, plus il est difficile de casser l'algorithme de hachage. Or, le principe même des fonctions de hachage est de produire un condensat le plus court autant que possible pour ne pas être encombrant, et être rapide à calculer [19] [20].

1.7 Signature numérique

La norme ISO 7498-2 définit la signature numérique comme des "données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données," permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon. La mention "protégeant contre la contrefaçon" implique que seul l'expéditeur doit être capable de générer la signature [12] [20] [21].

Comme pour les algorithmes de chiffrement, on distingue deux classes de signatures :

1. les signatures symétriques qui utilisent un cryptosystème à clef secrète. Ceci permet l'utilisation de la cryptographie à clé secrète pour la génération du sceau ou code d'authentification de message. Un code d'authentification de message (MAC : Message Authentication Code) est le résultat d'une fonction de hachage à sens unique dépendant d'une clé secrète : l'empreinte dépend à la fois de

l'entrée et de la clé. On peut construire un MAC à partir d'une fonction de hachage ou d'un algorithme de chiffrement par blocs. Une façon courante de créer le MAC est l'utilisation de l'algorithme CBC (cf. Chapitre 2). La seconde méthode est aussi appelée scellement.

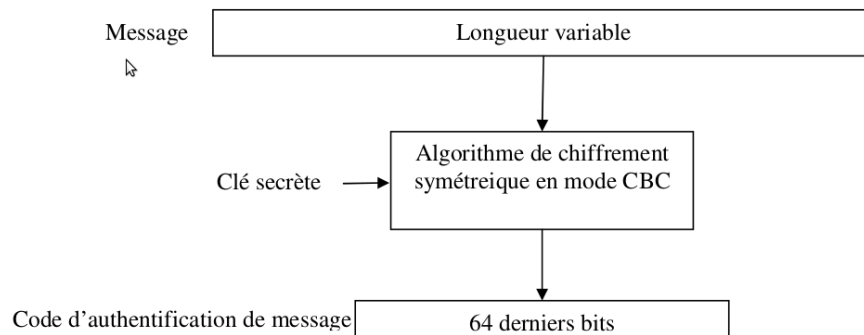


Figure 1.19 : Scellement à l'aide d'un algorithme symétrique

2. les signatures asymétriques qui utilisent un cryptosystème à clef publique et une fonction de hachage publique (sans clef) comme SHA-256 ou Whirlpool. Cette méthode consiste à calculer une empreinte du message à signer et à ne chiffrer que cette empreinte. Le calcul d'une empreinte par application d'une fonction de hachage étant rapide et la quantité de données à chiffrer étant fortement réduite, cette solution est bien plus rapide que la précédente.

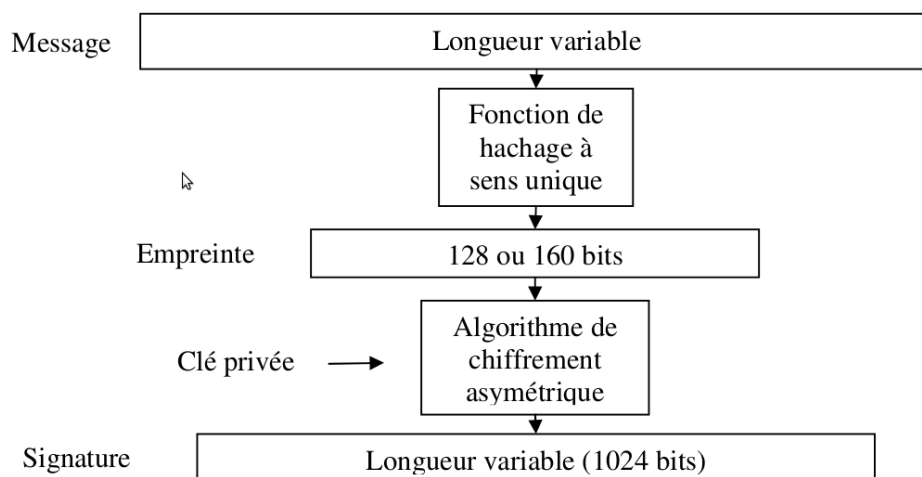


Figure 1.20 : Représentation de signature asymétrique

1.7.2 Exemples de signature à clé asymétrique

- DSS

Proposé par le NIST en 1991, puis finalement adopté en 1994, DSS (Digital Signature Standard) est la norme de signature numérique du gouvernement Américain. Elle se base sur l'algorithme DSA (Digital Signature Algorithm), qui utilise SHA comme fonction de hachage à sens unique et El Gamal pour la génération et la vérification de la signature.

- DSA

Si DSS est la norme officielle aux U.S.A., RSA est en revanche une norme de fait, qui est en pratique beaucoup plus utilisés pour la signature que DSA.

1.7.3 Scellement ou signature à clé symétrique

- Keyed-Hash

Un exemple simple de cette façon de procéder est de prendre pour MAC des valeurs du type $H(\text{secret}, \text{message})$, $H(\text{message}, \text{secret})$ ou $H(\text{secret}, \text{message}, \text{secret})$. Ces méthodes, présentées en 1992 par Gène Tsudik s'appellent respectivement méthode du préfixe secret, du suffixe secret et de l'enveloppe secrète. Elles ont une sécurité limitée [19] [21].

- HMAC

Une méthode de calcul de MAC à base de fonction de hachage plus élaborée et plus sûre est HMAC. La méthode HMAC peut être utilisée avec n'importe quelle fonction de hachage itérative telle que MD5, SHA-1 ou encore RIPE-MD. Soit H une telle fonction, K le secret et M le message à protéger. H travaille sur des blocs de longueur b octets (64 en général) et génère une empreinte de longueur en octets (16 pour MD5, 20 pour SHA et RIPE-MD-160). Il est conseillé d'utiliser un secret de taille au moins égale à l octets. On définit deux chaînes, $ipad$ (inner pudding data) et $opad$ (outer pudding data), de telle façon que [19] :

$$ipad = l'octet\ 0 \times 36 \text{ répété } b \text{ fois} \quad (1.30)$$

$$opad = l'octet\ 0 \times 5C \text{ répété } b \text{ fois} \quad (1.31)$$

MAC se calcule alors suivant la formule suivante :

$$HMAC_K(M) = H(K \text{ opad}, H(K \text{ ipad}, M)) \quad (1.32)$$

1.8 Cas pratique des signatures :

Le chiffrement de l’empreinte (par la fonction de hachage du message) permet d’avoir la signature [20] [21] [22]

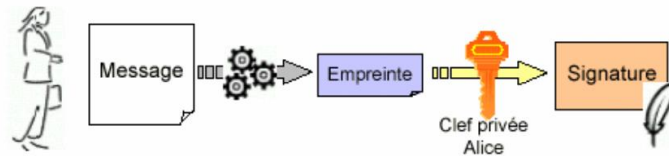


Figure 1.21 : Figure représentative du fonctionnement de la signature

Bob prend la signature et demande la clé public d’Alice, en le déchiffrant à partir de la clé normalement on devrait avoir l’empreinte du message. Dans le cas contraire on est sûr que le message ne vient pas d’Alice.

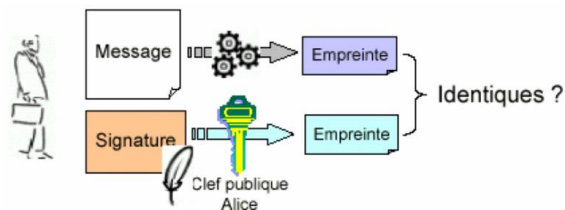


Figure 1.22 : Vérification de signature

1.9 Conclusion

La cryptographie est une science qui sert à chiffrer les informations. Son rôle principal est le C.A.I.N : Confidentialité, Authenticité, Intégrité et Non répudiation. La cryptographie est un du domaine de cryptologie qui consiste à rendre les informations secrètes. Ce domaine se divise encore en 3 parties : cryptographie classique, moderne et quantique. La cryptographie ancienne ou classique se base sur les algorithmes de permutation et substitution tandis que la cryptographie moderne se divise en deux : par blocs et système à flot. Pour assurer la confidentialité et l’intégrité, la notion de hachage et de signature doit être considérée et étudiée de manière concise.

CHAPITRE 2 ETUDE APPROFONDIE SUR LA CRYPTOGRAPHIE MODERNE

Le principe de Kerckhoff [12] [24] stipule que même en connaissant les différents protocoles et échange de clé dans le cryptosystème, il est impossible de reconnaître le texte en clair sans avoir la clé secrète. On peut formuler son principe par les 3 conditions suivantes :

- La sécurité repose sur la clé et non sur le secret de la méthode employée.
- Le déchiffrement de la clé doit être pratiquement impossible
- Trouver la clé à partir du clair et du chiffré doit être pratiquement impossible.

Pour assurer ce principe de Kerckhoff [25], les algorithmes modernes doivent assurer ces deux principes fondamentaux :

- Principe de confusion : rendre la relation entre le texte chiffré et la clé secrète la plus complexe que possible.
- Principe de Diffusion : toute modification (comprendre la plus faible possible) faite sur la texte clair doit se répercuter sur tout le texte chiffré.

Dans ce chapitre, nous allons voir la possibilité d'étudier la cryptographie moderne.

2.1 Cryptographie symétrique

Il y a donc une relation entre la clé de chiffrement et la clé de déchiffrement [16] [26] :

- En générale, elles sont égales : $K = K'$.
- Généralement, un seul algorithme est utilisé pour le chiffrement et le déchiffrement : $D = E$
- Le secret de la clé de déchiffrement/chiffrement est donc partagé entre émetteur A et récepteur B.

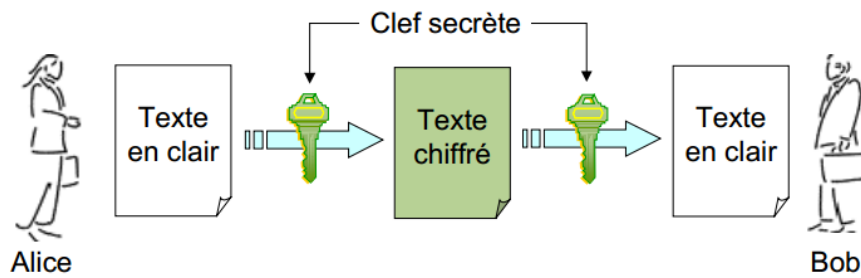


Figure 2.01 : *Chiffrement symétrique*

2.1.2 Les chiffrements à flot

Dans un cryptosystème à flot, le cryptage de message se fait en caractère par caractère. C'est aussi un chiffrement à la volée ou One Time Padding (OTP) ou masque jetable. Le principe consiste à créer un aléas ou keystream à partir d'une registre à décalage, on note ces keystream une suite de bit r_i et le message clair ou plaintext est une suite de bit m_i . Ces keystream sont obtenus en effectuant un décalage à chaque cycle de l'un oscillateur qui se répète à un nombre de fois bien déterminé [1].

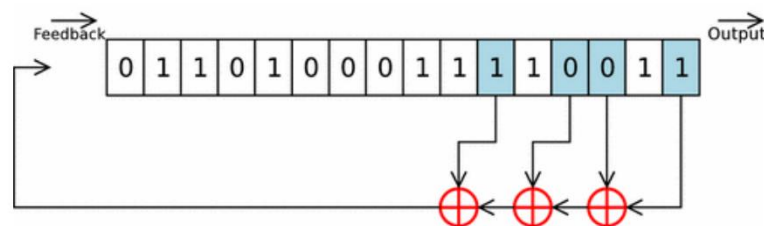


Figure 2.02 : *Exemple de registre de décalage*

Le texte chiffré ou ciphertext est obtenu à partir d'une opération xor (ou exclusif) entre le keystream et le message chiffré car le destinataire possède une copie du masque.

$$\text{Ciphertext} = \text{Keystream} \text{ xor } \text{Plaintext} \quad (2.01)$$

Opération « ou exclusif » : \oplus

	0	1
0	0	1
1	1	0

Figure 2.03 : Représentation de l'opérateur ou exclusif en OTP

L'avantage de l'opérateur xor, il suffit de faire la même opération pour avoir le message en clair c'est-à-dire :

$$\text{Plaintext} = \text{Keystream} \text{ xor } \text{CipherText} \quad (2.02)$$



Figure 2.04 : Représentation de système à flot dans un canal sécurisé

L'efficacité de l'algorithme dépend de la génération de keystream. Différents algorithmes ont été proposés [26] : LFSR, RC4 (le fameux algorithme de cryptage utilisé en Wifi inventé par Rivest), Py (Biham), E0 (le fameux algorithme de cryptage utilisé en Bluetooth), A5/1 ou A5/2 ou A5/3 (le fameux algorithme de cryptage utilisé en réseau mobile 2G/2.5G/2.75G)

2.1.3 Les chiffrements par blocs

Les entiers r et k sont respectivement la taille (en bits) d'un bloc et celle de la clef. Un chiffrement par bloc est une paire de fonctions F et G associant à toute clé K avec $\mathbb{K} = \{0,1\}^k$ une permutation F_k de l'espace de blocs $\{0,1\}^r$, et la permutation inverse G_k [25] [27] .

Pour ce mode de cryptage, le texte clair est fractionné en blocs de même longueur à l'aide d'une clé unique. Les algorithmes de chiffrement par blocs sont en général construits sur un modèle itératif. Ce modèle emploie une fonction F qui prend en paramètres une clé K et un message de longueur r bits. La fonction F est répétée un certain nombre de fois, on parle de ronde. A chaque ronde, la clé k utilisée est changée et le message qu'on chiffre et le résultat de l'itération précédente

$$\begin{aligned} C_1 &= F(k_1, M) \\ C_2 &= F(k_2, C_1) \\ &\dots \\ C_r &= F(k_r, C_{r-1}) \end{aligned}$$

Figure 2.05 : *Fonction de chiffrement par bloc*

L'émetteur et le destinataire partagent une clé K secrète. L'algorithme qui engendre les clefs k_i à partir de K se nomme l'algorithme de cadencement des clefs. La fonction F doit être inversible, ce qui veut dire que pour toute clef k et message M , il faut être capable de recalculer M à partir de $F(k, M)$, sinon le déchiffrement est impossible et on ne dispose d'aucun algorithme utilisable. Il existe donc une fonction G vérifiant $G(k, F(k, M)) = M$ et que F est une permutation. La sécurité d'un algorithme de chiffrement par blocs réside principalement dans la conception de l'algorithme de cadencement des clefs et la robustesse de la fonction F . Si l'algorithme de cadencement est mal élaboré, les k_i peuvent être déductibles les unes des autres. La fonction F doit donc être difficile à inverser sans connaître la clef k ayant servie dans le calcul de $C = F(k, M)$. En d'autres termes, connaissant seulement C , F et G , il est impossible retrouver le message M seulement en effectuant une recherche exhaustive de la clef. Les caractéristiques de ces systèmes sont en général liés à leur très forte sensibilité à la dépendance inter-symboles, ainsi qu'à leur mécanisme de propagation d'erreurs. Toute erreur commise sur un bloc de texte clair ou chiffré peut perturber gravement le chiffrement/déchiffrement de ses voisins.

Dans cette figure, on prend n bloc de message M ayant r bits.

$$M = M_1 M_2 \dots M_n \text{ avec } |M_i| = r \text{ bits}$$

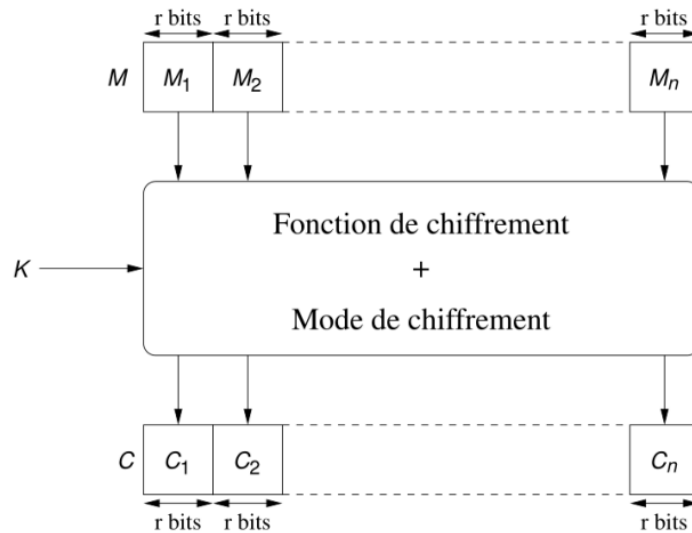


Figure 2.06 : *Principe de fonctionnement de chiffrement par bloc*

Le chiffrement par bloc le plus populaire sont [28] : DES, AES, RC6, IDEA, BLOWFISH, TOWFISH...

2.2 Le mode de chiffrement

Toute implantation d'un système de chiffrement doit donc choisir un mode d'opération. Quatre modes (ECB, CBC, CFB et OFB) ont été normalisées en 1980 par le NIST (anciennement NBS : National Bureau of Standards, organisme américain de normalisation). Différents modes ont été proposés dans divers contextes, et le NIST a commencé en 2001 une nouvelle sélection comme le CTR, CTS, XTS.... Mais le mode de chiffrement de base est ces 4 modes cités précédemment [1] [25] [27] [29].

2.2.1 ECB

2.2.1.1 Définition

ECB signifie Electronic CodeBook. Le message est découpé en blocs de taille b . Chaque bloc est chiffré séparément par E_k , le chiffrement se fait par la concaténation des blocs obtenus. Le déchiffrement se fait de façon similaire. Le message est découpé en blocs de taille b . Chaque bloc est déchiffré séparément par D_k , le texte clair est la concaténation des blocs obtenus [27] [29].



Figure 2.07 : *Chiffrement ECB*

2.2.1.2 Propriété

- Expansion

C'est un mode sans expansion, mais qui ne marche qu'avec des messages dont la longueur est un multiple de b . On peut généraliser le mode ECB aux autres longueurs de message, par exemple en complétant le dernier bloc avec un bit à 1 suivi du nombre adéquat de bits à 0. Le mode ECB a alors une expansion de 1 à b bits. La technique appelée Ciphertext stealing permet de chiffrer sans expansion les messages de longueur au moins b . Appelons m_{n-1} et m_n les deux derniers blocs, le dernier pouvant être incomplet ; leur chiffrement donne les deux blocs C_n et C_{n-1} tel qu'il existe une valeur C' avec $E_k(m_{n-1}) = C_n || C'$ et

$$D_k(C_{n-1}) = m_n || C'$$

- Performance

Le mode ECB peut être totalement parallélisé : le chiffrement d'un bloc ne dépend pas du chiffrement des autres.

- Résistance aux erreurs de transmission

Si un bloc c_i est modifié, seul le bloc m_i correspondant sera modifié.

Si un nombre de bits multiple de b est perdu par la transmission, seuls les blocs correspondants sont perdus.

2.2.1.3 Avantages de ECB

Les principaux avantages de l'ECB sont :

- Accès rapide à une zone quelconque du chiffré.
- Possibilité de déchiffrer qu'une seule partie.
- Simple à réaliser
- Blocs chiffrés indépendamment

- Erreurs affectent un seul bloc

2.2.1.4 Inconvénients de l'ECB

L'ECB montre également quelques inconvénients comme :

- Deux blocs identiques auront le même chiffré.
- Sensible aux attaques par jeu.
- Permet donc de construire un « dictionnaire de code » avec les correspondances clair/chiffré (codebook)
- Correspondance clair/chiffré sans connaître la clé

2.2.1.5 Conclusion

ECB ne possède pas de sécurité robuste donc il ne faut pas l'utiliser dans les applications cryptographiques

2.2.2 CBC

CBC signifie Cipher Block Chaining. Alice et Bob ont au préalable convenu d'une valeur publique IV faisant b bits [1] [25]. Le message est découpé en blocs de taille b . Le chiffrement d'un bloc se calcule en chiffrant par E_k l'opérateur OU bit-à-bit du bloc clair et du bloc chiffré précédent. La valeur IV sert de chiffré précédant le bloc 0. Le déchiffrement se fait de façon similaire. Un bloc du clair est obtenu en déchiffrant avec D_k puis en faisant l'opérateur ou bit-à-bit avec le bloc chiffré précédent.

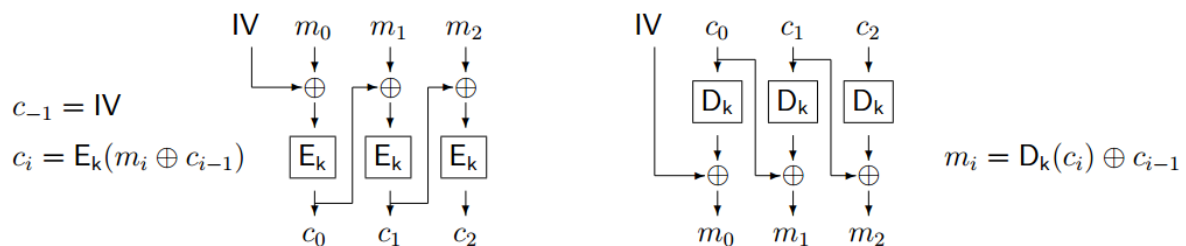


Figure 2.08 : Chiffrement et déchiffrement en mode CBC

2.2.2.2 Propriétés

- Expansion.

Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion, mais qui ne marche qu'avec des messages dont la longueur est un multiple de b . On peut généraliser le mode CBC aux autres longueurs de message, de la même façon que le mode ECB.

- Performance.

Le chiffrement CBC ne peut être parallélisé, mais le déchiffrement CBC peut être totalement parallélisé.

- Résistance aux erreurs de transmission.

Si un bloc c_i est modifié, seul les blocs m_i et m_{i+1} correspondants seront modifiés. Si un nombre de bits multiple de b est perdu par la transmission, seuls les blocs correspondants sont perdus.

2.2.2.3 Sécurité

Le mode CBC ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. En revanche, il a une bonne sécurité face aux attaques passives. On peut déduire de l'information sur le message dès que le chiffré contient deux blocs égaux :

$$\text{Si } c_i = c_j \text{ alors } m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$$

Si les c_i peuvent être considérés comme aléatoires, alors le paradoxe des anniversaires affirme qu'une collision apparaît lorsque $2^{b/2}$ blocs ont été chiffrés.

Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV aléatoire différente à chaque fois. En revanche, si l'attaquant a un contrôle sur IV, il ne faut pas chiffrer plusieurs messages avec la même clef. Comme conseil, il faut donc d'utiliser $E_k(IV)$ au lieu de IV, mais alors le déchiffrement utilise E_k et D_k . Le mode CBC est le mode utilisé dans la plupart des applications des systèmes de chiffrement par bloc.

2.2.2.4 Avantages du CBC

Les principaux avantages de CBC sont :

- Le morceau en clair est XOR-é avec le bloc précédent.
- Un même bloc peut donc avoir plusieurs chiffrés.
- Plus sûr que le mode ECB
- Vecteur d'initialisation C0 implique que deux textes identiques sont chiffrés différemment

2.2.2.5 Inconvénients du CBC

Le CBC présente également quelques inconvénients comme :

- Erreur dans un bloc se répercute dans les autres blocs
- Mode le plus utilisé

2.2.2.6 Schéma comparatif entre ECB et CBC

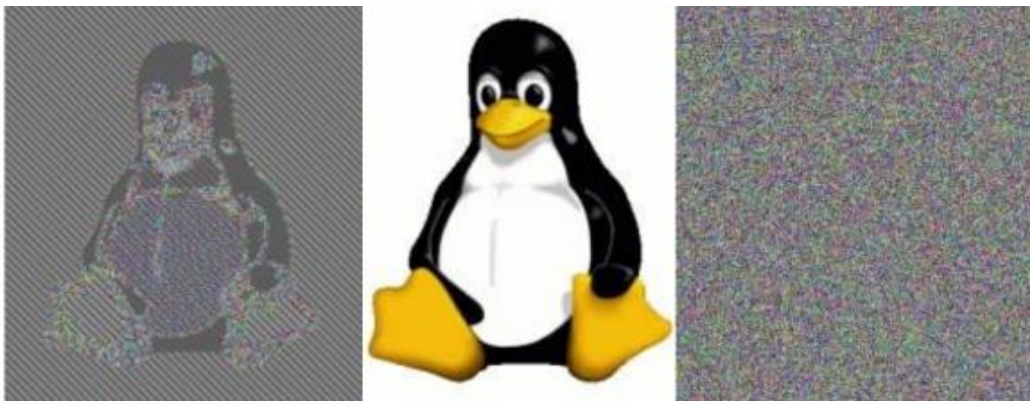


Figure 2.09 : *Comparaison ECB et CBC à partir de l'image de Tux*

On voit bien que l'image de Tux au milieu est l'image originale, celle à gauche, l'image chiffrée en utilisant l'ECB et celle à droite l'image chiffrée en utilisant le CBC. Contrairement à l'ECB, avec le CBC, on ne peut pas retrouver les motifs originaux.

2.2.2.7 Conclusion

Le CBC présente une bonne sécurité par rapport à l'ECB, n'affaiblit pas le cryptosystème.

2.2.3 CFB

Le mode CFB (Cipher FeedBack) fabrique un registre à décalage en mode CTAK (CipherText Auto Key), pour obtenir un chiffrement de flot. Le mode CFB est paramétré par un entier ℓ inférieur à b et utilise un registre de b bits, initialisé par une valeur publique IV (Initial Value) convenable à l'avance. Le message est découpé en blocs de taille ℓ . A chaque coup d'horloge, on calcule l'image du registre par E_k , dont on extrait ℓ bits qu'on appelle r_i . Le bloc est chiffré par l'opérateur ou exclusif : $c_i = m_i \oplus r_i$. La nouvelle valeur du registre est obtenue en faisant un décalage de ℓ bits, et en y entrant la valeur c_i [1] [29].

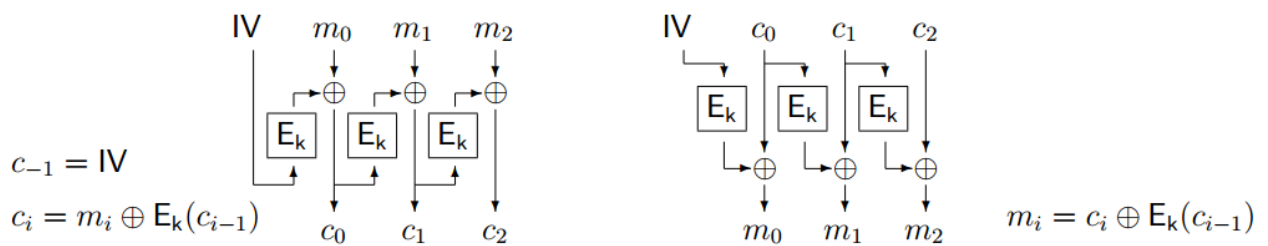


Figure 2.10 : Le principe du mode de chiffrement CBC avec $\ell = b$

2.2.3.2 Propriétés

- Expansion :

Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion : si la longueur du message n'est pas un multiple de ℓ , on utilise la méthode tronquée pour le dernier r_i

- Performances :

Le chiffrement CFB ne peut être parallélisé, mais le déchiffrement CFB peut être totalement parallélisé. Le chiffrement et le déchiffrement utilisent tous deux la fonction E_k , et donc le temps de calcul de D_k aucune influence sur les performances de CFB. En revanche, il faut un appel à E_k tous les ℓ bits du message.

- Résistance aux erreurs de transmissions :

Si un bloc c_i est modifié, seul le bloc m_i et quelques autres seront modifiés. Si un nombre de bits multiple de ℓ est perdu par la transmission, seuls les blocs correspondants sont perdus.

2.2.3.3 Sécurité

Le mode CFB ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. Comme le mode CBC, c'est à partir du chiffrement de $2^{b/2}$ blocs qu'un attaquant obtient éventuellement de l'information sur le message (grâce au paradoxe des anniversaires). Les contraintes de sécurité sur la fonction E_k sont d'autant moins sévères que ℓ est petit. Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV différente à chaque fois, si cette valeur est choisie aléatoirement. On conseille donc dans ce contexte l'utilisation de $E_k(IV)$ au lieu de IV.

2.2.3.4 Avantages du mode CFB

Le CFB présente plusieurs avantages comme :

- Pas besoin de fonction de déchiffrement
- Parfois plus rapide
- Utilisé dans les réseaux

2.2.3.5 Inconvénients du mode CFB

Le principal inconvénient est que l'algorithme est moins sûr.

2.2.3.6 Conclusion

Le CFB est un peu moins sûr que CBC, mais beaucoup plus rapide.

2.2.4 OFB

2.2.4.1 Définition

Le mode OFB (Output FeedBack) fabrique un registre à décalage en mode KAK (Key Auto Key), pour obtenir un chiffrement de flot. Le mode OFB est paramétré par un entier ℓ inférieur à b et utilise un registre de b bits, initialisé par une valeur IV (Init Value) convenable à l'avance. Le message est découpé en blocs de taille ℓ . A chaque coup d'horloge, on calcule l'image du registre par E_k , dont on extrait ℓ bits qu'on appelle r_i . Le bloc est chiffré par ou exclusif : $c_i = m_i \oplus r_i$. La

nouvelle valeur du registre est obtenue en faisant un décalage de ℓ bits, et en y entrant la valeur r_i [25][29].



Figure 2.11 : Principe du mode de chiffrement OFB avec $\ell = b$

2.2.4.2 Propriétés :

- Expansion :

Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion : si la longueur du message n'est pas un multiple de ℓ , on utilise une valeur tronquée pour le dernier r_i .

- Performance :

Ni le chiffrement, ni le déchiffrement OFB ne peuvent être parallélisés.

La séquence des s_i peut-être pré-calculée avant de connaître le message.

Le chiffrement et le déchiffrement utilisent tous les deux, la fonction E_k , et donc le temps de calcul de D_k n'a aucune influence sur les performances du mode OFB. En revanche, il faut un appel à E_k tous les ℓ bits du message.

- Résistance à des erreurs de transmissions :

Si un bloc C_i est modifié, seul le bloc m_i sera modifié. Si quelques bits sont perdus par la transmission, toute la fin du message est perdue.

2.2.4.3 Sécurité :

Le mode OFB ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. Les contraintes de sécurité sur la fonction E_k sont d'autant moins sévères que ℓ est petit.

Pour $\ell = b$, si on utilise une fonction E_k indistinguishable d'une permutation aléatoire, alors la valeur s_i fera un cycle avant d'avoir parcouru les 2^b valeurs possibles. Il sera donc préférable d'utiliser un système de chiffrement par blocs tel que chaque permutation E_k a un unique cycle. Il n'y a pas

d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV différente à chaque fois, si cette valeur est choisie aléatoirement. Il est conseillé donc dans ce contexte l'utilisation de $E_k(IV)$ au lieu de IV.

2.2.4.4 Les avantages de OFB

Les principaux avantages de l'OFB sont :

- Même avantages que CFB.
- Une erreur d'un bit affecte uniquement ce bit.
- Moins de câblage
- Utilisé dans les satellites

2.2.4.5 Inconvénient de l'OFB

L'inconvénient d'utilisation de l'OFB est que La perte ou l'ajout d'un bit est irrécupérable.

2.2.4.6 Conclusion

L'OFB est une variante de CFB avec une sureté équivalente à celle de CFB

2.3 Chiffrement à clé publique

Avec les algorithmes asymétriques, les clés de chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre [30] [31].

- Idée : On peut donc rendre l'une des deux publique tandis que l'autre reste privé. C'est pourquoi on parle de chiffrement à clé publique.
- Si la clé publique sert au chiffrement tout le monde peut chiffrer un message, que seul le propriétaire de la clé privée pourra déchiffrer.

Ainsi, on assure la confidentialité.

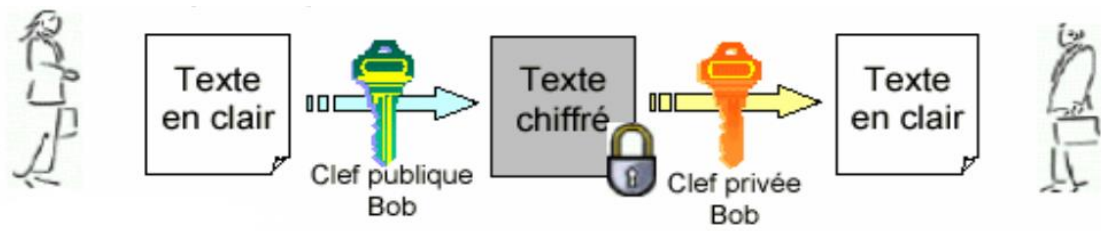


Figure 2.12 : *Chiffrement asymétrique*

Propriété

- On choisit D'_k et E_k telles qu'il est très difficile de déduire D'_k de la connaissance de E_k .
- Il faut trouver une fonction dont la fonction inverse est difficile à déterminer.
- On peut donc rendre E_k publique (notion de clé publique) connue de tous dans un annuaire car c'est très pratique. Tout le monde peut chiffrer $C = E_k(M)$
- Par contre la clé D'_k (la clé privée) reste secrète et particularise chaque utilisateur. Seul le destinataire peut déchiffrer $M = D'_k(C)$
- Propriété complémentaire (très utile) d'un système à clé publique : la commutativité.

$$D'_k(E_k(M)) = E_k(D'_k(M)) = M.$$

2.3.2 RSA

Ronald Rivest, Adi Shamir et Leonard Adleman ont publié leur chiffrement en 1978 dans « A Method for Obtaining Digital Signatures and Public-key Cryptosystems ». Ils utilisent les congruences sur les entiers et le petit théorème de Fermat, pour obtenir des fonctions à sens unique, avec brèche secrète (ou porte dérobée). Tous les calculs se font modulo un nombre entier n qui est le produit de deux nombres premiers. Le petit théorème de Fermat joue un rôle important dans la conception du chiffrement [2][3] [32].

2.3.2.1 Génération de clé

Choisir deux nombres entiers premiers p et q , de l'ordre de 100 chiffres au minimum, pour rendre la factorisation hors de portée, même avec les meilleurs ordinateurs [33] [34].

1) Calculer $n = p * q$

2) Calculer $m = (p - 1) * (q - 1)$

3) Choisir un nombre entier e tel que $e > 2$ et $\text{pgcd}(e, n) = 1$

4) Calculer d tel que $d * e \bmod m = 1$

On prendra comme clé d publique e et n et comme clé privée d et n

2.3.2.2 Chiffrement

Connaissant la clé publique e et n , et le message à chiffrer M . On peut le chiffrer de la manière suivante : le message doit être remplacé par un chiffre. Ensuite on découpera le message par bloc de x longueurs avec $x < n$. Le bloc B est chiffré par la formule :

$$C = B^e \bmod(n) \quad (2.03)$$

2.3.2.3 Déchiffrement

Pour le déchiffrement, il faut pratiquer quasiment la même façon avec le chiffrement mais en utilisant une formule inverse :

$$b = C^d \bmod(n) \quad (2.04)$$

2.3.3 Cryptosystème d'El Gamal

Alice veut transmettre un message M à Bob [3] [25] [35] :

- Bob choisit un grand nombre premier p_b (grand devant M) et deux nombres g_b et α_b inférieurs à p .
- Bob calcul $\beta_b = g_b^{\alpha_b}$, alors (β_b, g_b, p_b) est sa clé publique, α_b sa clé secrète.
- Alice choisit k_a et calcule : $y_1 = (g_b^{k_a}) \bmod p_b$ et $y_2 = \beta_b^{k_a} M$
- La paire (y_1, y_2) est le message chiffré qu'Alice envoie à Bob :

$$e_k(M, k_a) = (y_1, y_2) \quad (2.05)$$

- Pour déchiffrer, Bob calcule :

$$M = y_2 (y_1^{\alpha_b})^{-1} \bmod p_b \quad (2.06)$$

En effet $y_2 = \beta_b^{k_a} M$ et : $y_1 = (g_b^{k_a}) \bmod p_b$ donc $M = \beta_b^{k_a} M ((g_b^{k_a})^{\alpha_b})^{-1} \bmod p_b = (g_b^{\alpha_b})^{k_a} M ((g_b^{k_a})^{\alpha_b})^{-1} = M \text{ (c.q.f.d.)}$

2.3.4 *Cryptosystème PGP*

Le PGP (Pretty Good Privacy) est un cryptosystème inventé par Phil Zimmermann en 1991. Il combine à la fois les meilleures fonctionnalités de la cryptographie à clé privée et de la cryptographie à clé publique. PGP est donc un système hybride [3] [28] [36] [37].

2.3.4.1 Chiffrement

Quand un utilisateur chiffre du texte clair avec PGP, PGP compresse d'abord le texte clair. La compression de données économise le temps de transmission des données et de l'espace disque et, ce qui est important, renforce la sécurité cryptographique. La plupart des techniques de cryptanalyse exploitent les redondances trouvées dans le texte clair pour craquer le texte chiffré. La compression réduit ces redondances dans le texte clair, ce qui augmente grandement la résistance à la cryptanalyse.

PGP crée ensuite une clé de session, qui est une clé secrète et elle ne sert qu'une fois. Cette clé est un nombre aléatoire généré à partir des mouvements aléatoires de votre souris et des touches du clavier sur lesquelles vous tapez. Cette clé de session fonctionne avec un algorithme de chiffrement conventionnel très sûr et rapide qui chiffre le texte clair ; le résultat est le texte chiffré. Une fois que les données sont chiffrées, la clé de session est elle-même chiffrée avec la clé publique du destinataire. Cette clé de session chiffrée par la clé publique est transmise avec le texte chiffré au destinataire.

2.3.4.2 Déchiffrement

Le déchiffrement fonctionne de la manière inverse. La copie de PGP du destinataire utilise la clé privée de celui-ci pour retrouver la clé de session temporaire, que PGP utilise ensuite pour déchiffrer le texte chiffré de manière conventionnelle. La combinaison des deux méthodes de chiffrement (IDEA, RSA) associe la commodité du chiffrement à clé publique avec la vitesse du chiffrement conventionnel. Le chiffrement conventionnel est environ 1000 fois plus rapide que le chiffrement à clé publique. Le chiffrement à clé publique fournit quant à lui une solution aux problèmes de distribution de la clé et de transmission des données. Utilisées toutes les deux, la performance et la distribution de la clé sont améliorées sans aucun sacrifice sur la sécurité.

2.3.5 Cryptosystème RABIN

Le cryptosystème de Rabin est un cryptosystème asymétrique basé sur la difficulté du problème de la factorisation (comme RSA). Il a été inventé en 1979 par Michael Rabin : c'est le premier cryptosystème asymétrique dont la sécurité se réduit à la difficulté calculatoire de la factorisation d'un nombre entier. Le cryptosystème de Rabin a l'avantage de disposer d'une preuve de difficulté aussi grande que la factorisation d'entiers, preuve qui n'existe pas encore pour RSA. Il a par contre un désavantage dû à un non-déterminisme : une sortie produite par la fonction présente dans le cryptosystème peut être le résultat de quatre entrées distinctes [25] [37].

2.3.5.1 Génération de clé

- Choisir deux grands nombres premiers, p et q , au hasard.
- Calculer $n = p * q$, ce qui fait de n la clé publique. Les nombres premiers p et q constituent la clé privée.

2.3.5.2 Chiffrement

Pour le chiffrement, seule la clé publique, n , est utilisée. On produit le texte chiffré à partir du texte en clair m comme suit. Soit $P = \{0, \dots, n-1\}$ l'espace des textes en clair possibles (tous des nombres) et posons $m \in P$ comme étant le texte en clair. Le texte chiffré c se détermine comme suit :

$$c = m^2 \bmod n \quad (2.07)$$

2.3.5.3 Déchiffrement

Pour déchiffrer, la clé privée est nécessaire. Le processus est comme suit. Les racines carrées sont calculées par :

$$m_p = \sqrt{c} \bmod p \text{ et } m_q = \sqrt{c} \bmod q \quad (2.08)$$

L'algorithme d'Euclide étendu permet de calculer y_p et y_q , tels que $y_p \cdot p + m_q \cdot q = 1$. On invoque alors le théorème des restes chinois pour calculer les quatre racines carrées $+r$; $-r$; $+s$ et $-s$. Les quatre racines carrées sont dans l'ensemble :

$$r = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod n ;$$

$$-r = n - r; s = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod n; -s = n - s \quad (2.10)$$

2.3.5.4 Remarque

Quatre messages clairs différents peuvent produire le même message crypté. En d'autres termes, c'est une fonction de « quatre-en-un ».

2.3.6 *Cryptosystème basé sur les codes d'erreurs*

2.3.6.1 Cryptosystème de Mc Eliece

Le cryptosystème de McEliece est un schéma de chiffrement asymétrique, inventé en 1978 par Robert McEliece. Ce système, reposant sur un problème difficile de la théorie des codes [19] [25]. Le cryptosystème de McEliece consiste en trois algorithmes : un algorithme probabiliste de génération des clefs qui produit une clef secrète et une clef publique, un algorithme (probabiliste) de chiffrement et un algorithme (déterministe) de déchiffrement. Tous les utilisateurs partagent des paramètres de sécurité : n, k, t .

- Génération de la clé

Pour générer une clé, il faut suivre les étapes suivantes :

- Sélectionner aléatoirement un (n, k) -code linéaire C capable de corriger t erreurs. Ce code doit posséder un algorithme de décodage efficace.
- Alice génère une matrice génératrice G pour le code C (matrice $k \times n$)
- Sélectionner aléatoirement une matrice binaire $k \times k$ non singulière S .
- Sélectionner aléatoirement une matrice de permutation $P : n \times n$
- Calculer la matrice $\hat{G} = S G P$. Celle-ci est une matrice $k \times n$
- La clef publique est (\hat{G}, t) ; la clef privée est (S, G, P)
- Chiffrement

Pour chiffrer une suite binaire m de longueur :

- Calculer le vecteur $c' = m\hat{G}$

- Générer un vecteur d'erreur e de poids t
- Calculer le chiffré :

$$c = c' + e \quad (2.11)$$

- Déchiffrement
 - Calculer $c' = cP^{-1}$
 - Utiliser l'algorithme de décodage de C pour décoder \hat{c} en un mot \hat{m}
 - Calculer :

$$m = \hat{m}.S^{-1} \quad (2.12)$$

2.3.6.2 Cryptosystème de Niederreiter

Le cryptosystème de Niederreiter est une variation de McEliece cryptosystème développé en 1986 par Harald Niederreiter. Il utilise la même idée, le codage linéaire et le code correcteur d'erreur par le biais de la matrice de parité H et possède une sécurité équivalente que McEliece. L'encryptions est beaucoup plus rapide et il y a possibilité d'utiliser une signature pour ce genre de cryptage. L'algorithme basique de Niederreiter est cassé sauf en utilisant le Goppa code [38] [39] [40].

- Génération de clé :
 - Alice choisit un couple d'entier (n, k) -linéaire Goppa Code, G , capable de corriger t erreur. L'avantage de Goppa Code c'est qu'il possède un algorithme efficace de décodage
 - Alice génère une matrice de parité $H : (n - k) \times k$ à partir du code G
 - Alice choisit aléatoirement une matrice binaire $(n - k) \times (n - k)$ non singulière S .
 - Alice choisit aléatoirement une matrice de permutation $P : n \times n$
 - Alice calcul la matrice $(n - k) \times n$, $H^{pub} = SHP$
 - La clé publique de Alice est (H^{pub}, t) et la clé privée est (S, H, P)
- Chiffrement

On suppose que Bob veut chiffrer le message m à Alice avec la clé publique (H^{pub}, t)

- Bob code le message m sous forme de string de longueur n et de poids au plus t
- Le message chiffré est donc :

$$c = H^{pub}m^T \quad (2.13)$$

- Déchiffrement

Maintenant , Alice veut retrouver le message m à partir du message chiffré $c = H^{pub}m^T$.

- Alice calcul : $S^{-1}c = HPm^T$
- Alice applique à G l'algorithme de décodage par Syndrome pour déterminer Pm^T
- Alice calcul le message m par la formule :

$$m^T = P^{-1}Pm^T \quad (2.14)$$

Les paramètres recommandés sont : $n = 1024, t = 38, k = 644$

2.3.7 Cryptosystème Menezes-Vanstone

Ce système basé sur le groupe des points d'une courbe elliptique définie sur un corps fini F_q est une variante du cryptosystème El Gamal. Une courbe elliptique est l'ensemble des points de $K \times K$ vérifiant l'équation [3] [25].

$$y^2 = x^3 + ax + b \text{ avec } a, b, c \in K \quad (2.15)$$

Soit E une courbe elliptique définie sur le corps fini $F_p \simeq Z/pZ$ avec p supérieur à 3 telle que E contienne une sous-groupe cyclique H engendré par un point quelconque B de E dans lequel le problème LDP, soit difficile :

$$P = F_p^* \times F_p^*; \text{ les textes claires}$$

$$C = E \times F_p^* \times F_p^*; \text{ les textes chiffrés}$$

Posons $|H|$ le cardinal de H . et $K = \{(E, \alpha, a, \beta) : \beta = a \cdot \alpha\}$

K est l'espace de clé où $\alpha \in E$. Les valeurs α, β sont publiques et $a \in Z/|H|Z$ est secret et choisi au hasard. Pour $K = (E, \alpha, a, \beta)$ et pour un nombre aléatoire secret $k \in Z/|H|Z$ et pour $x = (x_1, x_2) \in P = F_p^* \times F_p^*$, on chiffre le message x à l'aide du clé K et le nombre aléatoire k en posant

$$\mathcal{H}_K(x, k) = (y_0, y_1, y_2)$$

$$\text{Où } \begin{cases} y_0 = k \cdot \alpha \text{ et } (c_1, c_2) = k \cdot \beta \\ y_1 = c_1 x_1 \pmod{p} \text{ et } y_2 = c_2 x_2 \pmod{p} \end{cases}$$

(2.16)

Pour déchiffrer donc le message chiffré de la forme $y = (y_0, y_1, y_2)$, on effectue l'opération :

$$\phi_K = (y_1 c_1^{-1} \pmod{p}, y_2 c_2^{-1} \pmod{p}) \text{ en sachant que } a.y_0 = (c_1, c_2) \quad (2.17)$$

Avec des sous-groupes cycliques de E à 2^{160} éléments on a une très bonne sécurité si l'on prend quelques précautions pour éliminer des courbes elliptiques indésirables pour lesquelles le problème du logarithme discret est facile [41]

2.4 Conclusion

La cryptographie se distingue en deux grandes catégories, le chiffrement à clé secrète qui utilise la même clé pour le chiffrement et déchiffrement ; le chiffrement à clé publique qui se base l'impossibilité de déduire la clé secrète à partir de clé publique. La cryptographie à clé secrète peut être un chiffrement par bloc ou un chiffrement à flot utilisant la méthode EBC, CBC, CFB, OFB. La cryptographie publique à clé publique peut être : RSA, El Gmal, PGP, RABIN, Cryptographie à base de correcteur d'erreur ou Menezes-Vanstone.

CHAPITRE 3 ORDINATEUR QUANTIQUE ET PROBLEME CALCULATOIRE

Dans ce chapitre, nous allons montrer comment les ordinateurs quantiques travaillent, comment les construit-on et comment ils font pour pouvoir effectuer des calculs rapides. De ce fait nous allons évoquer deux grands algorithmes quantiques du nom de Schor et Grover. Une question qui se pose : Parmi les cryptosystèmes cités précédemment, lesquels peuvent survivre à des attaques quantiques ? Comment évaluer la robustesse des algorithmes ? [23] [42] [43]

3.1 La loi de Moore – Génération d'ordinateur futur

Certains calculs sont très difficiles à effectuer en temps ou en espace. La théorie de la complexité s'attache à connaître la difficulté (ou la complexité) d'une réponse par algorithme à un problème, dit algorithmique, posée de façon mathématique. Pour pouvoir la définir, il faut présenter ces trois concepts qui sont les problèmes algorithmiques, les réponses algorithmiques aux problèmes, la complexité des problèmes algorithmiques. Les cryptanalystes essaient toujours de casser le cryptosystème pour des bonnes raisons (améliorations des cryptographie) ou des fins mal-vaillants. Ces calculs complexes peuvent être déjoués en utilisant des nouvelles générations d'ordinateur [44] [45].

Actuellement, l'informatique a pris une place considérable, voire même essentiel, dans le monde de l'industrie et des sciences. Dans le but d'être toujours plus rapides et toujours plus performants, les différents utilisateurs demandent un matériel informatique adapté à leurs besoins. Ainsi, nous assistons à une incessante amélioration de la capacité des processeurs, des mémoires et des périphériques composant un ordinateur. Le nombre de transistors qu'il est possible de placer sur un chip croît exponentiellement dans le temps, un constat appelé "Loi de Moore". Dans ce contexte, l'idée de changer le système informatique de base actuel au lieu de l'améliorer constamment est bien entendu prise en compte. Deux solutions peuvent être apportées :

- L'utilisation des calculs parallèles : Dans ces contextes, le cracker essaie de rechercher la clé en faisant des recherches exhaustives (brute force) en utilisant plusieurs machines travaillant en parallèle. Il utilise ce qu'on appelle un réseau d'ordinateur ou BotNet (milliards d'ordinateur travaillant en parallèle) en utilisant de la technologie Clustering ou Grid

Computing. Notons que le RSA, 1024bits peut être cassé par Botnet en 15minutes dans la conférence de CCC 2014. [46] [47]

- Les ordinateurs quantiques : Historiquement, c'est en 1985 que D. Deutsch démontra théoriquement qu'il était possible de réaliser une machine de Turing à l'aide des concepts de mécanique quantique. Le premier pas de la recherche pour un ordinateur quantique était fait. [48] [49]

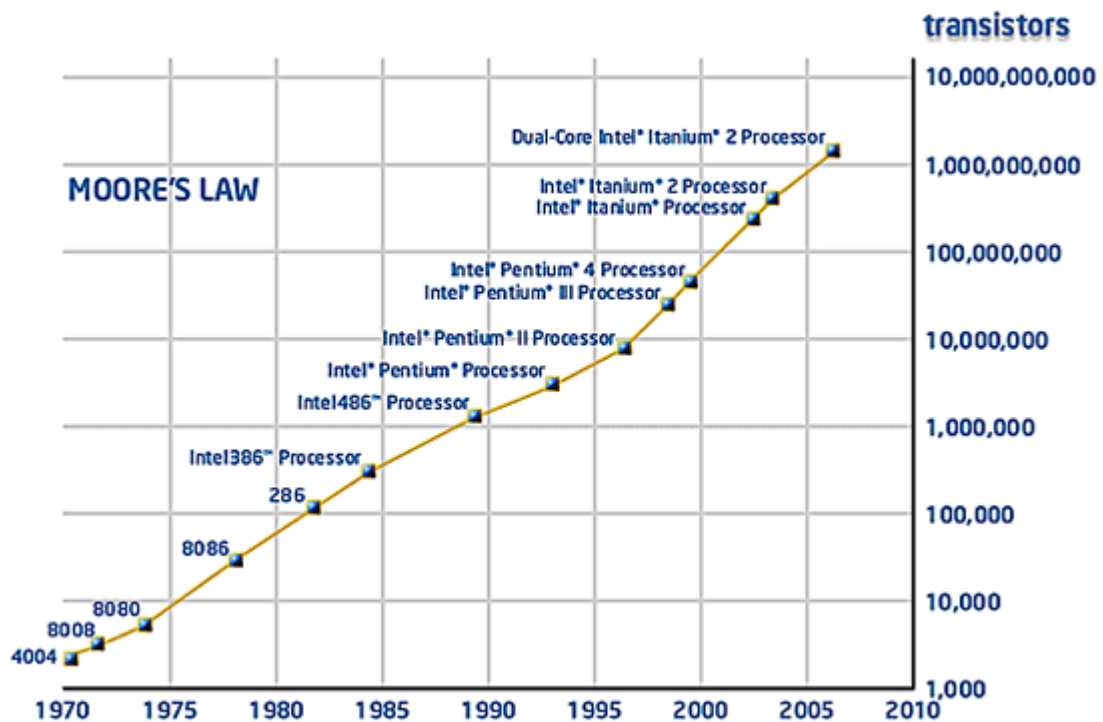


Figure 3.01 : *Illustration de Loi de Moore*

3.2 Les C-Bits et Q-Bits

Contrairement à ce que les gens pensent, les ordinateurs classiques résident en fait sur le phénomène quantique. Seulement, le vrai problème avec l'ordinateur classique est son registre. Les ordinateurs classiques utilisent ce qu'on appelle C-Bits (classiques bits). C'est une suite de zéros qui peut être interprétée par l'ordinateur : 10101010001. Paul Dirac a bien défini ce qu'on appelle C-nombre et Q-nombre. Pour faciliter la compréhension du nombre quantique, Dirac écrit les C-nombres sous cette forme : $|1\rangle|0\rangle|1\rangle$. Le nombre contient 3 bits. Cependant, il y a d'autres états qu'on n'a pu citer ici : $|0\rangle|0\rangle|0\rangle$; $|0\rangle|0\rangle|1\rangle$; $|0\rangle|1\rangle|0\rangle$; $|0\rangle|1\rangle|1\rangle$; $|1\rangle|0\rangle|0\rangle$; $|1\rangle|1\rangle|0\rangle$ et : $|1\rangle|1\rangle|1\rangle$

Les C-bits sont séparés de 3 registres. Mais on peut imaginer un registre avec une représentation en une seule boîte plus grande: $|000\rangle$; $|001\rangle$; $|010\rangle$; $|011\rangle$; $|100\rangle$; $|101\rangle$; $|110\rangle$ et : $|111\rangle$

Un tel registre est appelé Q-Bit. Mais au fur du temps la notation de Dirac fut abandonnée et se transforme aujourd'hui en Qu-bit ou qubit. Ce registre peut donc contenir le vecteur de tous les états possibles en utilisant le phénomène de superposition et le phénomène d'étrangement. Le bit quantique, ou qubit, est une abstraction d'un système quantique dont une observable possède deux états propres, notés $|0\rangle$ et $|1\rangle$, formant une base orthonormée $\mathcal{B}^+ = \{|0\rangle, |1\rangle\}$

Les propriétés d'un qubit découlent des postulats de la mécanique quantique. Par conséquent, l'état d'un qubit correspond à un vecteur d'état dans un espace de Hilbert complexe à deux dimensions :

$$|\omega\rangle = a|0\rangle + b|1\rangle = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (3.01)$$

Où les coefficients complexes a et b respectent la condition de normalisation : $a^2 + b^2 = 1$

Un qubit, contrairement à un bit, peut donc se trouver en superposition de ses états propres. Il constitue une unité d'information quantique et est une généralisation de l'information binaire classique. En posant $a = \cos\left(\frac{\theta}{2}\right)$ et $b = e^{i\rho} \sin\left(\frac{\theta}{2}\right)$; l'état $|\omega\rangle$ peut être représenté sur la sphère de Bloch à l'aide d'un vecteur unitaire faisant un angle θ avec l'axe z et un angle azimutal ρ avec l'axe x . On représente donc la sphère de Bloch [46] [50] par :

$$|\omega\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\rho} \sin\left(\frac{\theta}{2}\right)|1\rangle \quad (3.02)$$

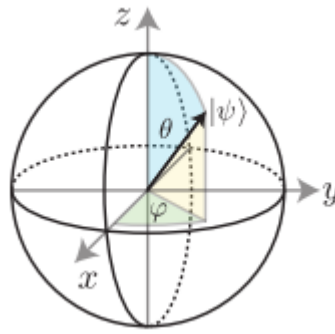


Figure 3.02 : La sphère de Bloch

L'information quantique est plus fragile que l'information classique. Ceci devient évident lorsqu'on applique le postulat de la mesure à un qubit. En effet, la mesure projective de l'état $|\omega\rangle$ dans la base \mathcal{B}^+ produira le résultat $|0\rangle$ avec la probabilité $|a|^2$ et $|1\rangle$ avec la probabilité $|b|^2$ et l'état suivant la mesure est projetée sur le résultat obtenu. A moins que l'état initial ne corresponde à un des états

de la base de mesure, l'état est perturbé par la mesure. Cette évolution probabiliste (et irréversible lorsque l'état initial était inconnu) est parfois appelée effondrement du paquet d'onde.

La superposition quantique a une étrange et remarquable conséquence lorsqu'on considère un système composé de plusieurs sous-systèmes : elle permet l'existence d'un état qui ne peut être décrit comme la somme des descriptions de chaque sous-système. Par exemple, l'état :

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle) \quad (3.03)$$

décrivant l'état conjoint de deux qubits intriqués, ne peut être factorisé en un produit tensoriel de deux états distincts : $|\phi^+\rangle \neq |\omega_1\rangle \otimes |\omega_2\rangle$ (impossibilité de décomposition en deux états de 1-qubit). On dit alors que les deux qubits sont dans un état intriqué, ou tout simplement qu'ils sont intriqués. L'intrication est une ressource quantique jouant un rôle essentiel dans plusieurs applications de la communication quantique et du calcul quantique.

Une autre propriété frappante de l'information quantique est le théorème de non-clonage. Les postulats de la mécanique quantique sont tels qu'une machine capable de cloner un qubit dans un état inconnu ne peut exister. Le contraste entre l'information classique et quantique est énorme car la possibilité de copier l'information classique, inconnue ou pas, est essentielle au bon fonctionnement de tous nos systèmes de traitement de l'information, du téléphone à l'ordinateur.

3.3 Espace d'Hilbert

Comme le registre du quantum augmente exponentiellement en fonction du nombre de Qu-bit, la meilleure représentation d'un **registre Qu-bit** choisie pour le nombre quantum est l'**espace d'Hilbert**. C'est un composé des nombres complexes donc des espaces linéaires. L'élément principal dans cet espace est le vecteur V . Selon la représentation de Dirac le vecteur sera noté de cette manière $|\rangle$. [51] [52]

Propriété de linéarité :

Imaginons deux vecteurs $|\alpha\rangle$ et $|\beta\rangle$ dans C^n . Ainsi, $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ et $\beta = (\beta_1, \beta_2, \dots, \beta_n)$. Notons $|\gamma\rangle = |\alpha\rangle + |\beta\rangle$. Comme l'espace d'Hilbert est linéaire alors $\gamma_i = \alpha_i + \beta_i$ avec $i \in [1; n]$. Comme les autres propriétés de vecteurs, l'addition est commutative et associative dans cet espace [53] [54] :

$$\begin{aligned}
|\alpha\rangle + |\beta\rangle &= |\beta\rangle + |\alpha\rangle \\
|\gamma\rangle + |\alpha\rangle + |\beta\rangle &= (|\gamma\rangle + |\alpha\rangle) + |\beta\rangle = |\gamma\rangle + (|\alpha\rangle + |\beta\rangle)
\end{aligned}
\tag{3.04}$$

En multipliant le vecteur par une constante c.à.d, on a les propriétés suivant :

$$\begin{aligned}
c(|\alpha\rangle + |\beta\rangle) &= c|\alpha\rangle + c|\beta\rangle \\
(c + d)|\alpha\rangle &= c|\alpha\rangle + d|\alpha\rangle \\
cd|\alpha\rangle &= c(d|\alpha\rangle)
\end{aligned}
\tag{3.05}$$

Propriété des vecteurs linéairement indépendant :

Les vecteur $|\alpha_1\rangle, |\alpha_2\rangle, \dots, |\alpha_m\rangle$ sont linéairement indépendant si il existe des nombres complexes c_1, c_2, \dots, c_m tel que :

$$c_1|\alpha_1\rangle + c_2|\alpha_2\rangle + \dots + c_m|\alpha_m\rangle = 0 \Rightarrow c_1 = c_2 = \dots = c_m = 0 \tag{3.06}$$

Propriété sur les produits scalaires :

On note $\langle\alpha|\beta\rangle$ le produit scalaire entre deux vecteurs $|\alpha\rangle$ et $|\beta\rangle$. Si $\langle\alpha|\beta\rangle = 0$ alors $|\alpha\rangle$ et $|\beta\rangle$ son orthogonaux. Soient c et d des nombres complexes [55][56]:

- Conjugué : $\langle\alpha|\beta\rangle = \langle\beta|\alpha\rangle^*$
- Linéarité : $\langle\alpha|c\beta + d\gamma\rangle = c\langle\alpha|\beta\rangle + d\langle\alpha|\gamma\rangle$
- Positivité : $\langle\alpha|\alpha\rangle \geq 0$
- Calcul : Si $|\alpha\rangle$ et $|\beta\rangle$ dans C^n . Ainsi, $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ et $\beta = (\beta_1, \beta_2, \dots, \beta_n)$ alors

$$\langle\alpha|\beta\rangle = \sum_{i=1}^n \alpha_i^* \beta_i \tag{3.07}$$

Notons que l'application directe de produit scalaire en ordinateur quantique et la théorie d'observabilité.

- Norme :

Si $|\alpha\rangle$ dans C^n . Ainsi, $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$, alors $||\alpha|| = \sqrt{\langle\alpha|\alpha\rangle}$ et encore

$$||\alpha\rangle|| = \sqrt{\sum_{i=1}^n |\alpha_i|^2} \quad (3.08)$$

Si la norme est égale à un, alors on a un vecteur unitaire.

- Complémentarité :

Prenons $a_i = \langle \alpha_i | \alpha \rangle$ avec $i \in [1; n]$, alors

$$\sum_{i=1}^n |\alpha_i\rangle \langle \alpha_i| = I \quad (3.09)$$

- Produit tensoriel :

C'est l'un des éléments très important en ordinateur quantique, car c'est grâce à l'opérateur produit tensoriel que l'on représente par \otimes qu'on peut superposer deux Qu-bits de registres différents. Deux registres Qu-bits donc signifie qu'on va travailler dans deux espaces d'Hilbert \mathcal{H}_1 et \mathcal{H}_2 . On va donc superposer ces deux registres Qu-bits en un seul plus grand [57] [58] :

$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. On peut citer les 3 propriétés suivantes :

- Pour $|\alpha\rangle \in \mathcal{H}_1$ et $|\beta\rangle \in \mathcal{H}_2$ et $c \in \mathbb{C}$; $c(|\alpha\rangle \otimes |\beta\rangle) = (c|\alpha\rangle) \otimes |\beta\rangle = |\alpha\rangle \otimes (c|\beta\rangle)$
- Pour $|\alpha_1\rangle, |\alpha_2\rangle \in \mathcal{H}_1$ et $|\beta\rangle \in \mathcal{H}_2$; $(|\alpha_1\rangle + |\alpha_2\rangle) \otimes |\beta\rangle = |\alpha_1\rangle \otimes |\beta\rangle + |\alpha_2\rangle \otimes |\beta\rangle$
- Pour $|\alpha\rangle \in \mathcal{H}_1$ et $|\beta_1\rangle, |\beta_2\rangle \in \mathcal{H}_2$; $|\alpha\rangle \otimes (|\beta_1\rangle + |\beta_2\rangle) = |\alpha\rangle \otimes |\beta_1\rangle + |\alpha\rangle \otimes |\beta_2\rangle$

Le calcul de produit tensoriel se fait de cette manière :

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1m}B \\ A_{21}B & A_{22}B & \cdots & A_{2m}B \\ \vdots & \vdots & & \vdots \\ A_{m1}B & A_{m2}B & \cdots & A_{mm}B \end{bmatrix} \quad (3.10)$$

Prenons comme exemple : $|\alpha\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ et $|\beta\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ en sachant que

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ et } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ alors : } |\alpha\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ et } |\beta\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\text{Donc } |\alpha\rangle \otimes |\beta\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\ -\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |00\rangle + |01\rangle - |10\rangle - |11\rangle$$

D'autres propriétés sont applicables dans l'espace d'Hilbert comme : inégalité de Cauchy- Swartz, décomposition de Gram-Schmidt, et produit Hermitien.

3.4 Les circuits quantiques

Comme avec l'ordinateur classique, l'ordinateur quantique possède ses propres opérateurs logiques quantiques (certains appellation boîte noire). Les opérateurs quantiques sont représentés sous forme matricielle si l'entrée est k qubits alors la matrice sera $2^k \times 2^k$. La représentation d'un qubit est donc : $v_0|0\rangle + v_1|1\rangle \rightarrow \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$ [55] [56]

en 2 qubits : $v_{00}|00\rangle + v_{01}|01\rangle + v_{10}|10\rangle + v_{11}|11\rangle \rightarrow \begin{pmatrix} v_{00} \\ v_{01} \\ v_{10} \\ v_{11} \end{pmatrix}$

Soit F la matrice de transfert, E la matrice d'entrée et S la matrice d'entrée. Donc, $S = F.E$

3.4.1 Hadamard Gate

C'est donc un opérateur logique qui a comme matrice de transfert la matrice d'Hadamard. Rappelons la formule de matrice Hadamard qui est une matrice récurrente de la taille $2^m \times 2^m$ [57]

$$\text{définie par [59] : } \begin{cases} H_m = \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix} \\ \text{avec } H_0 = 1 \end{cases} \text{ ou encore } H_m = H_1 \otimes H_{m-1} \quad (3.11)$$

Dans notre cas donc, on se contente d'un $m = 2$ vue que $k = 1$; $H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Donc $|0\rangle$ dévient donc $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ et $|1\rangle$ devient $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ L'interprétation de l'opérateur d'Hadamard signifie que les sorties ont la même probabilité. C'est aussi la combinaison de deux rotations : π suivant l'axe des abscisses et $\frac{\pi}{2}$ suivant l'ordonnée.

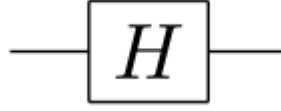


Figure 3.03 : *Circuit logique quantique d'Hadamard*

3.4.2 Pauli Gate

Le circuit de Pauli possède 3 variantes : X, Y, Z dont les matrices de transferts sont respectivement

$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ qui signifient respectivement une rotation de π radians suivant les axes des x, y, z dans la sphère de Bloch. En faisant le calcul on constate que le circuit Pauli-X permet de faire un circuit NOT. $|0\rangle$ dévient $|1\rangle$ en sortie et $|1\rangle$ dévient $|0\rangle$ en sortie. Le schéma de ce circuit est une case avec la lettre X, Y, Z [60].

3.4.3 Square root of Not Gate

La matrice de transfert de square root of not [53] :

$$\sqrt{NOT} = \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \text{ et } \sqrt{NOT} \cdot \sqrt{NOT} = NOT \quad (3.12)$$



Figure 3.04 : *Circuit quantique square root of Not*

3.4.4 Phase shift Gate

La matrice de transfert de Phase shift Gate étant : $R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$. C'est comme un tracage d'un cercle horizontal en écartant d'un angle ϕ . En général, on prend $\phi : \frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{2}$ et π (Pauli – Z). Le schéma de circuit est une case avec la lettre R8, R4, R2. La lettre R est remplacée par Z. [61] [54]

3.4.5 SWAP Gate

Le swap gate consiste à l'échange de 2qubits si en entrée on a : $|00\rangle, |01\rangle, |10\rangle, |11\rangle$,

en sortie on aura : $|00\rangle, |10\rangle, |01\rangle$ et $|11\rangle$. Sa matrice de transfert est représenté par la figure suivante. [62] [52]

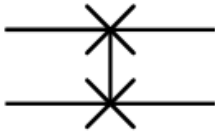
$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$


Figure 3.05 : Matrice de transfert et circuit quantique SWAP

3.4.6 Square root of swap Gate

Comme son nom indique donc $\sqrt{SWAP} \cdot \sqrt{SWAP} = SWAP$. [51]

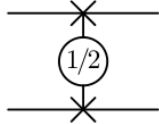
$$\sqrt{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(1+i) & \frac{1}{2}(1-i) & 0 \\ 0 & \frac{1}{2}(1-i) & \frac{1}{2}(1+i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$


Figure 3.06 : Matrice de transfert et circuit quantique Square root of Swap

3.4.7 Controlled Gate

La forme universelle de matrice de transfert de contrôle Gate est donc [52] [53]:

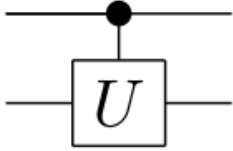
$$C(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}, \quad U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix},$$


Figure 3.07 : Matrice de transfert et circuit quantique de controlled-U Gate

On peut prendre des valeurs particulière de U, CNOT si $U = \text{diag}_2(2)$; Controlled-X , Y, Z si U est la matrice de Pauli-X, Y, Z.

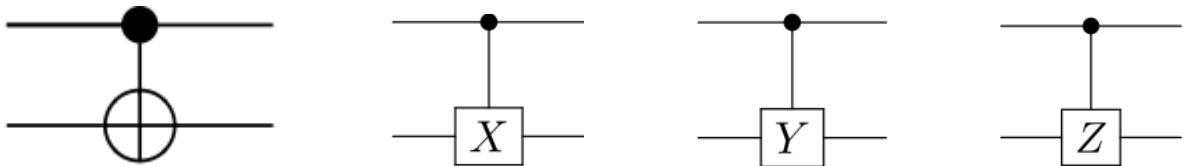


Figure 3.08 : *Circuit quantique CNOT, Controlled X, Y, Z*

3.4.8 Remarques

Selon le théorème de Solovay-Kitaev, il est possible de représenter tous les circuits quantiques avec seulement : H, R8, R4 et NOT. Seulement, le CNOT et \sqrt{SWAP} sont des circuits quantiques universelles pour minimiser le temps de calcul.

Des nombreux sites et programmes permettent les calculs quantiques, cependant, la registre augmente exponentiellement au fur et à mesure du nombre de qubits.

Les circuits de Toffoli et Fredkin travaillés en 3 qubits ne sont pas évoqués dans ce mémoire.

3.5 Algorithmes quantiques

3.5.1 QFT

L'expression de transformation de Fourier quantique est donnée par la formule suivante [52] [56] et $N = 2^n$:

$$|x_1 x_2 x_3 \dots x_n\rangle \rightarrow \frac{1}{\sqrt{N}} (|0\rangle + e^{2\pi i [0.x_n]} |1\rangle) \otimes (|0\rangle + e^{2\pi i [0.x_{n-1}x_n]} |1\rangle) \otimes \dots \otimes (|0\rangle + e^{2\pi i [0.x_1 x_2 \dots x_n]} |1\rangle) \quad (3.13)$$

On calcul $[0.x_1 x_2 \dots x_m] = \sum_{k=1}^m x_k \cdot 2^{-k}$. On peut créer directement avec les quantum Gate. Cependant, on réduit le calcul jusqu'à $1+2+3+\dots+n = n(n+1)/2$ c'est-à-dire $\mathcal{O}(n^2)$.

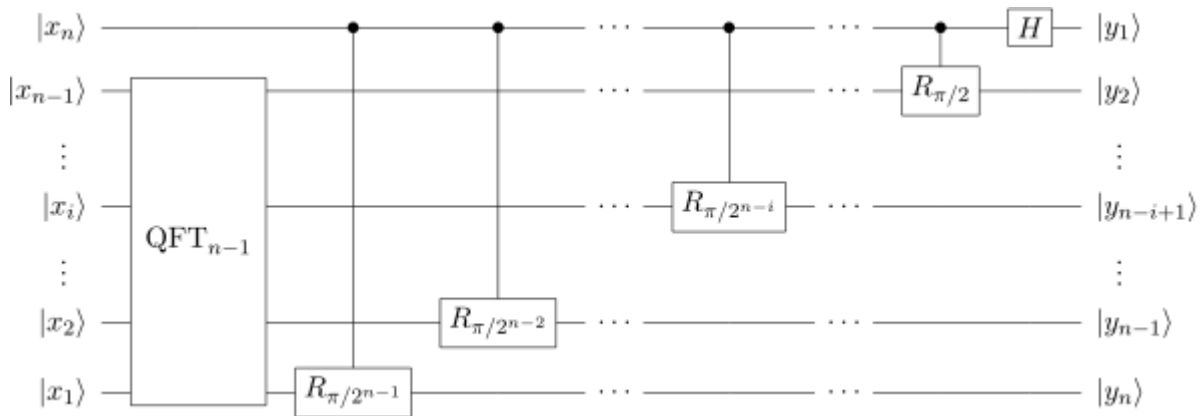


Figure 3.09 : *Recursiveité avec le circuit Quantum Fourier Transform*

En prenant $n = 3$ comme exemple, on aura :

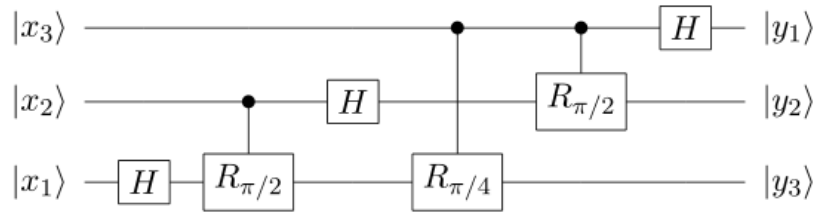


Figure 3.10 : *Quantum Fourier Transform de 3-qubits*

3.5.2 Algorithmes de recherche quantique de Grover

3.5.2.1 Le problème du mot de passe

Contrairement à ce que les gens pensent, les ordinateurs peuvent ne pas posséder la fréquence de travail rapide que les ordinateurs performants actuels. Sa rapidité réside du fait qu'ils peuvent résoudre certains problèmes combinatoires avec efficacité et rapide que les ordinateurs classiques. C'est la base même de l'algorithme de Grover. Imaginons que le chef de l'armée a oublié son mot de passe pour lancer un missile et qu'il demande à des informaticiens de le retrouver.



Figure 3.11 : *Le problème du mot de passe*

Et, là, l'informaticien va essayer tous les mots de passe possibles en prenant en compte la fonction de cryptage utilisé que l'on note f . Supposons que , le mot de passe est de n bits , donc pour pouvoir le casser, il faut $N = 2^n$ possibilités pour le faire. Nous sommes donc en face d' un problème de type NP.

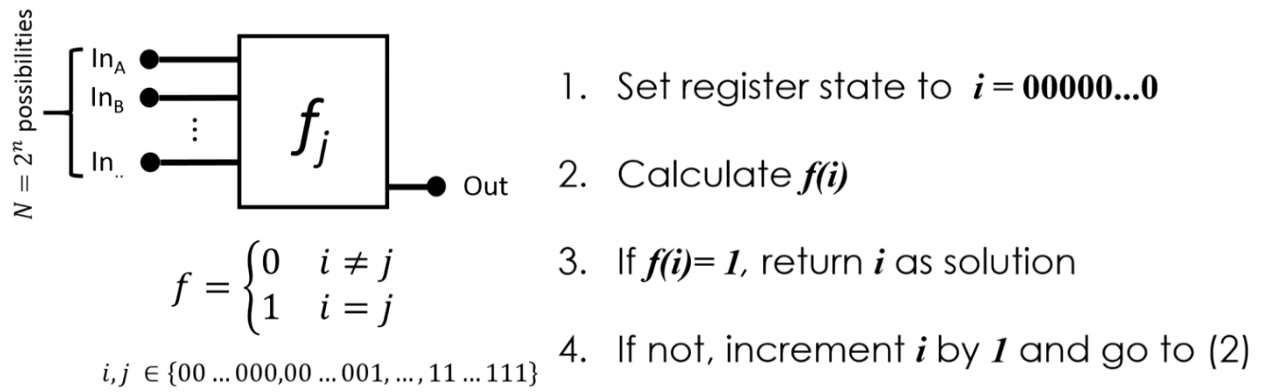


Figure 3.12 : Figure illustrative et algorithme des craquages par brute forcing

En tenant compte des algorithmes de cryptages actuels où $n = 1024$ bits voire même supérieure, alors l'ordinateur va prendre des milliers ou même des milliards d'année pour trouver le mot de passe.

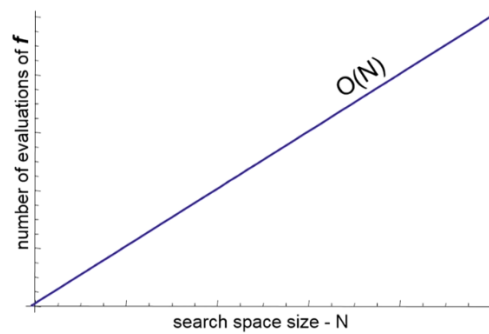


Figure 3.13 : Complexité temporelle de l'algorithme brute forcing

Cependant, on connaît que le registre quantiques de n bits contient déjà tous les 2^n états. Cependant même en ayant la possibilité de tester tous les mots de passes possible en une seule opération, on ne connaît pas lequel de ces registres contiennent la véritable réponse. Le fait de chercher les mots de passe parmi N possibilités revient au même problème. Dans le pire des cas, la recherche de mot de passe reste un problème de classe exponentielle dans le temps. Même donc en résolvant la complexité spatiale de l'algorithme, il faut considérer également la complexité temporelle.

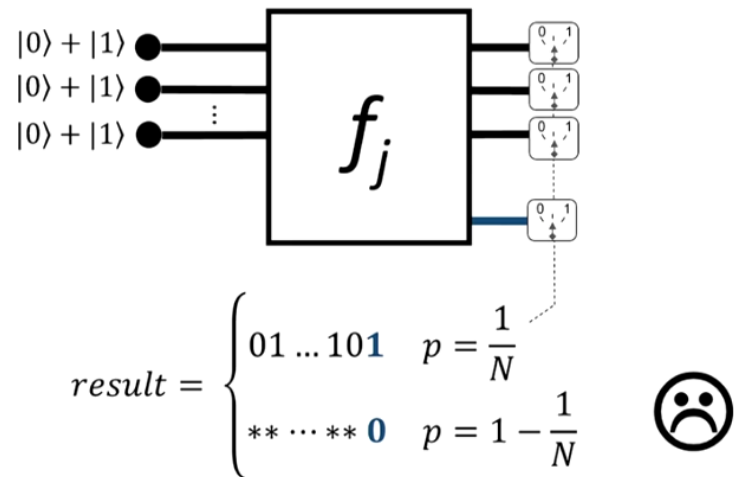


Figure 3.14 : *Problème lié à la complexité temporelle même via ordinateur quantique*

Grover inventa un algorithme, qui porta son nom, dont le but consiste à effectuer le même expérience pendant \sqrt{N} fois afin de déduire une conclusion.[57]

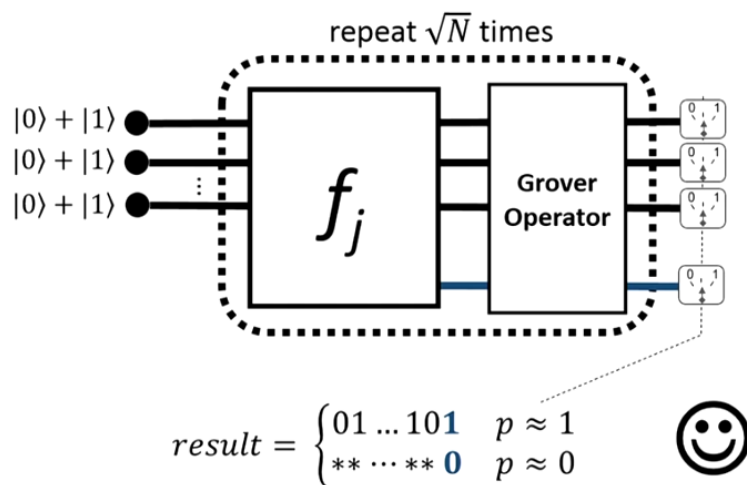


Figure 3.15 : *Illustration algorithme de Grover*

A partir de l'algorithme de Grover et l'algorithme, on a pu réduire la complexité temporelle en \sqrt{N} . Maintenant, on peut même améliorer l'algorithme de Grover en augmentant le qubit d'entrée.

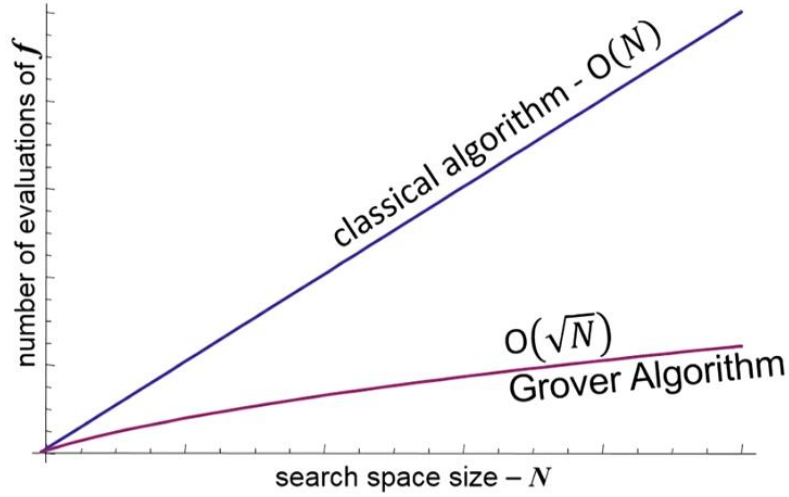


Figure 3.16 : *Tableau comparatif de complexité temporelle*

3.5.2.2 Algorithme de Grover

Le principe de Grover consiste à améliorer la recherche de résultats, applicable surtout à des problèmes combinatoires NP non quadratique. Il a été prouvé en 1999, par Christof Zalka, que c'est le meilleur algorithme de recherche non structuré de résultat quantique (non quadratique problème)

L'algorithme de Grover se divise en deux parties l'oracle ou le boite noire notés par \hat{O} et la matrice de diffusion $\hat{H}\hat{Z}\hat{H}$; en sortie, on a la formule suivante: $\hat{G} = (\hat{H}\hat{Z}\hat{H}) \hat{O}$

La boîte noire ou l'oracle va être représenté par une matrice de U_w défini par [57][62] :

$$U_w = \begin{cases} U_w|x\rangle = -|x\rangle \text{ quand } x = w \text{ c\`ad } f(x) = 1 \\ U_w|x\rangle = |x\rangle \text{ quand } x \neq w \text{ c\`ad } f(x) = 0 \\ U_w|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \end{cases} \quad (3.14)$$

Supposons maintenant $|s\rangle$ la superposition de tous les états qu'on va faire de recherche :

$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$ et la matrice H et la matrice d'Hadamard et la matrice Z sera représenté par $U_s = 2.|s\rangle\langle s| - I$. Voici l'étapes à suivre :

1. Initialisation de vecteur $|s\rangle$ en utilisant la formule $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$

2. A chaque itération : appliquer les deux matrices oracles et diffusions suivantes :
 - Matrice de l'oracle : $\hat{O} = U_w$
 - Matrice de diffusion : $\hat{H}\hat{Z}\hat{H} = H^{\otimes n}.U_s.H^{\otimes n}$
 - On a donc : $\hat{G} = (\hat{H}\hat{Z}\hat{H})\hat{O}$
3. La mesure du quantum bit final en prenant celui qui possède la probabilité la plus proche de 1 et la valeur finale.

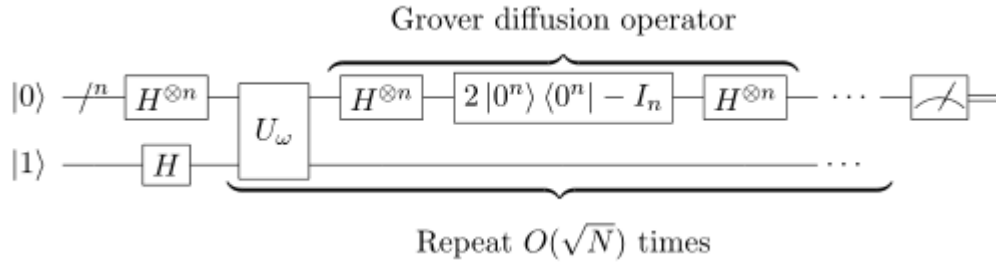


Figure 3.17 : Circuits quantiques de l'algorithmes de Grover

3.5.3 Algorithme de Schor – Factorisation quantique

3.5.3.1 Introduction au problème de factorisation des entiers

Non seulement l'algorithme de Grover menace les algorithmes de complexité exponentielle à problème non quadratique, mais aussi, l'algorithme de Shor s'attaque surtout au problème factorisation de nombre entier en produit de plusieurs nombres premiers connus sous le nom de FP (Factorization Problem). Si $\omega(n)$ le nombre de nombres premiers qui peut factoriser le nombre n , et p_i des nombres premiers qui peut factoriser le nombre n , la factorisation consiste donc [51][52]:

$$n = \prod_{i=1}^{\omega(n)} p_i \quad (3.15)$$

Nous pouvons constater la complexité en temps pour le problème de factorisation des entiers en deux produits de nombre premier ($r = q.s$ avec q et s des nombres premiers) avec un ordinateur classique qui est : $e^{1.9.\log[N]^{\frac{1}{3}}.\log(\log(\log N^{2/3}))}$. Cependant en utilisant l'algorithme de Shor, on peut réduire la complexité de l'algorithme en temps se réduit de $\mathcal{O}([\log(N)]^3)$ et en

espace $\mathcal{O}(\log(N))$. Actuellement, une réduction de résolution même une avec un complexité temporelle de $\mathcal{O}([\log(N)]^k)$ n'existe pas avec un ordinateur classique.

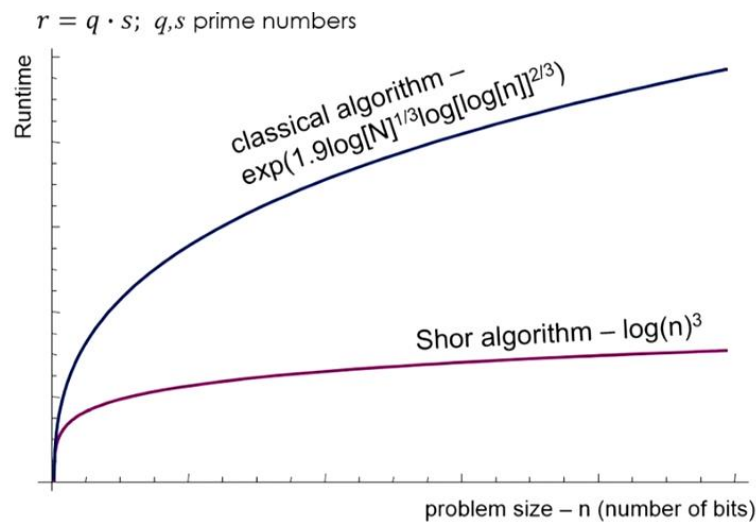


Figure 3.18 : Efficacité de l'algorithme de Shor

3.5.3.2 Mise en œuvre de l'algorithme de Shor

- En utilisant un algorithme classique
 - Prendre un nombre pseudo-aléatoire $a < N$
 - Calculer le PGCD (a, N) . Ceci peut être effectué par l'utilisation de l'algorithme d'Euclide.
 - Si $\text{PGCD}(a, N) \neq 1$, alors c'est un facteur non trivial de N , donc effectué.
 - Autrement, utiliser le sous-programme de recherche de période pour trouver r , la période de la fonction suivante : $f(x) = a^x \bmod N$ c.a.d. le plus petit entier r pour le quel $f(x + r) = f(x)$
 - Si r est impair, retourner à l'étape 1.
 - Si $a^{r/2} \equiv -1 \pmod{N}$, retourner à l'étape 1.
 - Les facteurs de N sont $\text{PGCD}(a^{r/2} \pm 1, N)$. Effectué
- En utilisant un algorithme : en s'inspirant de cet algorithme, Peter shor inventa l'algorithme quantique en utilisant le QFT pour résoudre le problème de factorisation.
 - Commencer avec des registres d'entrée et de sortie de chacun $\log_2(N)$ qubits, et les initialiser à : $N^{-1/2} \sum_{x=0}^{N-1} |x\rangle |0\rangle$
 - Construire $f(x)$ comme une fonction quantique et l'appliquer à l'état précédent, pour obtenir :

$$N^{-\frac{1}{2}} \sum_{x=0}^{N-1} |x\rangle f(x) \quad (3.16)$$

- Appliquer la transformée de Fourier quantique au registre d'entrée. La transformée de Fourier quantique sur N points est définie par :

$$N^{-1} \sum_x \sum_y e^{\frac{2\pi i[xy]}{N}} |y\rangle f(x) \quad (3.17)$$

- Effectuer une mesure. On obtient ainsi une certaine valeur y dans le registre d'entrée et $f(x_0)$ dans le registre de sortie. Comme f est périodique, la probabilité de mesurer un certain y est donnée par :

$$\left| N^{-1} \sum_{x:f(x)=f(x_0)} e^{\frac{2\pi i[xy]}{N}} \right|^2 = \left| N^{-1} \sum_b e^{\frac{2\pi i(x_0+rb)}{N}} \right|^2 \quad (3.18)$$

Le calcul montre que cette probabilité est plus haute quand $\frac{yr}{N}$ est proche d'un entier.

- Mettre $\frac{y}{N}$ sous forme irréductible, et extraire le dénominateur r' , qui est un candidat pour r.
- Vérifier si $f(x) = f(x + r')$. Si c'est le cas, c'est terminé.
- Autrement, obtenir plus de candidats pour r en utilisant des valeurs proches de y, ou multiples de r' . Si un autre candidat marche, c'est terminé.
- Sinon, retourner à l'étape 1 du sous-programme.

En 2001, par un groupe d'IBM, factorisa 15 en 3 et 5, en utilisant un calculateur quantique de 7 qubits. Pour simuler une factorisation de Shor, un langage de programmation du nom de qcl (quantum computation langage) fut inventé.

```

renaud@primestation:~/installations/qcl/qcl-0.6.4-x86_64-linux-gnu$ ./qcl
QCL Quantum Computation Language (64 qubits, seed 1416240575)
[0/64] 1 |0>
qcl> include "shor.qcl";
qcl> shor(77);
: chosen random x = 57
: measured 11621, approximation for 0.70929 is 61 / 86
: possible period is 86
: 57 ^ 43 + 1 mod 77 = 9 , 57 ^ 43 - 1 mod 77 = 7
: 77 = 7 * 11
[0/64] 1 |0>
qcl>

```

Figure 3.19 : Présentation de l'interface de qcl

Remarque :

D'autres algorithmes quantiques comme Deutsch-Joscha qui permettent de connaître si une fonction quantique est une constante ou non et l'algorithme quantique de Simon qui permet de connaître la périodicité d'une fonction ne sont pas évoqués dans mémoire vu que leur travail a peu de rapport avec la cryptanalyse en utilisant des ordinateurs quantiques.

3.6 Initiation à la création d'un ordinateur quantique**3.6.1 Concept de base**

Contrairement à ce que les gens pensent, tous les ordinateurs quantiques ne sont pas photoniques. On peut aussi créer un ordinateur quantique en utilisant la polarité et l'angle que fait l'électron positif et négatif. Différents systèmes physiques sont considérés pour la réalisation pratique. Les principaux sont les suivants [55] [63] [64] :

- Les ions piégés. Un ion piégé peut être vu comme une particule plongée dans un potentiel (Problème hydrogénoïde). Il est alors possible de considérer deux configurations différentes de l'ion, donc deux niveaux d'énergie distincts. A l'aide d'impulsions laser nous pouvons modifier l'état de chaque ion et l'intrication des états.
- Les boîtes quantiques (quantum dots). Les boîtes quantiques sont en fait une réalisation des puits de potentiel carré souvent considéré dans la théorie. Le Q-bit fait alors référence à deux niveaux d'énergies différents d'un électron placé dans un de ces puits.
- Les photons. Dans le cas des photons, les deux états du Q-bit sont deux états de polarisation et les opérations sont réalisées par d'astucieuses techniques d'interférométrie. L'intrication à trois photons a été démontrée. Comme le support de l'information n'est pas solide, l'interférométrie de photons est plutôt envisagée pour la communication et la cryptographie quantiques. (Utilisation des diamants NV-Center)

Il y a d'autres méthodes qui prennent de plus en plus de place importante : Superconducting Quantum bits, Résonance magnétique nucléaire, électron sur un superfluide d'Hélium, Condensateur de Bose-Einstein.

3.6.2 Evolution actuelle des ordinateurs quantiques

La première à s'intéresser à la fabrication des ordinateurs quantiques est le NSA. C'est une arme à double tranchante : pour vérifier si leur algorithme de cryptage n'est pas menacé et pour pouvoir espionner les grands publics. Le NSA a subventionné plus de 50 millions de dollars pour la recherche sur les ordinateurs quantiques.

La deuxième entreprise qui s'intéresse vraiment à l'ordinateur quantique est Google. L'algorithme de recherche rapide, et capable de résoudre la majorité des problèmes NP se fait par cette technologie. Des gens appellent ce phénomène « quantum speedup ». [53] [62]

D-Wave (D-Wave Systems) se présente comme première entreprise d'informatique quantique au monde, fondée en 1999 et basée en Colombie-Britannique (Canada). Ces machines utiliseraient une puce nommée Europa qui fonctionne uniquement en milieu cryogénique. Elle annonce en 2007 avoir construit le prototype d'un processeur de 28 qubits permettant de faire du recuit simulé quantique. En décembre 2009, un accord annoncé entre cette société et Google la remet sous les feux de l'actualité¹². En octobre 2010, elle présente dans le cadre des Google Techtalks le principe d'un classifieur quantique à grande échelle effectuant son apprentissage par une méthode de recuit simulé. Le 11 mai 2011, elle annonce son système D-Wave One comme le premier ordinateur quantique commercial. C'est un processeur de 128 qubits basé sur la méthode du recuit simulé quantique. En mai 2011, D-Wave vend à la société américaine de l'armement Lockheed Martin, pour 10 millions de dollars, un calculateur annoncé de 128 qubits, sur la nature quantique duquel planent cependant des doutes¹⁴. Cependant, en 2013, l'École polytechnique fédérale de Zurich et la société Google, ne trouvèrent « aucun indice de comportement quantique » surtout concernant le quantum speedup. Mais en décembre 2015, le D-Wave 2 de 1000 qubits sont des véritables ordinateurs quantiques à ce que Google affirme même si la plupart des chercheurs doute encore. En 2016, elle communique sur sa prochaine génération de processeurs contenant 2000 qubits.

3.6.3 Loi Rose

Fin 2012, Steve Jurvetson, l'un des investisseurs de D-Wave, énonce en référence aux lois de Moore une hypothèse concernant le développement des ordinateurs quantiques qu'il nomme « Loi de Rose », sous la forme : « La puissance de l'ordinateur quantique (le nombre de qubits disponibles) double

tous les ans. À la différence près, que contrairement à la loi de Moore, le doublement des qubits entraîne également une multiplication du pouvoir computationnel des machines. »

Le nom vient du créateur de la société : Mckeal Rose. Contrairement à la loi de Moore, qui est empiriquement vérifiée pendant des décennies, la loi de Rose est largement spéculative étant donné l'aspect encore embryonnaire et spéculatif du développement des ordinateurs quantiques, qui en 2014 ne sont pas réellement utilisés et qui existent à l'état de prototype de seulement quelques qubits, ce qui les rend inutiles en pratique.



Figure 3.20 : *Photo d'un ordinateur quantique D-wave*

3.7 Les problèmes calculatoires touchés par les ordinateurs quantiques

Merkle-Hellman est un cryptosystème asymétrique [11] [65] [66] [67]. Cependant, contrairement à RSA, il est à sens unique, c'est-à-dire que la clé publique est utilisée uniquement pour le chiffrement, et la clé privée uniquement pour le déchiffrement. Il ne peut donc pas être utilisé pour un protocole d'authentification. Il est basé sur le problème de la somme de sous-ensembles (ou subsetsum problem qui est un cas spécial du problème du sac à dos) : étant donnés n entiers et un entier p , existe-t-il un sous-ensemble de ces éléments dont la somme des valeurs est p ? Ce problème est NP-Complet. Cependant, l'algorithme de Hellman, Merkle et Diffie est sujet aux « portes dérobées » algorithmiques, ce qui implique qu'il est « cassé », c'est-à-dire cryptanalysé. Le problème du sac à

dos est un exemple classique de méprise en ce qui concerne les liens entre la NP-complétude et la cryptographie. Selon des mathématiciens notamment Gauss, Fermat, Euler, Lagrange et Legendre (Quadratic problem), Shafi Gold Wasser (Logarithm Discret Problem), et d'autres problèmes calculatoires énoncés dans le tableau 3.01 sont soit de la famille de Integer Factorization Problem soit on peut les réduire en Integer Factorization Problem.

Problem	Description
FACTORING	<i>Integer factorization problem</i> : given a positive integer n , find its prime factorization; that is, write $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.
RSAP	<i>RSA problem</i> (also known as <i>RSA inversion</i>): given a positive integer n that is a product of two distinct odd primes p and q , a positive integer e such that $\gcd(e, (p-1)(q-1)) = 1$, and an integer c , find an integer m such that $m^e \equiv c \pmod{n}$.
QRP	<i>Quadratic residuosity problem</i> : given an odd composite integer n and an integer a having Jacobi symbol $\left(\frac{a}{n}\right) = 1$, decide whether or not a is a quadratic residue modulo n .
SQROOT	<i>Square roots modulo n</i> : given a composite integer n and $a \in Q_n$ (the set of quadratic residues modulo n), find a square root of a modulo n ; that is, an integer x such that $x^2 \equiv a \pmod{n}$.
DLP	<i>Discrete logarithm problem</i> : given a prime p , a generator α of \mathbb{Z}_p^* , and an element $\beta \in \mathbb{Z}_p^*$, find the integer x , $0 \leq x \leq p-2$, such that $\alpha^x \equiv \beta \pmod{p}$.
GDLP	<i>Generalized discrete logarithm problem</i> : given a finite cyclic group G of order n , a generator α of G , and an element $\beta \in G$, find the integer x , $0 \leq x \leq n-1$, such that $\alpha^x = \beta$.
DHP	<i>Diffie-Hellman problem</i> : given a prime p , a generator α of \mathbb{Z}_p^* , and elements $\alpha^a \pmod{p}$ and $\alpha^b \pmod{p}$, find $\alpha^{ab} \pmod{p}$.
GDHP	<i>Generalized Diffie-Hellman problem</i> : given a finite cyclic group G , a generator α of G , and group elements α^a and α^b , find α^{ab} .
SUBSET-SUM	<i>Subset sum problem</i> : given a set of positive integers $\{a_1, a_2, \dots, a_n\}$ and a positive integer s , determine whether or not there is a subset of the a_j that sums to s .

Tableau 3.01: Problème combinatoire (calcul computation) non résistant à l'attaque quantique

De ce fait, certains algorithmes survivent à cette attaque notamment [2] [68] :

- Algorithme symétrique : AES et ses familles
- Algorithme asymétrique : cryptosystème basé sur le code d'erreur comme McEliece.

Notons que le RSA, DH, ECDH, El Gamal, Miller Rabin, Menezes-Vanstone, PGP ne sont pas résistant face à la cryptanalyse quantique. Et encore, il faut une certaine longueur de clé ou d'algorithme correcteur d'erreur utilisé pour le cryptosystème basé sur le code d'erreur.

3.8 Les différents calculs combinatoires post-quantiques

Les différents problèmes calculatoires post quantique sont multiples : Latticed Based cryptography, Multivariate cryptography, SIDH, et code based et HashBased cryptographie. [32] [42] [69] [70]

3.8.1 *Latticed based cryptography*

3.8.1.1 Définition

Le « Latticed based problem » ou encore le problème de réseau engendré dans une famille de base possède 3 types de problèmes : SVP, CVP, et SIVP [2] [10] [72] [73]. Mais avant toute chose, définissons ce que l'on entend par latticed based ou réseau engendré par une famille de base $B = (b_1, b_2, \dots, b_n)$ noté \mathcal{L} :

$$\mathcal{L} = \mathcal{L}(B) = \{Bx = \sum_{i=1}^n b_i x_i \mid x_i \in \mathbb{Z}\} \quad (3.19)$$

Un réseau L admet plusieurs bases. Le passage entre deux bases se fait à l'aide d'une matrice carrée à coefficients dans \mathbb{Z} , de déterminant ± 1 . Parmi toutes les bases possibles, certaines ont de meilleures propriétés que d'autres. La recherche d'une bonne base est un problème NP-complet. En 1982, Lenstra, Lenstra-Lovász ont proposé l'algorithme LLL qui détermine une base avec de très bonnes propriétés. L'algorithme LLL est très efficace puisque sa complexité est polynomiale. On remarque en particulier que l'algorithme LLL produit des bases avec des vecteurs assez courts, ce qui peut apporter une réponse partiellement satisfaisante à des problèmes (NP-durs).

3.8.1.2 Propriété

- $\det(\mathcal{L}) = \det(B)$
- Le dual de \mathcal{L} noté $\mathcal{L}^* = \mathcal{L}(B^{-T})$ et $\det(\mathcal{L}^*) = \frac{1}{\det(\mathcal{L})}$

3.8.1.3 Problème dans un latticed based

Soit $B \in GL_n(\mathbb{Z})$ une base et γ le paramètre d'approximation. Les problèmes combinatoires le plus connus sont :

- SVP (Shortest Vector Problem) : Etant donné une base B d'un réseau \mathcal{L} , trouver un vecteur non nul $v \in \mathcal{L}(B) \setminus \{0\}$ de \mathcal{L} le plus court possible pour la norme euclidienne.

$$\|v\| < \gamma \min(w)_{w \in \mathcal{L}(B) \setminus \{0\}}$$

- CVP (Closest Vector Problem) : Etant donné une base B d'un réseau \mathcal{L} et un vecteur $t \in \mathbb{R}^n$, trouver un vecteur $v \in \mathcal{L}$ le plus proche possible de t pour la norme euclidienne.

$$\|v - t\| < \gamma \min(\|w - t\|)_{w \in \mathcal{L}(B) \setminus \{0\}}$$

- SIVP (Shortest Independent Vector Problem) : Etant donné une base B d'un réseau \mathcal{L} , trouver un vecteur $U \in GL_n(\mathbb{Z})$ telque :

$$\|BU\| < \gamma \min(BV)_{V \in GL_n(\mathbb{Z})}$$

3.8.1.4 Exemple de problème

Les algorithmes post-quantique pour le moment qui ne peuvent pas être réduits en problème calculatoire de factorisation en entier premier et qui ne peuvent être résolus en temps linéaire en utilisant l'algorithme de Grover sont : NTRU, BLISS, RING-LWE, LWE ...

3.8.2 Multivariate cryptographie

3.8.2.1 Définition

Pour comprendre une cryptographie à schéma multivariée, le plus simple est de se référer d'abord à la définition d'une cryptographie à schéma univariée. C'est une cryptographie dont la difficulté de résoudre le problème se base sur une et une seule équation avec une seule variable. Le RSA par exemple, possède une fonction de chiffrement de la forme : $c = B^e \bmod(n)$. Adi Shamir, l'un des fondateurs de RSA incite l'utilisation des cryptographies à schéma multivariées. La difficulté de résolution du problème dépend des plusieurs équations avec plusieurs variables comme [73] [74] :

$$\begin{cases} y_1 = x_1x_2 + x_2x_2 + x_2x_3 + x_3x_3 \\ y_2 = x_1x_1 + x_2x_3 + x_3x_3 \\ y_3 = x_1x_2 + x_1x_3 + x_2x_3 \end{cases}$$

Le plus célèbre de schéma inventé dans ce domaine est celui de Matsumoto-Imai (MI) en 1988. Il y a aussi UOV, STS (Stepwise Triangle System), HFE (Hidden Field Equation). Kipnis-Patarin-Goubin invente une méthode pour réduire le crackage de la cryptographie à schéma multivarié. Cependant, son temps reste exponentiel que ce soit quantique ou non.

Soit f_1, f_2, \dots, f_m et un vecteur $x = (x_1, x_2, \dots, x_n)$ dans un automate fini k .

$$\begin{cases} f_1(x_1, \dots, x_n) = \sum_{1 \leq i \leq j \leq n} a_{1,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{1,i} x_i + c_1 \\ f_2(x_1, \dots, x_n) = \sum_{1 \leq i \leq j \leq n} a_{2,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{2,i} x_i + c_2 \\ \vdots \\ f_m(x_1, \dots, x_n) = \sum_{1 \leq i \leq j \leq n} a_{m,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{m,i} x_i + c_m, \end{cases} \quad (3.20)$$

Avec $a_{l,i,j}; b_{l,i}; c_l \in k$; MQ-Problème consiste à trouver un vecteur $s = (s_1, s_2, \dots, s_n)$ tel que $f_i(s_1, s_2, \dots, s_n) = 0 \forall i \in [1; m]$

3.8.2.2 UOV Problem

UOV (Unbalanced Oil and Vinegar) problème ou problème de mélange d'huile et du vinaigre est le plus célèbre problème lié à la cryptographie à schéma multivarié. Soit F un automate fini et $n, m \in \mathbb{N}$ et $m < n$ et

$\alpha'_i, \beta'_{i,j}, \gamma'_{i,j,k} \in F$; tel que $1 \leq i \leq m$ et $1 \leq j \leq k \leq n$. Le problème de l'huile et vinaigre consiste à mélanger correctement les deux produits en répondant aux critères suivants :

$$p'_i(x'_1, x'_n, \dots, x'_n) = \sum_{j=1}^{n-m} \sum_{k=1}^n \gamma'_{i,j,k} x'_j x'_k + \sum_{j=1}^n \beta'_{i,j} x'_j + \alpha'_i \quad (3.21)$$

Pour avoir un bon mélange de l'huile et de vinaigre, il faut et il suffit qu'il existe un vecteur $s' = (s'_1, s'_n, \dots, s'_n)$ tel que $p'_i(s'_1, s'_n, \dots, s'_n) = 0$

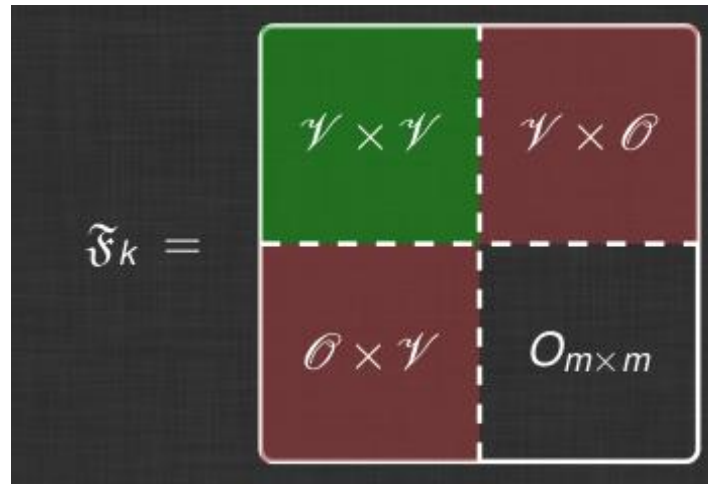


Figure 3.21 : *Schéma de Oil and Vinegar*

3.8.2.3 MQ-PKC

MQ-PKC ou Multivariate Quadratic Public Key Cryptosystem est le nom donné par le type de cryptosystème à schéma multivariable comme : MI, HFE, RAINBOW, ... Les MQ-PKC sont considérés comme les successeurs de RSA.

3.8.3 SIDH (*Supersingular Isogeny Diffie Hellman*)

3.8.3.1 Définition

- **Courbe elliptique super-singulière :** Une courbe elliptique sous forme cubique $x^3 = f(w, x, y)$ travaillant dans un corps de Gallois $GF(2^k)$ avec $p = 2^k$ est dite super singulière si et seulement si le coefficient $(wxy)^{p-1}$ est zéro quand on élève $f(w, x, y)$ à la puissance $p-1$ [29] [75] [76].

Exemple : Prenons un exemple $E_\alpha : x^3 = wy^2 + w^2y + \alpha wxy$

Pour que cette courbe soit super-singulière dans $GF(2)$ alors il faut que $\alpha = 0$. En prenant $w = 1$; on a notre courbe super singulière : $x^3 = y^2 + y$

- **Map et morphisme dans une courbe elliptique :**

Soient $\varepsilon_1 : y_1^2 = F_{\varepsilon_1}(x_1)$ et $\varepsilon_2 : y_2^2 = F_{\varepsilon_2}(x_2)$ deux courbe elliptique dans F_q . [77]

Un morphisme $\emptyset : \varepsilon_1 \rightarrow \varepsilon_2$ est appelé map. On a ainsi :

$$\emptyset : (x_1, y_2) \rightarrow (x_1, y_2) = (\emptyset_x(x_1, y_2), \emptyset_y(x_1, y_2))$$

Avec : $\emptyset_y^2 = F_{\varepsilon_2}(\emptyset_x)$

- Homomorphisme : un morphisme respectant toute la loi dans le groupe
- Isomorphisme : homomorphisme inversible
- Endomorphisme : homomorphisme d'une courbe elliptique est lui-même
- Automorphisme : Endomorphisme inversible
- **Isogénies** : C'est un map rationnel (non-constant) avec $\emptyset : \varepsilon_1 \rightarrow \varepsilon_2$ est un homomorphisme [78][79]

Le SIDH (super Isogenie Diffie Hellman), c'est un protocole d'échange de secret auquel il est très difficile de l'isogénie \emptyset en sachant le point E et $\emptyset(E)$.

	DH	ECDH	SIDH
Eléments	Entier g modulo nombre premier	Point P dans un groupe de point de courbe elliptique	Isogénie dans un courbe E
Secrets	Exponentiels	scalaires	Isogénie
Calcul	$g, x \rightarrow g^x$	$k, P \rightarrow [k]P$	$\emptyset, E \rightarrow \emptyset(E)$
Problème combinatoire	Trouver x en sachant g^x et g	Trouver k en sachant $[k]P$ et P	Trouver \emptyset en sachant $\emptyset(E)$ et E

Tableau 3.02: Les générations de Diffie Hellman

SIDH est inventé par Jao et Dee Feo et Plut en 2011. Ils ont recommandé l'utilisation des courbes super-elliptiques dans un corps de 2^p avec p un nombre premier de 760-bits. A l'heure actuelle pour une attaque classique, la complexité temporelle est $\odot (p^{\frac{1}{4}})$ et attaque quantique $\odot (p^{\frac{1}{6}})$

3.8.4 *Code based et Hash based cryptography*

La meilleure sélection de signature en cryptographie, est la signature à base de hachage utilisée par Merkle et Lamport en 1970. OTS (One Time Signature) est sur le marché depuis le 03 novembre 2016 qui regroupe toutes les signatures résistantes à l'attaque quantique développé par David McGrew et Michael Curcio depuis 02 juillet 2014. Pour la version non payante de Hashbased cryptography, Il est possible d'utiliser SHINCS, une commande pour signer des documents en Linux.

Les « code based cryptography » ou cryptographie à base de correcteur d'erreur, utilisent le principe de confusion de Shanon [16] (en ajoutant des erreurs corrigibles aux messages) et principe de diffusion (en utilisant la permutation et la matrice singulière G). Ces cryptographies consistent à Mc-Eliece cryptosystème et Niederreiter cryptosystème.

3.9 Conclusion

Pour éviter la loi de Moore, les créateurs de circuits électroniques ont deux solutions : les clusters ou le calcul en parallèle et l'ordinateur quantique. L'avantage de l'ordinateur quantique réside sur son registre qui peut contenir déjà toutes les possibilités d'états. Pour mieux comprendre comment fonctionne le Qubit, il faut travailler avec l'espace d'Hilbert et prendre en compte tous les propriétés fondamentales. Pour pouvoir effectuer des opérations, des circuits quantiques ont été créés. Les plus importants d'entre eux sont : Hadamard Gate, Pauli Gate, Squareroot of Not Gate, Phase shift Gate, SWAP Gate, Square root of swap Gate et Controlled Gate. Mais même avec un registre possédant tous les états, il faut avoir un algorithme quantique pour réduire le temps de calcul pour certains problèmes NP : QFT, Grover et Shor. La plupart des problèmes calculatoires doivent faire face à ce problème généralement le problème de factorisation. Pour ce faire, il faut inventer des nouveaux problèmes calculatoires de référence : Latticed based cryptography, Mutlivariate cryptography , SIDH, et code based cryptography et hash based cryptography.

CHAPITRE 4 IMPLEMENTATION D'UN ALGORITHME POST-QUANTIQUE

La plupart des algorithmes actuels est donc menacée par les cryptanalyses quantiques. Dans ce chapitre, nous allons voir les différentes possibilités qu'on peut choisir si nous voulons créer un algorithme post-quantique et nous allons faire une implémentation. De ce fait, il faut bien tenir compte qu'un algorithme post-quantique n'utilise pas des calculs quantiques, ce sont des algorithmes classiques travaillant dans des ordinateurs classiques et pouvant résister à la cryptanalyse quantique [5] [80] [81]. Chaque année également, le NIST organise une sélection d'algorithme de chiffrement pour se préparer de la cryptographie utilisée en 2020 pour l'avènement de l'Iot 5G, on les appelle H2020 (Horizon 2020). Ces algorithmes travaillent dans des protocoles à faible poids (light weight security). Certains résistent à des attaques de type implémentation (SPA, DPA, SCPA) et d'autres résistent à des attaques de types quantiques comme le cryptosystème Mc-Eliece.

4.1 Etude approfondie de Mc-Eliece Cryptosystème

4.1.1 Généralités sur les codes correcteurs d'erreurs

L'étude sur la théorie de code et code correcteur furent approfondis par de mathématiciens tel que Shannon, Golay et Hamming en 1940 [19] [82]. Leur but était de corriger les erreurs en ajoutant des bits de redondances lors de transmissions de données télégrammes à cause des bruits électriques de la ligne. Pour ce faire, il y a 2 méthodes : le codage en bloc, codage convolutif.

La méthode utilisée dans ce chapitre est le codage en bloc, ce nom est dû sur le fait qu'on va partager le mot à coder en une longueur de bit connue. Certains disent aussi que ce sont des (n,k) -code car un bloc de k bits d'information est codé en n bits appelés mots codes. Pour vérifier si un mot code est valide ou non, il faut multiplier le mot à coder par une matrice Générateur G et puis multiplier par une seconde matrice du nom de matrice de parité ou matrice de vérification H . Le syndrome à la réception, est une matrice qui permet de détecter s'il y a erreur ou non et de savoir la position de cette erreur et puis la corriger. Le nombre t est désigné par la capacité de corriger une erreur [85] [83].

- Soit F_q un automate fini de q élément et F_q^n un espace de vecteur de n -uplet dans cet automate. Un (n,k) -code linéaire nommé C est un vecteur sous espace de k -dimension. Ses vecteurs sont de la formule $(a_1, a_2, \dots, a_{q^k}) \in C$
- La distance de Hamming $d(x, y)$ entre deux vecteurs $x, y \in F_q^n$ est défini par le nombre de position qui est différent à chaque position des deux vecteurs. Càd $\forall x_i, y_i \in F_q^n$, donc, $d(x, y) = \{a \in \mathbb{R} \mid a = \text{card}\{x_i \neq y_i; 1 \leq i \leq n\}\}$. Et on note également, $w_t(x) = w_H(x) = d(x, 0)$ avec 0 désigne le vecteur nul.
- Une matrice est dite Génératrice $G \in F_q^{k \times n}$, de (n,k) -code linéaire si et seulement si :
 $C = \{x \cdot G \mid x \in F_q^k\}$. En général, il y a plusieurs matrices génératrices et on les donne de nom comme GRS, RS ou GoppaCode ... la matrice formée à partir de $(n-k)$ -ème colonne jusqu'à n -ème colonne forme ce qu'on appelle matrice de redondance. En général, la matrice G est de la forme $G = [I_k | Q]$ avec I_k matrice identité de dimension $k \times k$
- Pour code C dans (n,k) -code linéaire, il existe une matrice $H \in F_q^{(n-k) \times n}$ tel que $C = \{y \in F_q^n \mid H \cdot y^T = 0\}$. La matrice H est appelée matrice de parité pour le code C . Il y a plusieurs matrices de parité mais si :

$$G = [I_k | Q] \text{ avec } I_k \text{ matrice identité de dimension } k \times k, \text{ alors}$$

$$H = [-Q^T | I_{n-k}] \text{ avec } I_{n-k} \text{ matrice identité de dimension } (n-k) \times (n-k)$$

- Hiérarchies des classes de code : La hiérarchie de code permet de savoir quel code utilisé et quel code utilisé et leur rôle correspondant. On distingue le codage source et le codage canal respectivement pour la compression et pour la détection et correction d'erreur.

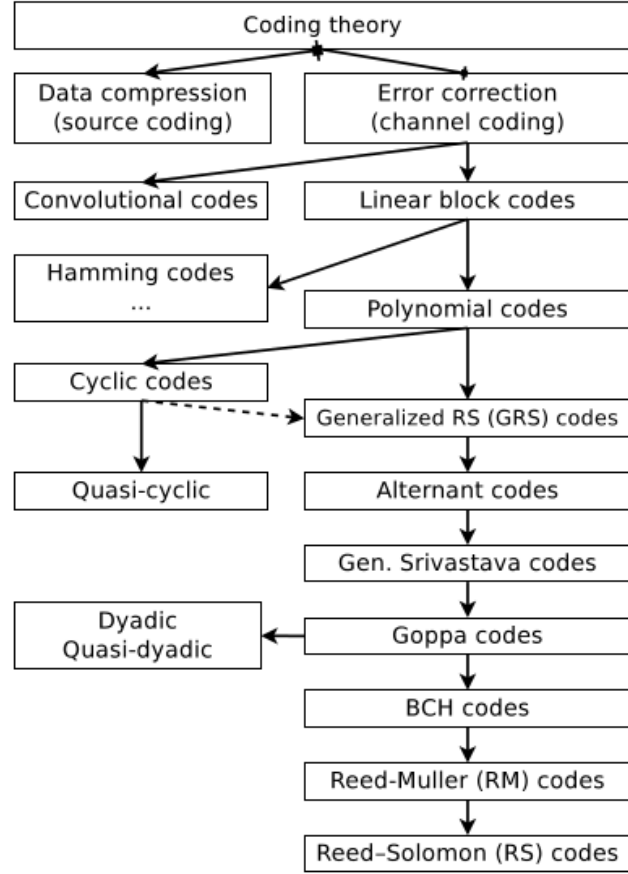


Figure 4.01 : *Hiérarchies des codes*

Les codes linéaires en bloc peuvent être cycliques ou généralisés [86]. Tous les codes correcteurs employés comme Alternant Codes, Generalized Srivastava Codes, Goppa Codes, BCH codes, Reed-Muller Codes et Reed-Solomon Codes sont des variants de GRS (Generalized Reed-Solomon codes). Ces codages sont basés sur la matrice de Vandermonde.

- Soit F_q un automate avec $q = p^m$ avec p premier et $m > 0$; La matrice de Vandermonde est définie par

$$\mathbf{V}_t(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} y_0 & \cdots & y_{n-1} \\ y_0 x_0 & \cdots & y_{n-1} x_{n-1} \\ \vdots & & \vdots \\ y_0 x_0^{t-1} & \cdots & y_{n-1} x_{n-1}^{t-1} \end{pmatrix}$$

(4.01)

Avec $(x = (x_0, x_1, \dots, x_{n-1}), y = (y_0, y_1, \dots, y_{n-1})) \in F_{p^m}^n \times F_{p^m}^n$

Les coefficients de GRS sont obtenus à partir des coefficients de la colonne de la matrice de Vandermonde.

- Soit $x = (x_0, x_1, \dots, x_{n-1}) \in (F_{p^m})^n$, à chaque x_i , on fait correspondre un y_i tel que $y = (y_0, y_1, \dots, y_{n-1}) \in (F_{p^m}^*)^n$; Le GRS de dimension t noté :
 $GRS_{n,k,t}(x, y) = \{(y_0 Q(x_0), y_1 Q(x_1), \dots, y_{n-1} Q(x_{n-1})) | Q \in F_{p^m}(Z) \text{ et } \deg(Q) \leq t-1\}$

4.1.2 Goppa Code

- On appelle Goppa Code Γ est défini par : $\Gamma_{n,k,p}(L, g) = GRS_{n,k,t}(L, g) \cap F_p^n$. Autrement dit,

Soit $q = p^m$, et $L = (\alpha_0, \alpha_2, \dots, \alpha_{n-1}) \in F_q^n$ et $g(z) \in F_q(z)$ un polynôme tel que $g(\alpha_i)_{1 \leq \alpha_i \leq n-1} \neq 0$ et $v = (g^{-1}(\alpha_0), g^{-1}(\alpha_1)), \dots, g^{-1}(\alpha_{n-1}) \in F_q^n$; Pour un Goppa code $\Gamma(L, g)$ et un code $c \in F_p^n$ alors $H \cdot c^T = 0$, avec H est la matrice de parité de Γ

- Un Goppa code binaire est un Goppa Code avec $p = 2$. Supposons maintenant que le polynôme g est de la forme

$$g(z) = \sum_{i=0}^t g_i \cdot z^i \in F_{2^m}(z) \quad (4.02)$$

Et prenons un support $L = (\alpha_1, \alpha_2, \dots, \alpha_{n-1})$, Pour un vecteur $\hat{c} = (c_0, c_1, \dots, c_{n-1}) \in F_{2^m}^n$ alors le syndrome de \hat{c} est défini par :

$$S_{\hat{c}} = - \sum_{i=0}^{n-1} \frac{\hat{c}_i}{g(\alpha_i)} \frac{g(z) - g(\alpha_i)}{z - \alpha_i} \text{ mod}(g(z)) \quad (4.03)$$

On peut donc améliorer la définition de Goppa code en utilisant le syndrome par :

$$\Gamma_{n,k,2}(L, g(z)) = \{c \in F_{2^m} | S_c(z) = \sum_{i=0}^{n-1} \frac{c_i}{z - \alpha_i} \equiv 0 \text{ mod}(g(z))\} \quad (4.04)$$

Si le polynôme g est irréductible alors on dit que le Goppa code est irréductible [87]

Prenons un exemple : Soit $g(x) = (001)x^2 + (100)x + 001$ travaillant dans un groupe de Galois $GF(2^8)$. Et prenons un support $L = (100,001,111,011,010,000,101,110)$ et la matrice de Vandermonde V par $V_{j,i} = L_i^{j-1}$ et la matrice diagonale D par $D_{i,i} = g(L_i)^{-1}$

$$V = \begin{pmatrix} 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 \\ 100 & 001 & 111 & 011 & 010 & 000 & 101 & 110 \end{pmatrix}$$

$$D = \begin{pmatrix} 001 & & & & & & & \\ & 111 & & & & & & \\ & & 110 & & & & & \\ & & & 110 & & & & \\ & & & & 011 & & & \\ & & & & & 001 & & \\ & & & & & & 111 & \\ & & & & & & & 011 \end{pmatrix}$$

On calcul la matrice de parité par :

$$H = VD = \begin{pmatrix} 001 & 111 & 110 & 110 & 011 & 001 & 111 & 011 \\ 100 & 111 & 100 & 001 & 110 & 000 & 110 & 001 \end{pmatrix} \quad H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4.1.3 Première version de Mc-Eliece par Goppa Code

Robert Mc-Eliece montre sa première version de cryptage à base de correcteur d'erreur que l'on nomme aujourd'hui MECS (McEliece Cryptosystem Security). L'idée est d'ajouter au message des erreurs et à la fois les chiffrer pour que des personnes malveillantes ne le connaisse pas. Au départ McEliece utilise des Goppa code pour chiffrer les messages. Des algorithmes de décodage utilisés pour vérifier les erreurs lors de la lecture CD notamment nommé l'algorithme de Patterson ou de Berklamp Massey ont des rôles importants. L'équation de base utilisée pour la cryptographie à base de correcteur d'erreur est [88]: $c = m.G + e$

$$\text{plaintext } \mathbf{m} \in \mathbb{F}_q^k \begin{pmatrix} 0, \dots, 1 \end{pmatrix} \begin{pmatrix} \text{public key } \mathbf{G}_{pub} \in \mathbb{F}_q^{k \times n} \\ \mathbf{G}_{pub} \end{pmatrix} + \begin{pmatrix} \text{error } \mathbf{e} \in \mathbb{F}_q^n \\ 1, 0, \dots, 0, 1 \end{pmatrix} = \begin{pmatrix} \text{ciphertext } \mathbf{c} \in \mathbb{F}_q^n \\ 1, 0, \dots, 1, 1 \end{pmatrix}$$

Figure 4.02 : Représentation d'un chiffrement à base de correcteur d'erreurs

Soit G une matrice génératrice $k \times n$ du code C et un bon algorithme de décodage D_F avec une possibilité de décoder un code jusqu'à t erreurs. Posons S une matrice non singulière aléatoire $k \times k$ et P une matrice de permutation aléatoire $n \times n$. MECS suit la manière suivante [89] :

Clé secrète : (D_F, S, P)

Clé publique : $\hat{G} = S.G.P$

Chiffrement : Soit m un message à chiffré et soit e un vecteur d'erreur aléatoire de n bit dans lequel la distance de Hamming est inférieure à t $w_H(e) < t$. $c = m.\hat{G} + e$

Déchiffrement :

1. $\hat{c} = c.P^{-1}$
2. $\hat{m} = D_F(\hat{c})$
3. $m = \hat{m}.S^{-1}$

Exemple :

Soit un Goppa code irréductible Γ de paramètre $(n,k,t) = (8,2,2)$.

$$S = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad G = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \hat{G} = SGP = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$S^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad P^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Pour le chiffrement, Prenons un message $m = (1,1)$

$$c = m\hat{G} + z = (1 \ 1) \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} + (1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)$$

$$c = (1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1)$$

Pour le déchiffrement, il faut d'abord calculer :

$$\hat{c} = cP^{-1} = (0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0)$$

En utilisant l'algorithme de décodage, on calcul :

$$\hat{m} = D_{\Gamma}(\hat{c}) = (0 \ 1)$$

Et enfin, il faut calculer le message d'origine :

$$m = \hat{m}S^{-1} = (0 \ 1) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1 \ 1)$$

4.1.4 QC-MDPC code

A cause de la lenteur en exécution du Goppa code surtout en décodage, et aussi pour que McEliece utilisant Goppa code soit résistant à la quantique, il faut une clé longue. Alors en 1960, un mathématicien Gallager a inventé un nouveau code du nom de LDPC (Low Density Parity Check) [90] [91]. Mais très vite, ce code fut remplacé par MDPC (Moderate Density Parity Check). Cependant, vu que McEliece par défaut utilise un Goppa code avec une longue clé, il faut donc aussi adapter l'algorithme à cette clé longue donc il faut utiliser un QC-MDPC (Quasi-cyclic-MDPC). La construction MDPC et QC-MDPC s'avère facile, le but est de trouver un vecteur aléatoire respectant un poids donné. Nous allons nous intéresser spécialement à la construction du code QC-MDPC. (n, r, w) -QC-MDPC est un code QC-MDPC possédant une matrice de parité aléatoire $H \in F_2^{r \times n}$ et de poids w . Comme c'est un code quasi-cyclique d'ordre p alors, $n = n_0 p$ et $r = p$. Ce qui signifie donc que :

$$H = [H_0 | H_1 | \dots | H_{n_0-1}] \quad (4.05)$$

H_i est donc une matrice de dimension $p \times p$. Chaque H_i possède donc un poids $w_i = w_H(H_i)$ (distance de Hamming). Et le poids de la matrice de parité H est défini par :

$$w = \sum_{i=0}^{n_0-1} w_i \quad (4.06)$$

La matrice génératrice est obtenue par la formule suivante :

$$G = \left[\begin{array}{c|c} & \begin{matrix} (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{matrix} \end{array} \right]$$

La sécurité du cryptosystème Mc-Eliece variante QC-MDPC réside sur les deux équations suivantes :

- Sécurité de clé : En sachant $h(x)$ il faut trouver des vecteurs non nuls : $h_0(x)$ et $h_1(x)$

$$\begin{cases} h_0(x) + h(x) \cdot h_1(x) = 0 \pmod{x^p - 1} \\ w_t(h_0) + w_t(h_1) \leq w \end{cases} \quad (4.07)$$

- Sécurité du message : En sachant $h(x)$ et $S(x)$, il faut trouver deux vecteurs non nuls : $e_0(x)$ et $e_1(x)$

$$\begin{cases} e_0(x) + h(x) \cdot e_1(x) = S(x) \pmod{x^p - 1} \\ w_t(e_0) + w_t(e_1) \leq t \end{cases} \quad (4.08)$$

4.1.5 Cryptosystème McEliece à base de graphe

L'algorithme de chiffrement et de déchiffrement de QC-MDPC est un peu particulier [92] [93] :

- Génération de clé :
 - Générer une matrice $H \in F_2^{r \times n}$, la matrice de parité de (n, r, w) -QC-MDPC capable de corriger t -erreur
 - Générer la matrice G en utilisant la formule de matrice génératrice du code MDPC dont la clé secrète est H et la clé publique est G .
- Chiffrement : soit le message à chiffrer $m \in F_q^k$
 - Déterminer un vecteur aléatoire d'erreur $e \in F_q^n$ de avec $w_t(e) = t$ (de poids t)
 - $x = m \cdot G + e$
- Déchiffrement :
 - Calcul $\Psi(x) = \text{bit_flipping}(x, H)$
 - On sait que $\Psi(x) = m \cdot G$ donc on aura $m = \Psi(x) \cdot G^{-1}$

- L'algorithme du bit flipping [94] [95]

bit_flipping(y, H) avec $y \in \{0,1\}^n$ et $H \in \{0,1\}^{(n-k) \times k}$

Calculer le syndrome $s = Hy^T$

Répéter

nb_equation_parity_bit_violé = compterNombreEquationViolant_j_eme_bit($j, H ; y$)

Pour ($j = 1, \dots, n$)

Si (nb_equation_parity_bit_violé $\geq b$)

corriger(y_j)

Calculer le syndrome $s = Hy^T$

Jusqu'à ce que (s un vecteur nul)

Remarques :

On sait que si le syndrome $s = Hy^T = \begin{pmatrix} s_1 \\ \vdots \\ s_{n-k} \end{pmatrix}$ et que si $s_t \neq 0$ alors t -ème équation est violée.

Pour compter le nombre d'équation pouvant violé j -ème bit, il suffit de considérer la matrice syndrome H .

compterNombreEquationViolant_j_eme_bit(j, H, y)

Calculer le syndrome $s = Hy^T$

nb_equation_parity_bit_violé = 0

Pour ($t = 1, \dots, n - k$)

Si($(H_{j,t} = 1)$ et ($s_t \neq 0$))

nb_equation_parity_bit_violé = nb_equation_parity_bit_violé + 1

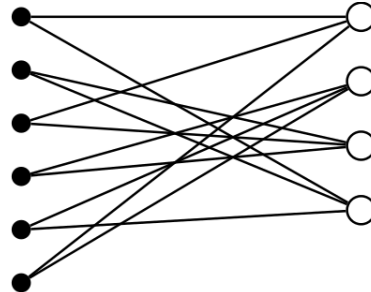
Il y a une classe de decoder qui calcul b est le maximum upc (unsatisfied parity check) noté Max_{upc} et ε un nombre de garde supposé très petit devant Max_{upc} . On calcul $b = Max_{upc} - \varepsilon$. Les algorithmes de flipping possèdent plusieurs versions mais ici on utilise la première version A.

Mais aujourd'hui, il y a plusieurs versions comme Decoder B, C1, C2, D, E, F. Le but est la même : corriger petit à petit les bits violés jusqu'à ce que le syndrome soit un vecteur nul.

Exemple :

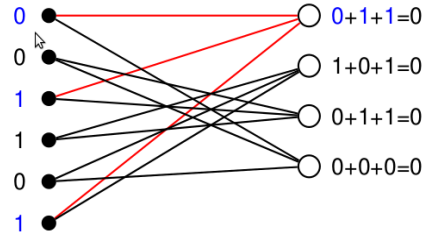
On définit une matrice de parité H représentée sous forme de graphe de Tanner :

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

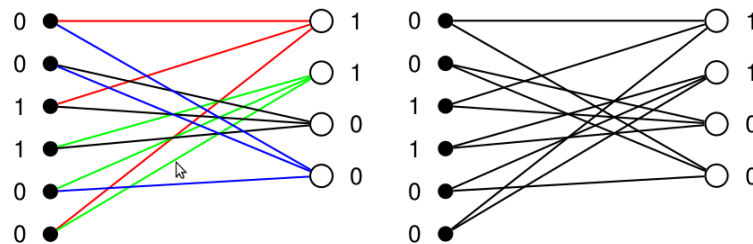


En comptant le syndrome pour un message : $y = (0,0,1,1,0,1)$ on voit bien que le syndrome est un vecteur nul

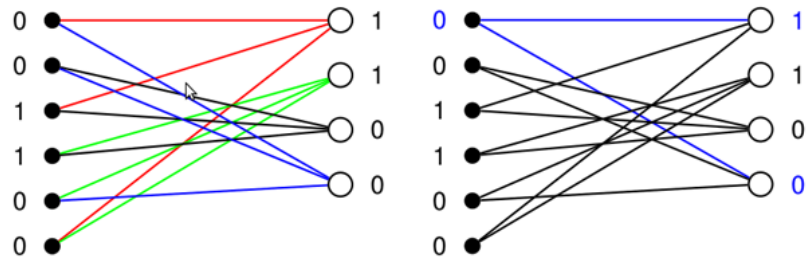
$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0+1+1=0 \\ 1+0+1=0 \\ 0+1+1=0 \\ 0+0+0=0 \end{pmatrix}$$



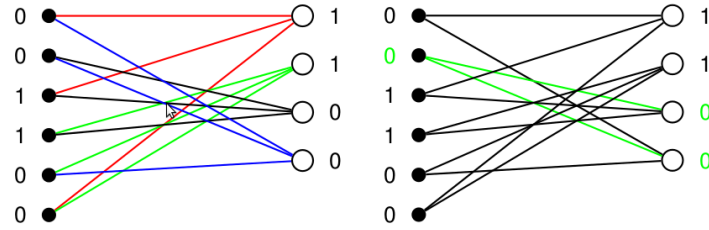
S'il n'y a pas d'erreur à la réception on voit bien que le syndrome doit être égal à zéro. Imaginons maintenant qu'à la réception on obtient $y = (0,0,1,1,0,0)$. Le bit_flipping est un algorithme inventé par Gallager pour décoder et corriger les erreurs. Prenons $Max_{upc} = 2$ et $\varepsilon = 0$; donc $b = 2$. Le calcul de syndrome s est comme nous voyons son graphe de Tanner et à droite le calcul de nombre maximal d'équation de parité pouvant violer j -ème bit.



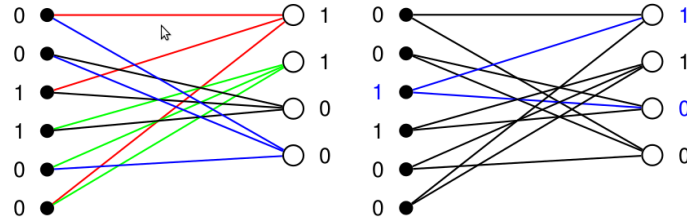
Pour $j = 1$; on a $nb_equation_parité_bit_violé = 1$; Pas de correction car $b = 2$



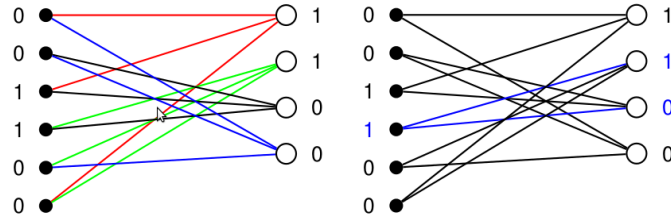
Pour $j = 2$; on a $\text{nb_equation_parité_bit_violé} = 0$; Pas de correction car $b = 2$



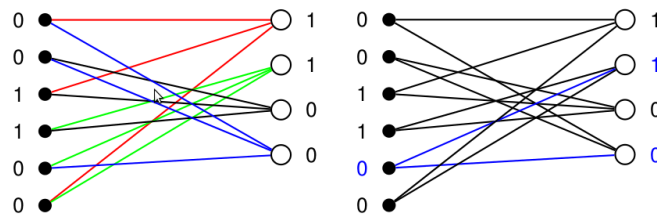
Pour $j = 3$; on a $\text{nb_equation_parité_bit_violé} = 1$; Pas de correction car $b = 2$



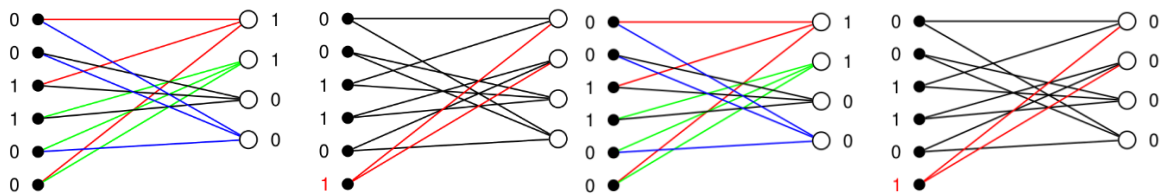
Pour $j = 4$; on a $\text{nb_equation_parité_bit_violé} = 1$; Pas de correction car $b = 2$



Pour $j = 5$; on a $\text{nb_equation_parité_bit_violé} = 1$; Pas de correction car $b = 2$



Pour $j = 6$; on a $\text{nb_equation_parité_bit_violé} = 2$; correction car $b = 2$



On refait ces opérations jusqu'à avoir un syndrome de la forme du vecteur nul. Les 2 graphes à gauches représentent la localisation et correction d'erreur. Les deux graphes à droite pour la nouvelle forme de matrice de parité et syndrome après la correction.

4.2 Cryptographie Salsa20-Poly1305

4.2.1 PKDF2

Le but de PKDF2 est de fournir une clé de dérivation constante à partir de passphrase ou password de taille aléatoire en utilisant des fonctions de fonction de hachage (Hash-based cryptography) [32] [63]

$$DK = \text{PBKDF2}(\text{PRF}, \text{Password}, \text{Salt}, c, \text{dkLen}) \quad (4.09)$$

PRF : pseudorandom Function

Password : Phrase de taille quelconque

Salt : la graine, bit aleatoire

c : nombre d'itération

dkLen : le nombre de bit de sortie voulue

DK : Derivated Key càd la clé obtenue de taille constante dkLen

Exemple : Pour la clé WPA2 on a :

$$DK = \text{PBKDF2}(\text{HMAC-SHA1}, \text{passphrase}, \text{ssid}, 4096, 256)$$

4.2.2 Salsa20

Le salsa20 est une famille de AES inventé par Daniel J. Bernstein en 2005 quand il a commencé à travaillé sur les cryptographies résistantes à l'attaque quantique. [96] [97] [98]

- **Mot :** un mot est un élément de $\{0, 1, \dots, 2^{32} - 1\}$ que l'on écrit en hexadécimal avec le symbole 0x. Exemple : $0xc0a8787e = 12 \cdot 2^{28} + 0 \cdot 2^{24} + 10 \cdot 2^{20} + 8 \cdot 2^{16} + 7 \cdot 2^{12} + 8 \cdot 2^8 + 7 \cdot 2^4 + 14 \cdot 2^0 = 3232266366$
- **L'addition :** L'addition se fait donc en hexadécimal mais on peut aussi effectuer le calcul à base décimale puis traduire le résultat en hexadécimal. Exemple : $0xc0a8787e + 0x9fd1161d = 0x60798e9b$
- **OU-Exclusif :** Supposons que $u = \sum_i 2^i \cdot u_i$ et $v = \sum_i 2^i \cdot v_i$. On peut généraliser par

$$u \oplus v = \sum_i 2^i (u_i + v_i - 2u_i v_i). \quad (4.10)$$

Exemple : $0xc0a8787e \oplus 0x9fd1161d = 0x5f796e63$

- **Rotation à gauche** : Une rotation à gauche de c -bit est noté par : $u \lll c$ et on le défini par :

$$u \lll c = \sum_i 2^{(i+c) \bmod 32} \cdot u_i \quad (4.11)$$

- **Quarterround** : C'est une fonction qui prend en entrée 4 mots et en sortie 4 mots également. On définit $quarterround(y) = (z_0, z_1, z_2, z_3)$ par :

$$z_1 = y_1 \oplus ((y_0 + y_3) \lll 7)$$

$$z_2 = y_2 \oplus ((z_1 + y_0) \lll 9)$$

$$z_3 = y_3 \oplus ((z_2 + z_1) \lll 13)$$

$$z_0 = y_0 \oplus ((z_3 + z_2) \lll 18)$$

(4.12)

Exemple : $quarterround(0xd3917c5b, 0x55f1c407, 0x52a58a7a, 0x8f887a3b)$
 $= (0x3e2f308c, 0xd90a8f36, 0x6ab2a923, 0x2883524c)$

- **Rowround** :

Si y est composé de 16 mots alors $rowround(y)$ est composé de 16 mots aussi. Si

$y = (y_0, y_1, \dots, y_{15})$ et $rowround(y) = z = (z_0, z_1, \dots, z_{15})$

$$(z_0, z_1, z_2, z_3) = quarterround(y_0, y_1, y_2, y_3)$$

$$(z_5, z_6, z_7, z_4) = quarterround(y_5, y_6, y_7, y_4)$$

$$(z_{10}, z_{11}, z_8, z_9) = quarterround(y_{10}, y_{11}, y_8, y_9)$$

$$(z_{15}, z_{12}, z_{13}, z_{14}) = quarterround(y_{15}, y_{12}, y_{13}, y_{14})$$

(4.13)

Exemple :

$rowround(0x08521bd6, 0x1fe88837, 0xbb2aa576, 0x3aa26365, 0xc54c6a5b, 0x2fc74c2f,$
 $0x6dd39cc3, 0xda0a64f6, 0x90a2f23d, 0x067f95a6, 0x06b35f61, 0x41e4732e, 0xe859c100,$
 $0xea4d84b7, 0x0f619bff, 0xbc6e965a)$
 $= (0xa890d39d, 0x65d71596, 0xe9487daa, 0xc8ca6a86, 0x949d2192, 0x764b7754,$
 $0xe408d9b9, 0x7a41b4d1, 0x3402e183, 0x3c3af432, 0x50669f96, 0xd89ef0a8,$
 $0x0040ede5, 0xb545fbce, 0xd257ed4f, 0x1818882d)$

- **Columnround** :

Si x est composé de 16 mots alors $columnround(x)$ est composé de 16 mots aussi. Si

$x = (x_0, x_1, \dots, x_{15})$ et $columnround(x) = y = (y_0, y_1, \dots, y_{15})$

$$\begin{aligned}
(y_0, y_4, y_8, y_{12}) &= \text{quaterround}(x_0, x_4, x_8, x_{12}) \\
(y_5, y_9, y_{13}, y_1) &= \text{quaterround}(x_5, x_9, x_{13}, x_1) \\
(y_{10}, y_{14}, y_2, y_6) &= \text{quaterround}(x_{10}, x_{14}, x_2, x_6) \\
(y_{15}, y_3, y_7, y_{11}) &= \text{quaterround}(x_{15}, x_3, x_7, x_{11})
\end{aligned} \tag{4.14}$$

On peut écrire aussi :

$$\begin{aligned}
& (y_0, y_4, y_8, y_{12}, y_1, y_5, y_9, y_{13}, y_2, y_6, y_{10}, y_{14}, y_3, y_7, y_{11}, y_{15}) = \\
& \text{rowround}(x_0, x_4, x_8, x_{12}, x_1, x_5, x_9, x_{13}, x_2, x_6, x_{10}, x_{14}, x_3, x_7, x_{11}, x_{15})
\end{aligned} \tag{4.15}$$

Exemple :

Columnround (0x08521bd6, 0x1fe88837, 0xbb2aa576, 0x3aa26365, 0xc54c6a5b, 0x2fc74c2f, 0x6dd39cc3, 0xda0a64f6, 0x90a2f23d, 0x067f95a6, 0x06b35f61, 0x41e4732e, 0xe859c100, 0xea4d84b7, 0x0f619bff, 0xbc6e965a)
= (0x8c9d190a, 0xce8e4c90, 0x1ef8e9d3, 0x1326a71a, 0x90a20123, 0xead3c4f3, 0x63a091a0, 0xf0708d69, 0x789b010c, 0xd195a681, 0xeb7d5504, 0xa774135c, 0x481c2027, 0x53a8e4b5, 0x4c1f89c5, 0x3f78c9c8)

- **Doubleround :**

Si x est composé de 16 mots alors doubleround(x) est composé de 16 mots aussi.

$$\text{doubleround}(x) = \text{rowround}(\text{columnround}(x)) \tag{4.16}$$

Exemple :

Doubleround (0xde501066, 0x6f9eb8f7, 0xe4fbbd9b, 0x454e3f57, 0xb75540d3, 0x43e93a4c, 0x3a6f2aa0, 0x726d6b36, 0x9243f484, 0x9145d1e8, 0x4fa9d247, 0xdc8dee11, 0x054bf545, 0x254dd653, 0xd9421b6d, 0x67b276c1)
= (0xccaaaf672, 0x23d960f7, 0x9153e63a, 0xcd9a60d0, 0x50440492, 0xf07cad19, 0xae344aa0, 0xdf4cfdfc, 0xca531c29, 0x8e7943db, 0xac1680cd, 0xd503ca00, 0xa74b2ad6, 0xbc331c5c, 0x1dda24c7, 0xee928277)

- **LittleEndian :**

C'est une représentation en hexadécimal qui consiste à écrire à l'inverse tous les nombres hexadécimaux. On prendra dans le salsa20, b un mot de 4-byte.

$$\text{Si } b = (b_0, b_1, b_2, b_3) \text{ donc } \text{littleendian}(b) = b_0 + b_1 \cdot 2^8 + b_2 \cdot 2^{16} + b_3 \cdot 2^{24} \tag{4.17}$$

Exemples :

Littleendian (86, 75, 30, 9) = 0x091e4b56.

Littleendian (255, 255, 255, 250) = 0xfaffffff.

- **Fonction de Hachage Salsa20 :**

Si x est une séquence de 64-byte alors $salsa20(x)$ est aussi une séquence de 64-byte par :

$$Salsa20(x) = x + \text{doubleround}^{10}(x) \quad (4.18)$$

et l'entrée est représentée en littleendian

$$x_0 = \text{littleendian}(x[0], x[1], x[2], x[3]),$$

$$x_1 = \text{littleendian}(x[4], x[5], x[6], x[7]),$$

$$x_2 = \text{littleendian}(x[8], x[9], x[10], x[11]),$$

.

.

$$x_{15} = \text{littleendian}(x[60], x[61], x[62], x[63]).$$

Exemples :

$$\begin{aligned} &Salsa20(211, 159, 13, 115, 76, 55, 82, 183, 3, 117, 222, 37, 191, 187, 234, 136, 49, 237, 179, 48, \\ &1, 106, 178, 219, 175, 199, 166, 48, 86, 16, 179, 207, 31, 240, 32, 63, 15, 83, 93, 161, 116, 147, \\ &48, 113, 238, 55, 204, 36, 79, 201, 235, 79, 3, 81, 156, 47, 203, 26, 244, 243, 88, 118, 104, 54) \\ &= (109, 42, 178, 168, 156, 240, 248, 238, 168, 196, 190, 203, 26, 110, 170, 154, 29, 29, 150, 26, 150, \\ &30, 235, 249, 190, 163, 251, 48, 69, 144, 51, 57, 118, 40, 152, 157, 180, 57, 27, 94, 107, 42, 236, \\ &35, 27, 111, 114, 114, 219, 236, 232, 135, 111, 155, 110, 18, 24, 232, 95, 158, 179, 19, 48, 202) \end{aligned}$$

- **Salsa20 avec une clé k**

Soit k une séquence de 32byte ou 16byte et n une séquence de 16byte, $salsa20_k(n)$ est un mot de 64-byte. On a la formule :

$$salsa20_k(n) = salsa20(\tau_0, k, \tau_1, n, \tau_2, k, \tau_3) \text{ avec}$$

$$\tau_0 = (101, 120, 112, 97) \text{ et } \tau_1 = (10, 100, 32, 49) \text{ et } \tau_2 = (54, 45, 98, 121) \text{ et}$$

$$\tau_3 = (116, 101, 32, 107)$$

(4.19)

- **Le chiffrement salsa20 :**

On veut chiffrer ℓ -byte de message m , on a la formule suivante :

$$(c[0], c[1], \dots, c[\ell - 1]) = salsa20_k(n) \oplus (m[0], m[1], \dots, m[\ell - 1]) \quad (4.20)$$

4.2.3 Poly1305

Poly1305 [99] [100] a été inventé par Daniel J. Bernstein et Google l'a normalisé dans plusieurs RFC 7539. Le but est de vérifier l'intégrité de données en générant un MAC (Message authentication Code). L'algorithme Poly1305 est utilisé dans le protocole que Google utilise aujourd'hui notamment avec TLS/SSL et HTTPS. [101] [102]

```
clamp(r): r &= 0xffffffffc0xffffffffc0xffffffffc0xffffffff
poly1305_mac(msg, key):
  r = (le_bytes_to_num(key[0..15])
  clamp(r)
  s = le_num(key[16..31])
  accumulator = 0
  p = (1<<130)-5
  for i=1 upto ceil(msg length in bytes / 16)
    n = le_bytes_to_num(msg[((i-1)*16)..(i*16)] | [0x01])
    a += n
    a = (r * a) % p
  end
  a += s
  return num_to_16_le_bytes(a)
end
```

4.3 Etude pratique d'un cryptosystème quantique

Comme le célèbre cryptosystème PGP très employé dans le monde de cryptographie (notamment utilisé dans le monde de l'OpenSource vu que la vérification d'installation des paquets en linux se vérifie par ce cryptosystème). Il faut inventer un nouveau cryptosystème post-quantique appelé maintenant librairie PQP (Post Quantum PGP) reconnu par NIST comme la cryptographie post-quantique à partir de 28 septembre 2016. En fait, la librairie n'est pas encore en production et représente vu que le délai maximal des choix de cryptosystème post-quantique, le délai maximal pour la proposition d'implémentation ou de proposition sera l'année 2017. Et les algorithmes ne

sont employés que l'année 2020, le début de 5G-Iot et la fin IP-V4. On dit ces algorithmes H2020 ou Horizon 2020. Comme PGP, le cryptosystème PQP est un cryptosystème hybride : l'échange de clé se fait par une cryptographie asymétrique et le chiffrement et déchiffrement se fait par une cryptographie symétrique. Ce cryptosystème utilise le protocole dicté par Fujisaki-Okamoto dont le principe réside sur la génération d'un nombre aléatoire appelé Token. PGP utilise les algorithmes post-quantique proposé par Daniel Bernstein : utilisation de QC-MDPC McEliece pour l'échange de clé et Salsa20 pour le chiffrement et déchiffrement et enfin Poly1305 pour l'intégrité de donné. Imaginons que bob veut envoyer un message à Alice dans un canal sécurisé.

- **Chiffrement :**

- Bob génère un nombre aléatoire Token nommé T
- Bob demande la clé publique d'Alice et crypte le Token via QC-MDPC McEliece
- Le Token T sert à générer (via PKDF2) la clé k_1 lors de chiffrement symétrique et k_2 pour générer le MAC
- Calculer le MAC via poly1305
- Chiffrer les message en utilisant message et Token T et le MAC via Salsa20
- Envoyer à Alice : Token chiffré + Message chiffré + MAC chiffré

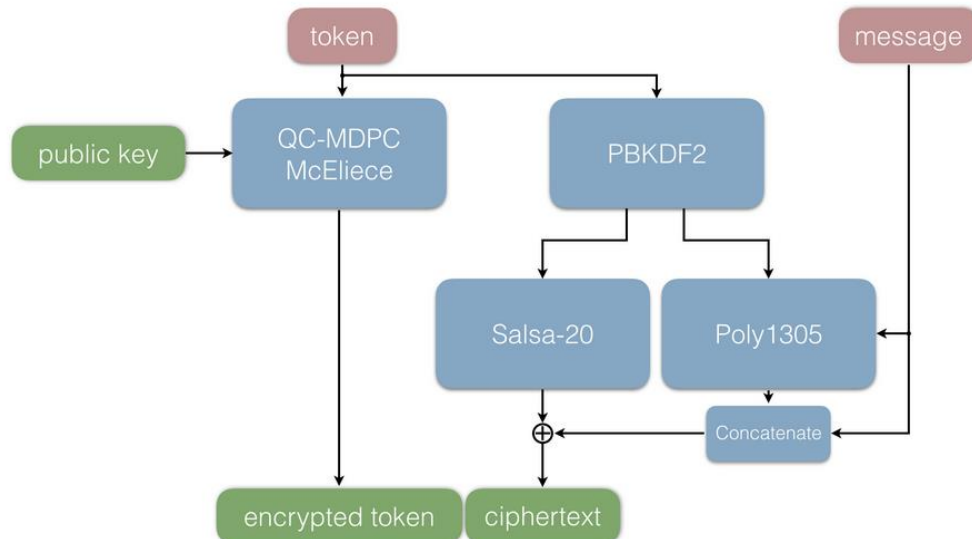


Figure 4.03 : Chiffrement avec PQP

Le chiffrement se fait donc en utilisant un cryptosystème hybride : L'échange de clé et l'encryptions de Token se fait par QC-MDPC McEliece. Ensuite le Token va générer un DK (Derivated Key)

pour pouvoir faire un algorithme de chiffrement (ici Salsa20) et une fonction de hachage (ici Poly1305). L'avantage de l'utilisation d'une cryptographie hybride est que l'échange de clé utilise une cryptographie à clé publique et le chiffrement se fait par l'utilisation d'une cryptographie à clé secrète. Dans cette méthode, on peut combler même le désavantage du cryptosystème McEliece, car le fait d'ajouter des codes d'erreurs va augmenter considérablement la taille du message alors, le chiffrement du Token seulement est considéré par le McEliece.

- **Déchiffrement :**

- Alice décrypte le Token vector T.
- En utilisant le PBKDF2, Alice calcule : la clé k_1 lors de chiffrement symétrique et k_2 pour générer le MAC
- Alice décrypte le message et peut extraire le MAC du message que je note MAC_{rec}
- Alice génère un MAC à partir de la fonction Poly1305 et le message reçu que je note MAC_{gen}
- Si le MAC reçu MAC_{rec} et le MAC généré MAC_{gen} sont égaux Alice accepte le message sinon elle refuse.

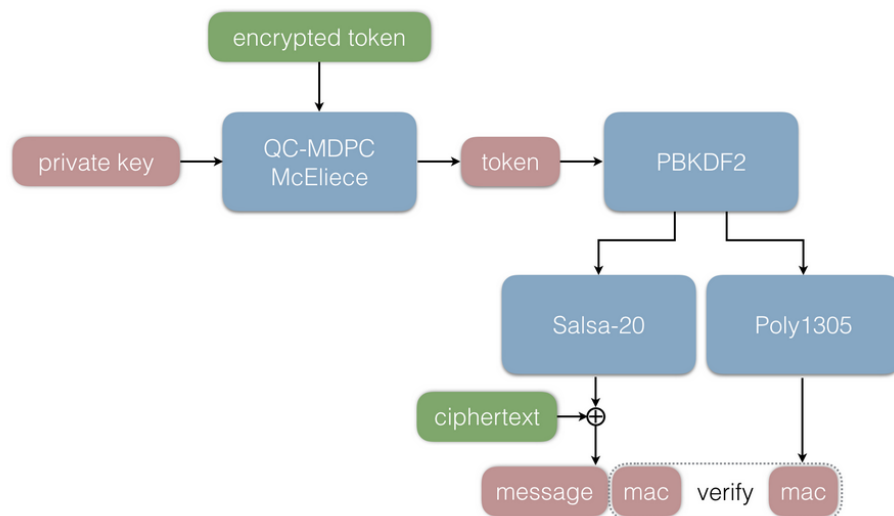


Figure 4.04 : *Déchiffrement en utilisant PQP*

Le déchiffrement est l'étape inverse du chiffrement. L'échange de clé se fait encore par le cryptosystème QC-MDPC mais la seule différence, il faut d'abord déchiffrer le Token puis faire un algorithme de de génération de clé PBKDF2. Les deux clés générer doivent être exactement la même que lors de chiffrement car ils ont la même Token au départ. Le déchiffrement montre si le MAC

est égal ou non après avoir effectué la clé correspondante par une fonction de hachage Poly1305. L'autre clé sera la clé secrète pour déchiffrer l'information par le biais de l'algorithme Salsa20 qui est aussi un algorithme Post-Quantique.

- **Format de clé**

```
-----BEGIN PQP PUBLIC KEY-----
MIICXgOCAIoHMFAsKjyqD4MKBPfKLCdUBAqC6rY4jd10k/RQc4MiGfkUSw6hBh41eFa1XogH9MN
b49TPLQRZYBXygp7eGP1oUM4Pqvvdww01UoTPNdWYeAiqEpOe4nP7sq+fir54n184I/ArMdCsUyo
hgYtCVumC0XkYm1Ww8tsW+RU94gQoQd7pLvgs98ah0gTNARRjW8/yALfGrB1pV7ZuLUScfNIFq8q
ZIZE8P80VK6kS0tcy+k4h9vHeB6QGP1PGB/jQ7mz8jDzw7v3m5QfnS1n1HvBSYWU/hADIIy13uh2
mEJmtIzDbwOZ7v40JtDXrKgK9iMJ40jCtntbmdAMSNwdMhNp2mH205a7b0MoELVILx6CTpjB64D2
toFwXw1867QEgBCk4imZHxMgnLw9TxnM8g+5gQzFC5BCI6afaGS91ZXwbM+ssZN2DbRZIEVS7rI
12nu12rqquMC0buMM0Yt6ebD3bMRTeuUY3KLmEkUVjn3fTg7YUm82UuyGmG3cKqB0AZb24yswX17
z3b0rMTMrggQXw0KPR3AoPh49+PG9pt9ySRp/9KZ8k+apXBtvxC0Ix3J+6WYeB9zGTnu841j1+WD
LTBW5ePqglxGjown91Z1mw2Rgps17o6wSG51b/d+B6hTw8w6KIowsQywEyDuF45B7U6W5EE8kgS1
9S01PKqkpYd9YV7oJzpYLFuS6dDW71WavV9DdW29uN0n501BAk79SbAjs1iSojSVv5ZrYMPp16Sp
0Y0kRoH/WeLL+xBwJtqMNRi134nH/B45ibEibCbFp04rEDUs01aYAA==
-----END PQP PUBLIC KEY-----
```

Figure 4.05 : *Exemple de Clé publique*

La clé publique est obtenue en utilisant la matrice Génératrice G de la forme :

$$G = \left[\begin{array}{c|c} & \begin{matrix} (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{matrix} \end{array} \right] \quad (4.21)$$

Avec $H = [H_0|H_1| \dots |H_{n_0-1}]$ la matrice de parité QC-MDPC formé aléatoirement et respecter certains conditions. La taille de la clé publique est : 880 octets. On peut demander n'importe quelle clé publique d'une personne. C'est le principe du cryptosystème à clé publique : Tout le monde peut chiffrer le message mais seule la personne destinée peut le déchiffrer en utilisant sa clé secrète.

re

Figure 4.06 : *Exemple d'une clé privée*

La clé privée est la représentation de la matrice H citée précédemment (matrice de parité). La clé privée sert à déchiffrer le message. Comme nous voyons dans la figure 4.06, la taille de la clé privée est : 2526 octets (2,5 kibi-octets). Le rôle essentiel de cette clé privée est de déchiffrer le Token pour pouvoir déchiffrer le message entier. Cette clé privée doit rester secrète et il est totalement impossible de la déterminer à partir de la clé publique.

- **Chiffrement**

Imaginons maintenant que l'on veut chiffrer le message : « *this is a really secret message that is padded with some random.* »


```

-----BEGIN PQP MESSAGE-----
MIIFHg0CAloH0jScypeSXUoEDjfC+E3GGYbQ7t2QQ8Z7LtfPG0S0jySJWuWszQ2F593vDr41hQFV
dpgdSElPrWmiAxbqZ11fj3DVXCTZtu5DwXBijtVKg55xs/fu9rh6RvywIKdRLTpxJfVrN1oGA7rs
oaPvToZP0ziXdymst4Z3dDc5vcyP0RQXZz3KmhLKowFtrHpPx7HMDi2iXy2ohKGBafdtPzS0Hek
K8JgtLQh1C/cQbXkd5uWniWn/qSei6zKVp0C07PvqLadJCJI2ruFue2Lie+AmQGVmn4DK8X5c+I8
4BEVTY6PtZ819vyQm0AR00qSNr/6qiV0u1X7VFLGw+tvYjrDAvmiWTujQ7uY9Qs2ef0idk+4GmnK
JqImt+ExnKedH805NWxZewHnFoU23gL3Qz1e0DnHYV0QxfvUtQnuZMRuHgbrc+av22pi0C+aj0qR
JW4vcqNEazeD6usot28Uo7tpnqs1kAssHqGAQ0rAtDkdogpq5ntQidb4yheaaj1orgcT7/VyugKbH
Di0NeyoG/wo9vi5aCDlw0e0KPHKat0wvR21FYSCqtd9Gmc6/McDGYTwaJLZ4NK53ETnaJh46X9jd
OZTL9eS0KrbElZj1IMWAVXZgU4651GoENZVDNCV9AgzMsKu0o/VV2Djh8Hw/Ggx0sAlpbjc+AuD4
xEa3gETXKdmsq/PjATe9c9+5I0o017rIjwlc+Wc5w4NGJuBrm0MntkzEzVyteXsn00njiNdrxRks
gaUcpkwleLawIZh7gs1X3SRI7+WvBy/8t/n9LxERC6/rkyJbC5kAA0CAloHHTIcwG1S+00bmFb8
C5uYT47XcP/eP+U9tjVswkgj/wu/RjmqGuGISw8ys2vmi79429gvXQRwLiH3NaPP8WpF2vo52H0
gVVLyqmwTmG5gLKwxEfr220Ge/nNnj49Rk57nhpp+i2EUFmWbigrAcClcVxeB5BY1x1MqVeTJeU
OeR1RJaaBFk2GRETnAPST3RsyLwz9dW0njZkeQGPG+rIXp5J/exUsn+WSq/omYivTujCHACZ1Fxt
Bx0fD1PXxGVhGrXodENAn1R0AeG/E3g/M++vfTaG2nba+kk0/21rMKfWTop/cvpyzHk9gNDc5EXM
A1MQUI7z9/defLHVIwo6u5xHjAx3c+qZ2YwEJmaar7c70NUfjv0PhGQsH9Pcs18BVv7WyswDT8/C
wtKCv9PY5BVUBvp2wKCFmhuPdNYxzA3F/zPL+ryFUjMrdbNwYaI1hQXdeUY0DzM6mWPEfLgWnIG
PKX6CmsbJNQs3+GC31ps7GkCZkYdBoS1Bf9faFnLCszx2AMK0x32ZKVyU4v0WzvrNF1VQyq6NQyh
Ap7phcIvq9DkBSdWe2pEpbctYbUqVLeZ5tpz0zCFwrdDHvvDSg40A3J3QEvSA4V1fCzQY6QG9EGC
8m1XorT+aK1vJz6gW0ZcurwrM307cY0x9yC0QACQPItWXXt8E6fpXjxgJs4mMwnWGoW7H51xzIRH
m+VIP0qdHCvEzI2180qE0oE0fIh8MBLFWktovkjy94US6AGTxZ/GP2o8ALKvRfVb603X5m8PjBlj
JGPYik2nqjqIJXK95F9omr4FgARgSuda/5bzJK/tJUuixF0WwaimI+WtgJDsIV8ve1kaaVvxL8xz
K7P10CNoMaZYC+z/GkenwvNr0f+uGiPYShao0/Ie7NhC2C2tj610ENmc+0J1zy1qLE1ApJ01VL3
KHde
-----END PQP MESSAGE-----

```

Figure 4.07 : Exemple de message chiffré

Dans le premier exemple, nous allons chiffrer un message : « *this is a really secret message that is padded with some random.* ». Après avoir générer la clé publique et la clé secrète via le cryptosystème QC-MDPC. Le Token va être chiffré en utilisant la méthode de chiffrement de QC-MDPC : $x = m.G + e$ et sera envoyé dans le réseau. Le chiffrement QC-MDPC est décrit dans le paragraphe 4.15. L’algorithme PBKDF2 va générer deux clés que nous ne pouvons voir. L’une va permettre de chiffrer le message en Salsa20 selon le procédé décrit dans le paragraphe 4.2.2 et l’autre subira une fonction de hachage Poly1305 selon le procédé décrit dans le paragraphe 4.2.3 pour produire le MAC du message. Ces deux algorithmes sont post-quantiques et sont proposés par Daniel J. Bernstein en 2014 et sont utilisés dans les algorithmes TLS/SSL et HTTPS de nos jours.

Le déchiffrement comment par le déchiffrement du Token crypté en utilisant l'algorithme de bit_flipping décrit dans le paragraphe 4.15 par :

- Calcul $\Psi(x) = \text{bit_flipping}(x, H)$
- On sait que $\Psi(x) = m \cdot G$ donc on aura $m = \Psi(x) \cdot G^{-1}$

Une fois que le Token est obtenu ; on peut régénérer les deux clés pour vérifier le MAC du message (en utilisant Poly1305) et la clé secrète de déchiffrement (en utilisant Salsa20). Le déchiffrement selon Salsa20 est décrit par le paragraphe 4.2.2

- **Commande pour chiffrer un vidéo**

On peut constater également différent exemple avec le chiffrement de vidéo :

```
$ time ./pqp --encrypt libpqp.mov --pubkey pub --output libpqp.enc
Encrypting: libpqp.mov

real    0m11.331s
user    0m9.293s
sys     0m1.715s

$ time ./pqp --decrypt libpqp.enc --privkey priv --output libpqp2.mov
Decrypting: libpqp.enc
MAC verified.

real    0m11.422s
user    0m8.633s
sys     0m2.446s

$ ls -la libpqp*
-rw-r--r--  1 carl  staff  258900672 17 Jul 00:57 libpqp.enc
-rw-r--r--@ 1 carl  staff  191652442 16 Jul 23:22 libpqp.mov
-rw-r--r--@ 1 carl  staff  191652442 17 Jul 00:58 libpqp2.mov

$ md5 libpqp2.mov
MD5 (libpqp2.mov) = 88ebe9d8aa74ebba7c5de6faa048af46

$ md5 libpqp.mov
MD5 (libpqp.mov) = 88ebe9d8aa74ebba7c5de6faa048af46
```

Figure 4.08 : *Commande linux pour le teste de chiffrement de vidéo*

La commande PQP en linux permet à la fois de générer la paire de clé : publique, privée et peut chiffrer, déchiffrer un message. Avec la commande time du système Linux, nous pouvons évaluer le temps de chiffrement et déchiffrement de vidéo. Pour un fichier de 191 Kio le temps réel d'exécution est de 11.31seconde. La deuxième commande sert à effectuer le même procédé pour la décryptations. On peut constater le temps d'exécution, et surtout, la vérification si le MAC est le même ou non. Le fichier vidéo libpqp.mov va être chiffrer en libpqp.enc et après le déchiffrement on obtient le fichier vidéo libpqp2.mov. La vérification de la présence de ses fichiers se fait par la commande ls. Maintenant, il faut vérifier si les deux fichiers sont exacts. De ce fait, nous utilisons la commande md5 (quelquefois md5sum) pour faire un résumé par le biais de MD5. Nous constatons que les condensats des deux fichiers sont exacts.

4.4 Conclusion

La cryptographie à base de correcteur d'erreur ajoute des erreurs corrigibles aux messages. Le cryptosystème de McEliece a été spécifié par NIST. Sa première version utilise le Goppa code puis le code LDPC et maintenant NIST propose l'utilisation de QC-MDPC. Cependant, le principal inconvénient de Chiffrement, ce que la taille du message est deux fois plus grand que la taille réelle. Le mieux est donc d'utiliser un chiffrement hybride, par l'utilisation de PKDF2 et des algorithmes de chiffrement post-quantique et fonction de Hachage post-quantique proposés par Daniel J. Bernstein Salsa20 et Poly1305. PQP (Post-Quantum PGP) est le nom de ce cryptosystème une cryptographie H2020 avec les opérations classiques (chiffrement, déchiffrement, génération de clé).

CONCLUSION GENERALE

Dans ce travail, on a pu constater que la cryptographie doit respecter les conditions CAIN (Confidentialité, Authenticité, Intégrité et Non-répudiation). Cette science peut être caractérisée en 3 parties : cryptographie ancienne (utilisation de permutation et de substitution), cryptographie moderne et cryptographie quantique. Les cryptographies modernes peuvent être à clé secrète ou à clé publique. La cryptographie à clé secrète n'utilise qu'une seule clé pour le chiffrement et déchiffrement et peut être en bloc ou par flots par l'utilisation des modes ECB, CBC, OFB, CFB. La cryptographie publique possède de clé différent pour le chiffrement et le déchiffrement et peut-être : RSA, El Gmal, PGP, RABIN, Cryptographie à base de correcteur d'erreur ou Menezes-Vanstone. Ces algorithmes dépendent du problème calculatoire de factorisation des entiers (et ses variant DLP, ECDH, DH) sont facilement cassable (crackable) en utilisant des ordinateurs quantiques. Ces ordinateurs utilisent des méthodes perfectionner comme le Quantum Fourier Transform, Grover Algorithm pour l'algorithme de recherche et Shor Algorithm pour la factorisation des entiers en nombres premiers. Il faut utiliser des problèmes calculatoires beaucoup plus difficiles : Latticed Based Cryptography, Multivariate Cryptography, Hash based Cryptography, Code based Cryptography et SIDH. Une de proposition que nous avons montrée dans ce mémoire est l'utilisation de cryptosystème PQP (Post-Quantum Cryptography). PQP est un cryptosystème hybride utilisant à la fois QC-MDPC avec l'utilisation de PKDF2, Salsa20 et Poly1305. Le QC-MDPC sera utilisé généralement dans le futur réseau 5G Iot, surtout dans les cryptographies embarquées dans des matériels. Il fait partie également des algorithmes H2020 Post-Quantique. L'inconvénient de l'utilisation de ce cryptosystème est expliqué par son inefficacité contre les attaques de type implémentation. Pour cela, comme travail futur, il faut créer un algorithme à la fois à l'ordinateur quantique et l'implémentation sans que le matériel utilisé ne soit trop cher comme l'utilisation des algorithmes SIDH. Les avantages de SIDH concernent aux avantages apportés par l'utilisation des courbes elliptiques en cryptographie : son robustesse face à des attaques de types implémentations notamment : le SPA ou Single Power Analysis, le DPA ou Differential Power Analysis ou le SCPA ou Slide Channel Power Analysis. Comme futur travail aussi, il faut étudier l'impact des calculs parallèles sur ordinateurs quantiques faces à ses algorithmes post-quantiques proposés

ANNEXE

A1 Algorithme de décodage de Patterson

Quand nous utilisons le GoppaCode pour le code correcteur d'erreur, il faut avoir un algorithme de décodage. Cet algorithme s'appelle algorithme de Patterson.

Algorithm 5 PATTERSON ALGORITHM FOR DECODING BINARY GOPPA CODES

Input: Syndrome $s = S_{\mathcal{C}}(z)$, Goppa code with an irreducible Goppa polynomial $g(z)$

Output: Error locator polynomial $\sigma(z)$

```
1: if  $s \equiv 0 \pmod{g(z)}$  then
2:   return  $\sigma(z) = 0$ 
3: else
4:    $T(z) \leftarrow s^{-1} \pmod{g(z)}$ 
5:   if  $T(z) = z$  then
6:      $\sigma(z) \leftarrow z$ 
7:   else
8:      $R(z) \leftarrow \sqrt{T(z)} + z$ 
9:      $(a(z), b(z)) \leftarrow \text{EEA}(R(z), G(z))$ 
10:     $\sigma(z) \leftarrow a(z)^2 + z \cdot b(z)^2$ 
11:   end if
12: end if
13: return  $\sigma(z)$ 
```

A2 Certaines classes de complexité algorithmique

Classical Complexity Class	Intuitive Meaning	Examples
P (or PTIME)	Polynomial-time: the running time of the algorithm is, in the worst case, a polynomial in the size of the input. All problems in P are tractable.	Multiplication
NP	Nondeterministic polynomial time: a candidate answer can be checked for correctness in polynomial time.	Factoring composite integers
NP-complete	A subset of problems in NP that can be mapped into one another in polynomial time. If just one of the problems in this class is shown to be tractable, then they must all be tractable.	Scheduling Satisfiability Traveling salesman problem
ZPP	Can be solved, with certainty, by PTMs in average case polynomial time.	
BPP	Can be solved in polynomial time by PTMs with probability $> 2/3$. Probability of success can be made arbitrarily close to 1 by iterating the algorithm a certain number of times.	Problems in BPP are tractable

Les problèmes mathématiques sont évalués selon leur complexité en temps et en espace. Ce tableau illustre le nom de certains complexité algorithmiques.

BIBLIOGRAPHIE

- [1] D. Boneth, S. Victor, « *A Graduate Course in Applied Cryptography* », August 17 2015
- [2] J. Hoffstein, P. Jill, J. H. Silvermann, « *An Introduction to mathematical cryptoraphy* », Springer 2008
- [3] P. Broyer, « *Cryptographie 2013* », Université Paris 13, Janv-Fev 2013
- [4] B. Khaled, Mémoire de Fin d'Etudes « *Master Cryptographie et sécurité de l'information* », Université Mohammed V - Agdal Faculté des Sciences – Rabat, 20 sept 2010
- [5] C. Anne, « *Attaques de cryptosystemes à mots de poids faible et construction de fonctions t-resilientes* », thèse de doctorat de l'université Paris 6 spécialité Informatique, 10 oct 1996
- [6] S. Bruce, « *Applied Cryptography* », New York, 2nd Ed., John Wiley & sons Inc., 1998
- [7] J. Stern, G. Louis, N. Phong, P. David, « *Conception et preuves d'algorithmes Cryptographiques* », Cours de magistère M.M.F.A.I.E Ecole normale supérieure, Ed. 2004
- [8] Viennot, Laurent, « *Cryptographie appliquée* », Paris, 2è me Éd., International Thomson Publishing France, 1997 pour l'édition francophone
- [9] J. Stern, « *La science du secret* », Paris, Éditions Odile Jacob, 1998
- [10] P. Thomas, « *Arithmétique modulaire pour la cryptographie* », Université Montpellier II, thèse specialité Informatique,
- [11] F. Arnault, « *Théorie des nombres et cryptographie* », Cours D.E.A Université Limoges, 07 mai 2002
- [12] D. Stinsons, « *Cryptographie, Théorie et pratique* », Vuibert, Paris 2001
- [13] D. E. Knuth, « *The art of computer programming - Fundamentals algorithms* », Editions 1, 1962
- [14] D. E. Knuth, « *The art of computer programming - Seminumerical Algorithms* », Editions 2, 1963
- [15] D. E. Knuth, « *The art of computer programming - Sorting and Searching* », Editions 3, 1972
- [16] M. Marine, « *Preuve d'analyse de de sécurité en cryptographie à clé secrète* », Doctorat de l'Université Limoges spécialité Mathématiques, 30 mars 1992
- [17] G. Castagnos, « *Quelques schémas de cryptographie asymétrique probabiliste* », Thèse Université de Limoges, 03 octobre 2006
- [18] S. Yannick, « *Primitives et protocoles cryptographiques à sécurité prouvée* », Université de Versailles Saint-Quentin-en-Yvelines, spécialité informatique, Laboratoire PRISM, 01 juillet 2009

- [19] J. Doumas, J. Rock, E. Tannier, V. Sébastien, « *Théorie des codes compression, cryptage, correction* », Dunod, Paris, 2007
- [20] P. Rafael, S. Abhi, « *A course in cryptography* », January 2010
- [21] Bergen, « *Preproceedings The International Workshop on Coding and Cryptography* » April 15–19, 2013
- [22] B. Christiana, « *Analyse de Fonctions de Hachage Cryptographiques* », These de doctorat de L'université Pierre et marie curie, Ecole doctorale Informatique, Télécommunications et Electronique (Paris), 07 dec 2012
- [23] K. H. Rosen, « *An introduction to cryptography* », Taylors & Francis group 2007, 2éme édition
- [24] O. Goldreich, « *Modern Cryptography, Probabilistic Proofs and Pseudorandomness* », Department of Computer Science and Applied Mathematics Weizmann Institute of Science, Rehovot, Israel. May 10, 2000
- [25] B. Daniel, G. Dartois, « *Cryptographie Paris 13* », 01 oct 2010
- [26] M. Videau, « *critères de sécurité des algorithmes de chiffrement à clé secrète* », Thèse de Doctorat de l'Université paris 6, spécialité Informatique, 10 nov 2005
- [27] J. Katz, Y. Lindell, « *Introduction to modern cryptography* », Taylor & Francis group 2008
- [28] K. Czeslaw, K. Mirosław, S. Marian, Modern « *Cryptography Primer Theoretical Foundations and Practical Applications* », Ed. Springer, Berlin 2013
- [29] R. Boris, F. Audrey, « *Basics for contemporary cryptography for it practitionners* », Siberian State University of Telecommunications and Computer Science, Russia 2005
- [30] C. Benoit, « *Cryptographie à clé publique : Constructions et preuves de sécurité* », Thèse université Paris, 2006
- [31] A. Salomaa, « *Public-Key Cryptography* », Ed. Springer, Sept 1996
- [32] O. Goldreich, « *Foundations of cryptography : Basics Techniques* », Cambridge 2004
- [33] G. Goos, J. Hartmanis, J. van Leeuwen, « *Topics in cryptology CT-RSA* », The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13-17 2003
- [34] S. Burnett, S. Paine, « *RSA's security Official Guide to Cryptography* », The McGraw-Hill Comanie 2001
- [35] S.C. Coutinho, « *The mathematics of Ciphers number theory and RSA Cryptography* », Rio Janeiro, 18 July 1998
- [36] G. Daniel, J.M. Thomas, « *PGP & GPG : Assurer la confidentialité de ses e-mails et fichiers* », Ed. Eyrolles 2006

- [37] O. Goldreich, « *Modern Cryptography, Probabilistic Proofs and Pseudorandomness* », Department of Computer Science and Applied Mathematics, Ed. Springer Berlin 1999
- [38] M. G. Parker, « *Cryptography and Coding* », Dec 2009, Ed. Springer Berlin 2009
- [39] Z. Cao, « *New Directions of Modern Cryptography* », Taylor & Francis Group 2012
- [40] D. Micciano, « *Theory of cryptography* », Zurich, zwitzerland, 9-11-February 2010, Ed. Springer
- [41] R. Dumond, « *Cryptographie et Sécurité informatique* », Cours Université Liège, Faculté des Sciences Appliquées, 2009-2010
- [42] D. J. Bernstein, J. Buchmann, E. Dahmen, « *Post-Quantum Cryptography* », Springer 2009
- [43] R. D. Alexandre, « *Bringing Theory Closer to Practice in Post-quantum and Leakage-resilient Cryptography* », École Polytechnique Fédérale De Lausanne, Thèse No 6807, 13 Nov 2015
- [44] D. H. Phan, « *Sécurité et efficacité des schémas cryptographiques* », Laboratoire ENS – Paris, Thèse septembre 2005
- [45] A. S. Tanenbaum, « *Modern Operating System* », 3ème Ed., Pearson Education 2009
- [46] J. R. Simard, « *Classical and quantum strategies for bit commitment schemes in the two-prover model* », School of Computer Science, McGill University Montréal, Québec February 2007
- [47] J. Talbot, D. Welsh, « *Complexity and cryptography : An introduction* », Cambridge University 2006
- [48] M. Bellach, « *Physique quantique* », CNRS 2ème Ed., Paris 2007
- [49] Y. Leroyer et G. Sénizergues, « *Introduction à l'information quantique* », 2014-2015
- [50] S. Weinberg, « *Lectures on Quantum Mechanics* », The University of Texas at Austin 2013
- [51] C. P. Williams, S.H Clearwater, « *Exploration in quantum computing* » Ed. Springer, New York 1998
- [52] G. Benenti, G. Casati, G. Strini, « *Principle of Quantum computation and information, volume 1 : basic concept* ».
- [53] N. D. Mermin, « *Quantum Computer Science : An introduction* », Cambridge 2007
- [54] P. P. Norkalaos, « *Quantum Information Theory and application to quantum cryptography* », University of Athens, 2001
- [55] P. Mladen, « *Quantum computation and Quantum communication : Theory and Experiment* », University of Zagreb, Springer 2006
- [56] A. Dorit, « *Quantum Computation* », The Hebrew University, Jerusalem, Israel, 15 Decembre 1998,

- [57] G. Cheng, K. B. Rane, « *Mathematics of Quantum Computation* », Chapman&Hall 2002
- [58] L. B. Samuel, « *Quantum Computing, where do we want to go tomorrow* », Wiley 1999
- [59] F. Faure, « *Notes de cours sur la mécanique Quantique* », Université Joseph Fourier Grenoble, 11 oct 2003
- [60] H. Weier, « *Experimental Quantum Cryptography* », PHysik-Departement
- [61] G. Gaëlle, « *Traitement quantique de l'information* », Institut de théorie des phénomènes physiques (ITP), Ecole Polytechnique Fédérale de Lausanne, 23 déc 2011
- [62] W. Steeb, « *Problems and solutions in Quantum Computing & Quantum Information* », World Scientific 2004
- [63] B. Félix, « *Intrication temporelle et communication Quantique* », Université Montréal, oct 2009
- [64] G. Messin, « *Source de lumière pour l'information quantique* », Université Paris IV, U.F.R D'Orsay, 10 juillet 2008
- [65] D. Paul, « *Hypothèses Calculatoires en cryptographie Quantiques* », Université Montréal, Thèse en Informatique, Juillet 2002
- [66] J. Buchmann, J. Ding, « *Post-Quantum cryptography* », Springer 2008
- [67] G. Tabia, « *Security analysis of a basis-independent scheme for quantum cryptography* », Science Department Of Physics, National University Of Singapore 2009
- [68] A. J. Menezes, P. C. Oorschot, S. A. Vanstone, « *Handbook of applied cryptography* », Massachusetts Institute of Technology June 1996
- [69] G. Umana, Valérie, Knudsen, L. Ramkilde, Leander, Gregor, « *Post-Quantum Cryptography* », Technical University of Denmark, 19 Jul 2016,
- [70] S. Heyes, « *Post Quantum cryptography : implementing alternative public key schemes on embedded devices, Preparing for the Rise of Quantum Computers* », German 2013
- [71] M. Candeau, « *Le cryptosystème NTRU* », Université Bordeaux 1, Master 2 Cryptologie et Sécurité Informatique, 14 mars 2011
- [72] T. Lepoint, « *Design and implementation of lattice-based cryptography* », Dissertation Defense held on 30/06/2014 in Paris, France
- [73] N. T. Courtois, « *La sécurité des primitives cryptographique basées sur des problèmes algébriques multivariés : MQ, IP, MinRank, HFE* », Université de Paris 6 - Pierre et Marie Curie, Thèse de doctorat Spécialité : Informatique, 25 sept 2001

- [74] A. Petzoldt, « *Selecting and Reducing Key Sizes for Multivariate Cryptography* », Université de Darmstadt, 15 Juillet 2013
- [75] D. Hankerson, A. Menezes, S. Vanstones, « *Guide to Elliptic Curve Cryptography* », Ed. Springer, New York, 2004
- [76] H. Dale, « *Elliptic Curves, Second Edition* », Ed. Springer, New York 2004
- [77] J. H. Silverman, J. T. Tate, « *Rational Points on Elliptic Curves* », Ed. Springer, New York 2015
- [78] D. Shumow, « *Isogenies of Elliptic Curves: A Computational Approach* », University of Washington 2009
- [79] A. R. Lozano, « *Elliptic Curves, Modular Forms and their L-functions* », Department of Mathematics, University of Connecticut, 2009
- [80] S. Lopp, V. K. Wooters, « *Protecting Information from Classical Error correction to Quantum cryptography* », Cambridge University 2006
- [81] T. Chou, « *Accelerating pre- and post-quantum cryptography* », Technische Universiteit Eindhoven, 27 juin 2016
- [82] J. Schreck, « *Signatures et authentifications pour les cryptosystèmes basés sur les codes correcteurs en métrique de Hamming et en métrique rang* » Thèse Université Limoge, 27 nov. 2013
- [83] M. Finiasz, « *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique* », Thèse INRIA, 01 oct. 2004
- [84] G. Murat, « *Résultants de polynômes de Ore et Cryptosystèmes de McEliece sur des Codes Rang faiblement structurés* », Thèse de Doctorat Université Limoge, 09 dec 2014
- [85] Lönghal, Carl, « *Some Notes on Code-Based Cryptography* », Lund University, 01 janv 2015
- [86] M. Rafael, « *Two Approaches for Achieving Efficient Code-Based Cryptosystems* », Thèse de Doctorat présentée à L'université Pierre et Marie Curie, 25 Nov. 2013
- [87] V. Myslivec, « *Asymetrický šifrovací algoritmus McEliece* », Université Páze, sept 2016
- [88] P. Frédéric, « *Sécurité algébrique et physique en Cryptographie fondée sur les codes correcteurs d'erreurs* », Thèse Université Pierre et Marie Curie, Paris-VI, 17 Av. 2015
- [89] Timový, « *McEliece s LDPC kodmi* », Slovenska Technická Univerzita v Bratislave, Bratislava 15 sept 2016
- [90] G. A. Malema, « *Low-Density Parity Check Codes : Construction and Implementation* », University of Adelaide, Australia Nov 2007

- [91] S. Lucile, « *Codes LDPC multi-binaires hybrides et méthodes de décodage itératif* », Université de Cergy-Pontoise, 03 oct 2008
- [92] G. Landais, « *Mise en œuvre de cryptosystèmes basés sur les codes correcteurs d'erreurs et de leurs cryptanalyses* », Thèse Université Pierre et Marie Curie, Paris-VI, 18 sept. 2014
- [93] J. Sarah, « *Low-Density Parity-Check Codes from Combinatorial Designs* », The Univesity of New Castle, Australia, Avr. 2004
- [94] N. Mobini, « *New Iterative Decoding Algorithms for Low-Density Parity-Check (LDPC) Codes* », Master of Applied Science in Electrical Engineering, Août 2011
- [95] A. Masoud, « *Efficient Analysis, Design and Decoding of Low-Density Parity-Check Codes* », Univesity of Toronto 2004
- [96] D. J. Bernstein, « *The Salsa20 family of stream ciphers* », The University of Illinois at Chicago
- [97] D. J. Bernstein, « *The Salsa20 Specification* », The University of Illinois at Chicago
- [98] D. J. Bernstein, « *The Salsa20 design* », The University of Illinois at Chicago
- [99] D. J. Bernstein, « *The Poly1305-AES message-authentication code* », The University of Illinois at Chicago
- [100] S. Vaarala, « *Poly1305-AES MAC* », Helsinki Univesity of Technology
- [101] D J. Barrett, R. Silverman, « *SSH, The Secure Shell : The Definitive Guide* », O'Reilly Janv. 2001
- [102] OpenSSL Software Foundation, « *OpenSSL cryptography and SSL/TLS Toolkit* », 09 Juin 2015
- [103] V. Suresh, « *FPGA implementation of low density parity check codes decoder* », University of North Texas, August 2009
- [104] G. K. Joakim, « *On iterative decoding of high-density parity-check codes using edge-local complementation* », the selmer center department of informatics university of bergen norway, August 2010

FICHE DE RENSEIGNEMENTS

Nom : RAKOTONDRAMANANA

Prénom : RADIARISAINANA Sitraka

Adresse : II G 10 Bis IBH Ambatomaro Mangarivotra

Tel : 0344636387

E-mail : radiarisainanasitraka@yahoo.fr, radiarisainanasitraka2@yahoo.fr

Titre du mémoire :



« Post-Quantum crypto spécifié par l'organisation NIST »

Nombre de pages : 109

Nombre de Tableaux : 4

Nombre de figures : 62

Directeur de Mémoire

Nom : RANDRIAMITANTSOA

Prénoms : Andry Auguste

Grade : Maître de Conférences

E-mail : andriau23@gmail.com

Tel : 0330744666

RESUME

Pour la technologie 5G Iot, il faut un algorithme de chiffrement résistant à l'attaque quantique qui porte le nom de H2020 (Horizon 2020). De ce fait, la cryptographie à base de factorisation des entiers en nombres premiers doivent être abandonné notamment RSA, El Gamal, Rabin, PGP. Ainsi, Des nouveaux problèmes calculatoires doivent être inventés ou reconsidérées surtout selon la proposition de NIST : Latticed based Cryptography, Multivariate Cryptography, Code Based Cryptography, Hash based Cryptography, SIDH. Le choix de cryptographie hybride du nom de PQP (Post-Quantum PGP) à base de correcteur d'erreur QC-MDPC utilisant McEliece et de PKDF2, Salsa20 et Poly1305 respecte la norme RFC 7539. Ce cryptosystème est généralement utilisé pour la cryptographie embarquée aux matériels.

Mots Clés : H2020, NIST, QC-MDPC, PQP, Iot

ABSTRACT

For the technology 5G Iot, We should find any cipher's algorithm resistant with the quantum attack which have the name H2020(Horizon 2020). In this fact, the ciphering with computational problem about the integer factorisation number is deprecated like RSA, El Gamal, Rabin, PGP. So, more computational problem should be invented or considered with respecting the NIST proposition like Latticed based Cryptography, Multivariate Cryptography, Code Based Cryptography, Hash based Cryptography, SIDH. The choice of hybrid cypher with the name of PQP (Post-Quantum PGP) which is a code based cryptography with QC-MDPC McEliece and PKDF2 with Salsa20 and Poly1305 respect the norm RFC 7539. This cryptosystem is generally used for the embedded ciphering

Key Words : H2020, NIST, QC-MDPC, PQP, Iot