Cryptographie : De RSA à l'algorithme de Shor

Simon MAGNIN-FEYSOT

Enseignant suiveur : Frédéric HOLWECK
Enseignant responsable de l'UV : Michel BRIAND

8 janvier 2014

Introduction

Je vais dans ce rapport m'intéresser au système de cryptographie le plus utilisé de nos jours, ainsi qu'à la cryptographie quantique, qui n'est encore qu'au stade de la recherche. J'ai choisi ce sujet car je suis passionnée d'informatique, surtout dans le domaine de la sécurité informatique. Ce rapport sera donc pour moi l'occasion d'enrichir mes connaissances sur ce sujet, mais aussi d'élargir ce domaine à travers la théorie de l'information quantique. Je suis content de pouvoir étudier ce sujet, car un ordinateur quantique d'après les spécialistes pourrait résoudre de très nombreux problèmes auxquels nous ne pouvons pas répondre actuellement avec les ordinateurs classiques, certains pensent pourvoir simuler la formation de l'univers depuis le big-bang avec un calculateur quantique de 300 qubits.

Table des matières

Chapitre 1

Définition et Historique de la cryptologie

1.1 Définition des termes

Le sens étymologique du mot cryptologie signifie "science du secret". On utilise les applications de cette science tous les jours sans s'en rendre compte. La cryptologie est divisée en deux domaines :

- La cryptographie : c'est la partie de la cryptologie où l'on va chiffrer un message dit clair (compréhensible de tous).
- La cryptanalyse : cette partie s'occupe de déchiffrer un message crypté.

Dans le chapitre 2, nous étudierons un des procédés de cryptographie le plus connu et utilisé à ce jour : le chiffrement RSA [1]. Le chapitre 5 quant à lui s'occupera de trouver une méthode de cryptanalyse efficace pour déchiffrer le RSA rapidement.
De plus il existe 2 types de chiffrement :

- le chiffrement symétrique
- le chiffement asymétrique

Le chiffrement symétrique Le chiffrement symétrique est le premier type de chiffrement à être apparu historiquement. Une même clé [2] est utlisée pour crypter le message et décrypter le message. L'inconvénient de cette technique est qu'il faut déjà communiquer la clé qu'on utlise à notre destinataire.

Le chiffrement asymétrique Le chiffrment asymétrique ou aussi appelé chiffrement à clé publique est apparut à la fin du XXième siècle. Cette méthode permet de passer outre le principal problème du chiffrage symétrique : On n'a pas besoin de communiquer la clé nécessaire au décryptage du message. Ce procédé empêche une tierce personne de pourvoir l'intercepter et lire ensuite toutes les communications. Par contre ce chiffrement est bien plus lent qu'un chiffrage symétrique.

1. Rivest, Shamir, et Adleman. Chaque lettre représente le nom des personnes l'ayant inventé.
2. Une clé est un nombre ou bien un mot qui sert à chiffrer ou déchiffrer le message, On effectuera une opération mathématique (l'opération dépend du chiffrement utilisé) entre le message et la clé.

Aujourd'hui en pratique, nous utilisons en général une combination des deux types de chiffrage (symétrique et asymétrique). On va se servir premièrement d'un chiffrage asymétrique (généralement RSA), ce chiffrage va nous permettre de pouvoir s'échanger une clé pour le chiffrage symétrique (en général AES : advanced encrytion standard) en toute séucrité, pour pouvoir ensuite profiter d'une plus grande vitesse d'echange. Exemple : La connexion par SSH (Secure SHell) entre deux ordinateurs.

Dans la suite de ce rapport, je vais vous expliquer différentes méthodes de chiffrement, pour cela je vais utiliser les conventions de la cryptologie : Alice et Bob désignent les personnes souhaitant communiquer de façon sécurisée ensemble.
Dans la suite de ce rapport tout les calculs se feront sur un ensemble de type : $\mathbb{Z}/n\mathbb{Z}$. Voici comment il est défini :

L'ensemble $\mathbb{Z}/n\mathbb{Z}$ C'est l'ensemble des entiers de 0 à $n-1$ par la congruence modulo n.

L'additon dans $\mathbb{Z}/n\mathbb{Z}$ on note les éléments de cet ensemble $\bar{x}$ si le reste de la division euclidienne par n est x.

$$\bar{x} + \bar{y} = x \bar{+} y \tag{1.1}$$

x et y admettent pour cette loi + un élément symétrique : opposé. L'élément neutre est 0

L'addtion dans $\mathbb{Z}/n\mathbb{Z}$ La multiplication est commutative associative et distributive par rapport à l'addition. L'élément neutre est $\bar{1}$. La multiplication est :

$$\bar{x} \times \bar{y} = x \bar{\times} y \tag{1.2}$$

Dans le prochain chapitre on travaillera sur l'ensemble $\mathbb{Z}/26\mathbb{Z}$ et sur le chapitre RSA sur l'ensemble $\mathbb{Z}/n\mathbb{Z}$ ou n représente un nombre très grand (produit de deux nombres de plusieurs centaines de chiffres)

1.2 Chiffrement de César

Voici maintenant une des plus vieilles méthodes de chiffrement connue qui est attribuée à César. C'est un chiffrage de type symétrique. Ce chiffrage très simple consiste à associer chaque lettre de l'alphabet à son rang (entre 0 et 25). Ensuite Alice et Bob se mettent en commun sur le choix d'une clé entre 1 et 25 (le choix de clé k = 0, n'a aucun intérêt). Alice n'a plus qu'à ajouter à chaque rang des lettres consistuant le message, la clé k :

$$C = M + k \ mod(n)$$

Bob pour déchiffrer a juste à appliquer l'opération inverse :

$$M = C - k \ mod(n)$$

Cette méthode de cryptograhie très simple n'est pas sécurisée. Si Charlie intercepte le message chiffré, il lui aisé de remonter au message original même en ne connaissant pas la clé k ayant servi au chiffrement. Car même sans utiliser l'aide d'un ordinateur il est envisageable de tester les 26 possibilités de chiffrement.
Exemple de cercle pour chiffrer et déchiffrer facilement le chiffrement de césar : (voir figure)

FIGURE 1.1 – cercle pour chiffrement de César

Exemple de chiffrement : Prenons le message suivant : "CRYPTOGRAPHIE" et on choisit la clé de chiffrement k = 3. On associe à chaque lettre son rang on a donc : "2 17 24 15 19 14 6 17 0 15 7 8 4" On ajoute 3 au rang de chaque lettre ce qui donne :"5 20 2 18 22 17 9 20 3 18 10 11 7". Il ne rest plus qu'a associer les lettres au rang nouvellement obtenu. Le message chiffré est donc : "FUBSWRJUDSKLH".
Le destinataire pour le décoder doit appliquer la même procédure mais en soustrayant la clé au lieu de l'ajouter.

1.3 Chiffrement de Vigenère

Le chiffrement de César n'apportant pas un degré de sécurité satisfaisant. Le français Blaise de Vigenère a introduit au milieu du XVIème siècle une méthode de chiffement avec un niveau de sécurité plus satisfaisant. Cette fois-ci une lettre n'est pas toujours remplacée par la même. Par exemple la lettre A dans le chiffrement de César qui dans un message était toujours chiffré par un T, sera par le chiffrement de Vigenère, chiffrée par exemple une fois par un T, une fois par un O, etc ... C'est ce qu'on appelle une substitution polyalphabétique.
C'est toujours un chiffrement symétrique, donc Alice et Bob doivent se mettre d'accord sur une clé. Cette fois-ci la clé ne sera pas un nombre entre 0 et 25, mais un mot.

Chiffrage Pour chiffrer le message à l'aide du mot clé, par exemple si nous avons le mot clé : MATHS. Notons U_k le décalage de la k^{ieme} lettre. Nous faisons donc un décalage de $U_1 = 12$ sur la première lettre, puis une décalage $U_2 = 0$ sur la deuxième lettre, $U_3 = 19$ sur la troisième lettre, un décalage de U_4 sur la quatrième lettre, un décalage $U_5 = 18$ sur la cinquième lettre. Et on recommence ainsi de suite jusqu'à la fin du message.

Déchiffrage Pour déchiffrer il suffit d'appliquer les calculs inverse. Les mêmes calculs que ceux utilisés pour déchiffrer le code de César, mais en changeant les clés à chaque fois. Pour simplifier la tâche, on utilise, la table de Vigenère pour coder et décoder plus vite.(voir figure)

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

FIGURE 1.2 – Table de Vigenere

Pour le chiffrement on utilise cette table la façon suivante :

- Sur la première ligne on cherche la lettre du message clair à chiffrer, nous avonc donc la colonne.
- Ensuite on cherche sur la première colonne la lettre du mot clé que l'on veut, cela nous donne la ligne.
- Pour trouver le lettre correspondante chiffrée il ne reste plus qu'à aller, à l'intersection de la ligne et colonne trouvée.

Pour décrypter il faut trouver la lettre chiffrer dans la ligne correspondant à la lettre du mot clé. On remonte sur la première ligne pour trouver le message clair.

Attaque Pour attaquer ce chiffrement, il faut avoir un message chiffré assez long. Ensuite on essaie de repérer des séquences d'au moins 3 lettres qui se répétent. Il y a de fortes chances pour que ce triplet (au minimum) soit une séquence chiffrée de la même façon. Le nombre de lettres séparant les deux séquences se répétant est donc un multiple du nombre de lettres du mot-clé. En trouvant plusieurs fois cette même répétition (plus que deux fois), on calcule le pgcd et on trouvera la longueur du mot-clé.
Maintenant on sépare le message en n listes (n étant le nombre de lettre du mot-clé). La

première liste contenant toutes les lettres du message suivant cette suite : U_k = U× k + 1, une deuxième liste contenant toutes les lettres du message suivant cette suite : U_k = U× k + 2. Ainsi de suite jusqu'à la n^{ieme} liste.
Pour toutes les listes on regarde quelle lettre se répète le plus souvent. Cette lettre chiffrée correspond au E clair. Cette déduction est faite d'après les statistiques d'apparition des lettres qui donnent la lettre E en première position d apparition (en génral largement en tête) dans quasiment toutes les langues. Il nous reste donc plus qu'à l'aide de la table de Vigenère de trouver les lettres du mot-clé, et déchiffrer le message intercepté.

Exemple de chiffrement de Vigenère Prenons le message suivant :"Ceci est un message" et prenons la clé secrète : MATHS. On ne s'occupe pas des espaces. Le C sera donc chiffré à l'aide du M, le E à l'aide du A, le deuxième C à l'aide du T, le I à l'aide du H, le E à l'aide du S, et on recommence ainsi de suite jusqu'à la fin du message.
En nous aidant de la table de Vigenère, le C chiffré à l'aide du M donne : O, le E chiffré avec le A donne E ... Ce qui donne finalement le message chiffré : OEVPWFTNUEQSLHYQ

Chapitre 2

Cryptographie actuelle : RSA

Le chiffrement RSA porte les initiales de ses trois inventeurs : Ron Rivest, Adi Shamir, et Leonard Adleman. Cet algorithme a été décrit en 1977. Cet algorithme qui utilise des théorèmes mathématiques assez simples, trouve sa force dans la difficulté à factoriser un grand nombre entier (actuellement environ 200 chiffres) qui est le produit de deux entiers premiers.

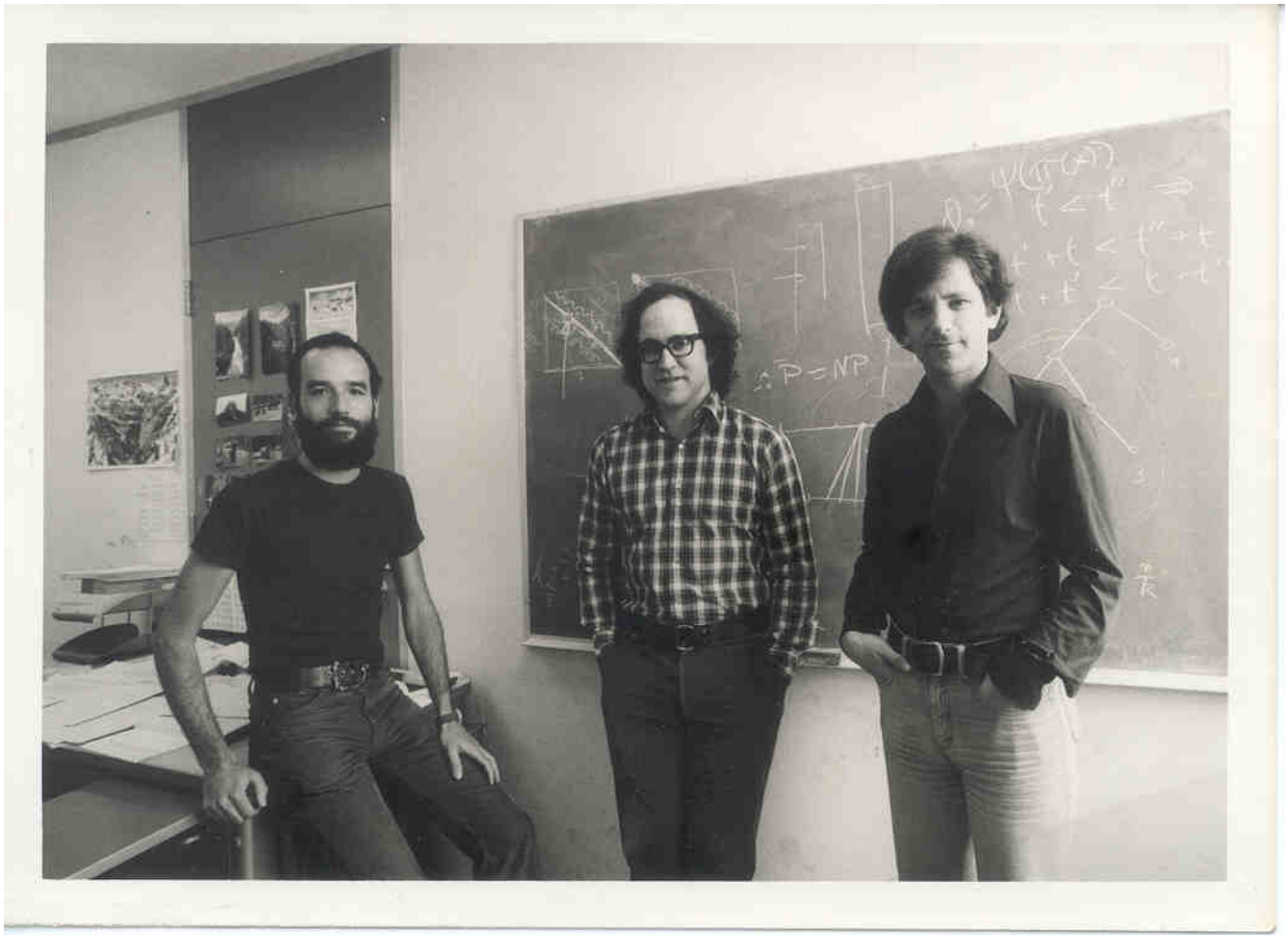

FIGURE 2.1 – Rivest, Shamir, et Adleman

2.1 Création des clés

La première étape pour que Alice puisse communiquer avec Bob, c'est de créer les deux jeux de clés (publique qui sera diffusée, et le jeu privé qui sera gardé par Alice pour décrypter les messages que Bob aura envoyés).

2.1.1 Le jeu de clé publique

Pour créer le jeu de clé publique, Alice va choisir 2 grands nombres premiers distincts (actuellement, il est conseillé d'utiliser des nombres premiers ayant au moins 200 chiffres) Pour cela, on pourrait utiliser les nombres de Mersennes. Ce sont des nombres qui sont très souvent premiers, de la forme :

$$2^{n-1}$$

Mais il est préférable de n'avoir aucune information sur le nombre choisi. On va donc choisir un nombre au hasard et testé sa primalité. Pour tester la primalité d'un nombre nous avons plusieurs tests (Rabin-Miller, Solovay-Strassen, AKS). Ces tests sont détaillés dans le Chapitre 3. Le fait d'avoir des nombres particuliers facilite leur factorisation.
Une fois que nous avons nos deux entiers p et q nous les multiplions ensemble, cela nous donne :

$$n = p \times q$$

Nous avons maintenant la première partie de la clé de chiffrement (clé publique). La deuxième partie de la clé de chiffrment est e : l'exposant de chiffrement. Pour calculer e nous faisons appele à la fonction indicatrice d'Euler qui dans le cas d'un nombre, donnée n étant le produit de deux nombres premiers, est égale à :

$$\varphi(n) = (q-1)(p-1)$$

Pour avoir e, il suffit de choisir un nombre inférieur à ϕ et premier avec ϕ.

Alice à maintenant la clé de chiffrement (n, e) en sa possession, elle peut l'envoyer à Bob pour qu'il puisse lui transmettre des messages chiffrés en toute sécurité.

2.1.2 Le jeu de clé privé

Maintenant que Bob a la clé publique, il faut qu'Alice ait sa clé privée pour pouvoir être la seule à déchiffrer les messages que Bob lui enverra.
La clé privé se constitue de :

- p
- q
- d, l'exposant de déchiffrement

p et q, sont les grands entiers premiers qui ont servi à définir n.

Le calcul de d Il faut calculer d, de façon à satisfaire l'équation suivante :

$$e \times d \; mod(\varphi(n)) = 1$$

Nous pouvons donc écrire cette équation comme ceci, avec $k \in$ appartenant à $\mathbb{Z}$:

$$e \times d = 1 + k\varphi(n)$$

Cette équation a toujours une solution d'après le au théorème de Bezout.

Théorème de Bezout Soient a, $b \in \mathbb{N}$, $\exists$ u, v $\in \mathbb{Z}$ tels que :

$$au + bv = pgcd(a, b)$$

On a e premier avec $\varphi(n)$ donc pgcd(e,$\varphi(n)$) = 1. Le théorème de Bézout s'applique bien. Maintenant il est aisé de trouver les coefficients à l'aide de l'algorithme d'Euclide.

Algorithme d'Euclide Avec a, b les mêmes entiers que pour le théorème de Bezout et q_n et r_n sont les quotients et restes des divisions euclidiennes successives :

$$\begin{aligned} a &= b.q_1 + r_1 \\ b &= r_1.q_2 + r_2 \\ r_1 &= r_2.q_3 + r_3 \\ &... \\ r_k &= r_{k+1}.q_{k+2} + r_{k+2} \\ r_{k+1} &= r_{k+2}.q_{k+3} + 0 \end{aligned}$$

r_{k+2} est le dernier reste non nul, c'est donc le PGCD de a et b. Pour trouver les coefficients u et v, on isole le PGCD :

$$r_{k+2} = r_k - r_{k+1}.q_{k+2}$$

On remplace r_{k+1} de la même façon en remontant jusqu'a trouver, une équation de la forme :

$$au + bv = r_{k+2}$$

Si $a = e$ et $b = \phi(n)$ alors $r_{k+2} = 1$ et les entrées uu et v déterminées par l'algorithme permettent de trouver d.

Alice et Bob ont maintenant toutes les clés pour communiquer ensemble de façon sécuriser. Regardons maintenant comment les messages sont chiffrer et déchiffrer avec les clés que nous venons de calculer.

2.2 Chiffrage/Déchiffrage

2.2.1 Le chiffrage

Pour chiffrer le message il faut transformer son nombre (souvent en informatique on choisira le code ASCII des caractères) puis découper ensuite son message en bloc de taille juste inférieur à n. Ensuite le chiffrement se fait de cette façon :

$$C = B^e mod(n)$$

Si on a fait l'erreur de découper le message en bloc de taille plus grand que n, on voit que comme les calculs se font mod(n), on perd des informations.
Il ne reste plus qu'à Bob à envoyer C, les blocs de messages chiffrés à Alice.

2.2.2 Le Déchiffrage

Pour déchiffrer le message C qu'Alice vient de recevoir. Il lui suffit d'appliquer le calcul inverse à l'aide de d, son exposant de déchiffrement :

$$B = C^d mod(n)$$

2.2.3 Vérification du message

Ce paragraphe sert à démontrer mathématiquement que le message B de départ (celui de Bob), sera bien le même que celui qu'Alice retrouvera après déchiffrement. Pour celà nous allons utiliser le petit théorème de Fermat

Petit théorème de Fermat (voir D) Si p est premier et $a \in \mathbb{Z}/\mathrm{p}\mathbb{Z}$ alors,

$$a^p = a[p]$$

et voici son corrolaire :

$$a^{p-1} = 1[p]$$

Pour en revenir sur notre message chiffré, nous avons les équations suivantes :
Alice reçoit C et calcule :

$$C^d = B^{ed}\ mod(n)$$

D'après la définition de d

$$C^d = B^{(p-1)(q-1)+1}\ mod(n)$$

Calculons C^d dans $\mathbb{Z}/\mathrm{p}\mathbb{Z}$

$$\begin{aligned} C^d &= B^{(p-1)(q-1)+1}\ mod(p) \\ &= B\ mod(p) \text{ D'après fermat} \end{aligned}$$

De même dans $\mathbb{Z}/\mathrm{q}\mathbb{Z}$

$$\begin{aligned} C^d &= B^{(p-1)(q-1)+1}\ mod(q) \\ &= B\ mod(p) \text{ D'après fermat} \end{aligned}$$

Il nous faut alors utliser le lemme de Gauss :

Lemme de Gauss : (voir D) Si $a|c$ et $b|c$ et a et b sont premiers entre eux alors $ab|c$.

Puisque :

$$\begin{aligned} C^d - B = 0\ [p],\ p|(C^d - B) \\ C^d - B = 0\ [p],\ p|(C^d - B) \end{aligned}$$

Donc par le lemme de gauss $C^d - B = 0\ [pq]$, c'est à dire $C^d = B\ [n]$
Alice retrouve bien le message B en calculant $C^d\ mod(n)$

2.2.4 Exemple de chiffrement par RSA

Prenons les 2 nombres premiers $p = 11$ et et $q = 13$. Ce qui nous donne $n = 11 \times 13 = 143$. On calcule $\varphi(n) = (p-1)(q-1) = 120$. On choisit un nombre premier à $\varphi(n)$, prenons e = 13. On cherche d tel que :

$$ed \equiv 1[\varphi(n)] \tag{2.1}$$
$$\iff ed - k\varphi(n) = 1 \tag{2.2}$$
$$\iff 13d - 120k = 1 \tag{2.3}$$

On utilise l'algorithme d'Euclide :

$$120 = 13 \times 9 + 3 \tag{2.4}$$
$$13 = 3 \times 4 + 1 \tag{2.5}$$
$$3 = 3 \times 1 + 0 \tag{2.6}$$

On remonte pour trouver les coefficients de Bezout :

$$1 = 13 - 4 \times 3 \quad = 13 - 4 \times (120 - 13 \times 9) = 13 - 4 \times 120 + 36 \times 13 \quad = 13 \times 37 - 4 \times 120 \tag{2.7}$$

Par identification on voit que d = 37. Nous avons maintenant toutes les clés. Le message que Bob veut passer à Alice est : 20. Pas besoin de séparer en blocs car 20 comporte un chiffre de moins que n. On chiffre donc :

$$20^{13} \equiv 124[143] \tag{2.8}$$

Alice reçoit le message 124, il le décrypte :

$$124^{37} \equiv 20[143] \tag{2.9}$$

Alice reçoit bien le message 20.

Chapitre 3

Algorithme pour RSA

Dans cette partie nous allons nous occuper de comparer quelques algorithmes de tests de primalité en termes de complexité algorithmique. Nous comparons ces tests avec un algorithme de factorisation, pour montrer que la factorisation (décryptage) est bien plus longue que création des deux nombres premiers (création de clés). L'algorithme de factorisation de Sohr sera étudié dans le chapitre 5.

3.1 Complexité d'un algorithme

La complexité temporelle d'un algorithme évalue le temps d'éxecution de celui-ci en fonction de la taille des entrées.
Dans ce chapitre elle sera représentée par une fonction de n où n est la taille (souvent en binaire) des valeurs d'entrées. On prendra les conventions suivantes :

- Pour une addition de deux nombres à un chiffre, on pose une complexité de 1.
- Pour une addition de deux entiers à n chiffres, la complexité sera donc de n. Car cela revient à n addtions de nombres à un seul chiffre.
- Une multiplication quant à elle, a une complexité de n^2. Chaque chiffre de chaque nombre doit se multiplier à chaque chiffres de l'autre nombre.

3.2 Test de primalité

Les tests que nous allons voir ici sont des tests probabilistes, c'est-à-dire qu'ils nous donnent seulement une très forte probabilité que le nombre n, étudié soit premier. Les probabilités que le nombre n, soit déclaré premier par le test mais ne le soit pas réellement (aussi appelé pseudopremier) sont très faibles.
Il existe des tests déterministes (par exemple : le test AKS), ils ne seront cependant pas étudiés ici.

3.2.1 Test de fermat

Le test de primalité de Fermat comme son nom l'indique est un test probabilistique se basant sur le théorème de Fermat :$a^{p-1} \equiv 1[p]$ avec p un nombre premier et a un entier premier avec p. On va donc partir de cette équation vrai pour les nombres premiers, et la vérifier pour le nombre n dont on veut tester la primalité. La réciproque du petit théorème de Fermat n'est pas vraie mais en testant pour plusieurs valeurs de a, nous aurons une très

bonne probabilité que n soit premier. Voici donc l'algorithme de ce test de primalité. Dans ce test 4 valeurs de a suffisent, on prend les quatres premiers nombres premiers : 2, 3, 5, 7.

Data: Entier n
Result: booléen (1 = premier ou pseudopremier, 0 = composé
begin
  for $i = 2, 3, 5, 7$ **do**
    if $i^{n-1} \not\equiv 1[n]$ **then**
      return 0
    end
  end
  return 1
end

Algorithm 1: Algorithme de test de primalité de Fermat

La complexité de cette algorithme revient à la complexité de l'algorithme de l'exponentiation rapide multiplié par le nombre de valeurs que prend i (ici 4).

Exponentiation rapide L'exponentiation rapide consiste à décomposer la puissance à calculer en puissance de 2, (exemple : $a^{89} = a^1 * a^8 * a^{16} * a^{64}$). Or chaque puisance de 2 est le carré de la précédente, cet algorithme est donc plus rapide que de faire 88 multiplications. Le nombre total de multiplications est égal au nombre de chiffres de l'exposant en numération binaire, plus le nombre de chiffres (de l'exposant en numération binaire) qui sont égaux à 1. La complexité est donc de $log(2^n) = n \times log(2)$.
La complexité du test de fermat est donc de i $\times n \times log(2)$. ce qui est une complexité linéaire.
Nous avons ici choisit 4 valeurs pour i, cela suffit amplement dans la pratique (environ 0.0007% de chance de tombe sur un pseudopremier). Il existe des nombres appelés : Nombre de Carmichael qui sont pseudopremiers pour toutes les bases sans être premiers. Même si ils ne représentent qu'une petite proportion, il y en a une infinité.

3.2.2 Test de Solovay-Strassen

Je vais présenter maintenant un autre test de primalité celui de Solovay-Strassen. Ce test est aussi probabiliste. Il utlise le même principe que le test de primalité de Fermat. Il se base sur une affirmation vrai pour les nombres premiers : $a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}$, avec p premier, a entier. $\left(\frac{a}{p}\right)$ est le symbole de Legendre qui vaut :

- 0 si a est divisible par p.
- 1 si a est un résidu quadratique de p ($\exists k$ tel que : $a \equiv k^2 \pmod{p}$)
- -1 Si a n'est pas divisible par p

On va vérifier k fois cette équation pour des valeurs aléatoires de a $\in$ [2, n-1]. k est choisi par l'utilisateur, ce sera la précision. Cet algorithme a une probabilité de $1/4^k$ de donner un nombre pseudopremiers.

Cet algorithme a une complexité de O(k$\times\log^3$(p))

Data: n entier à tester, k entier : la précision
Result: boolean : 1 pseudopremier, 0 composé
begin
  for *i from 1 to k* **do**
    choisir un a aléatoirement entre [2, n-1]
    x $\longleftarrow \left(\frac{a}{p}\right)$
    if $x = 0$ *or* $a^{(n-1)/2} \not\equiv x$ **then**
      return 0
    end
  end
  return 1
end

Algorithm 2: Algorithme de test de primalité de Solovay-Strassen

3.3 Factorisation des entiers

Maintenant que nous avons vus que les tests de primalité sont de complexité polynomiale, regardons maintenant le test de factorisation basique dit naïf qui lui est de complexité exponentielle, ce qui rend n le produit des nombres premiers infactorisable dans un temps raisonnable.
Cet algorithme va se contenter de tester tous les nombres premiers qui peuvent diviser n, en excluant ensuite tous les nombres pairs si 2 ne divise pas n. On part de $\sqrt{n}$ car il y a forcement un facteur plus petit et un autre plus grand que $\sqrt{n}$ et ils sont en général relativement rapprochés.

Cet algorithme très simple de la factorisation de facteurs premiers a une complexité de type exponenetielle. Cependant même les algorithmes plus perfectionnés n'arrive pas à faire mieux qu'une complexité sous-exponentielle, ce qui ne permet toujours pas de factoriser n dans un temps correct.

Compléxité On considère le nombre N dont on test la primalité sous la forme binaire : 2^n. On divise N/2 fois le nombre 2^n. La complexité est donc de $N \times 2^{2(n-1)}$. C'est à dire une complexité exponentielle.

```
Data: Entier n
Result: rien
begin
    if n%2 = 0 then
        Output: 2
        Output: n/2
    end
    for i from √n(arrondi à l'entier au supérieur) to n do
        if i%n = 0 then
            Output: n/i
            Output: i
        end
        else
            if i % 2 = 0 then
                i ⟵ i + 1
            end
            else
                i ⟵ i + 2
            end
        end
    end
end
```

Algorithm 3: algorithme de factorisation de nombres entiers en facteurs premiers

Chapitre 4

Introduction à la théorie de l'information quantique

Ce chapitre est une introduction à la théorie de l'informatique quantique où je vais présenter les notions fondamentales nécessaires pour comprendre l'algorithme de Shor que nous verrons dans le Chapitre 5.
La théorie de l'information quantique a pour objet l'étude de protocoles de communication et de calcul basés sur les propriétés de la mécanique quantique. Comme nous le verrons cette théorie peut être définie à partir d'axiomes issus de cette physique sous atomiques.

4.1 Les Qubits

4.1.1 Définition d'un quantum bit (qubit)

Un qubit est l'analogue quantique du bit en informatique classique. C'est l'état qui représente la plus petite unité de stockage en informatique quantique.
Comme pour un bit il existe deux états de base : $|0\rangle$ et $|1\rangle$. La notation utilisée est la notation de Dirac, elle permet de réprésenter les qubits en tant que vecteurs et également de représenter le produit hermitien à l'aide des bra-ket [1].
La théorie de l'information quantique est définie sur trois axiomes :

- Axiome 1 : un état est une superposition d'états de base. $|\psi\rangle$ s'écrit donc : $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ avec α et $\beta \in \mathbb{C}$, et $|0\rangle$, $|1\rangle$ formant une base orthonormée
- Axiome 2 : Mesurer l'état d'un qubit $|\psi\rangle$ [2] revient à le projeter dans une base orthogonale $(|\psi_1\rangle, |\psi_2\rangle)$ d'états possibles. Après la mesure l'état $|\psi\rangle$ sera dans l'état $|\psi_1\rangle$ ou dans l'état $|\psi_2\rangle$. La probabilité d'observer l'état $|\psi_i\rangle$ est donnée par $|\langle\psi_i|\psi\rangle|^2$
- Axiome 3 : Un état $|\psi\rangle$ évolue selon une transformation unitaire. C'est-à-dire, on passe de l'état $|\psi_1\rangle$ à $|\psi_2\rangle$ en multipliant le vecteur $|\psi_1\rangle$ par une matrice unitaire.

  La sphère de Bloch Une façon de représenter un qubit est la sphère de Bloch. Chaque point de la sphère représente un état possible d'un qubit. Pour voir un qubit

1. Le produit hermitien des vecteurs $|\rho\rangle$ et $|\psi\rangle$ sera noté $\langle\rho|\psi\rangle$
2. $|\psi\rangle$ est le vecteur qui représente l'état du qubit

à l'aide de la sphère de Bloch, il faut modifier l'écriture de notre qubit. Nous avons :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$
$$\iff |\psi\rangle = e^{i\gamma}(cos(\frac{\theta}{2})|0\rangle + e^{i\phi}sin(\frac{\theta}{2})|1\rangle)$$

Où $e^{i\gamma}$ est un nombre complexe de module 1. Donc sa présence (ou son absence) ne modifie pas les probabilités d'observer $|0\rangle$ ou $|1\rangle$.

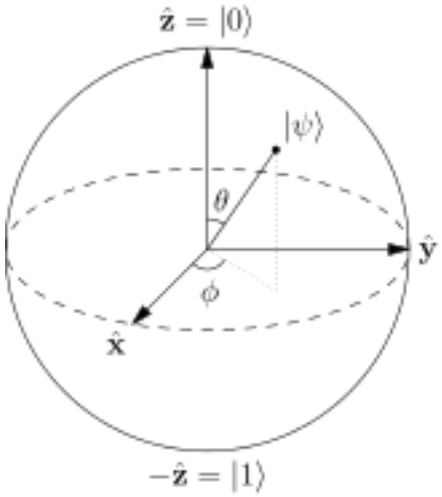

FIGURE 4.1 – Sphere de Bloch

Remarque sur la réalisation physique d'un Qubit Un qubit peut être réalisé physiquement de différentes manières :

- Orbital atomique : On prend une électron autour d'un noyau, et son niveau d'énergie correspond à son état. $|0>$ correspond à l'état de base non excité et $|1\rangle$ correspond à l'état excité
- Spin : l'état $|0\rangle$ correspond à un spin up et l'état $|1\rangle$ à un spin down.

4.1.2 Espace à plusieurs Qubits (Espace de Hilbert)

Supposons maintenant que nous avons deux qubits. Nous avons donc maintenant 4 états de bases : $|00\rangle, |01\rangle, |10\rangle$, et $|11\rangle$. Le vecteur qui décrit l'état des 2 qubits peut être vu comme une superposition de ces 4 états et s'écrit :

$$|\psi\rangle = a_1|00\rangle + a_2|01\rangle + a_3|10\rangle + a_4|11\rangle$$

Comme pour un espace à un seul qubit $|a_n|^2$ représente la probabilité que ce soit l'état ayant le coefficient a_n qui soit lu lors de la mesure.
De même nous avons toujours : $\sum_{n=0}^{4} |a_n|^2 = 1$. Cette construstion se généralise à n qubits. L'espace des états possibles et appelé espace de Hilbert.
Pour un espace à n qubit nous avons 2^n états de bases possibles notés $|i_{n-1}...i_0\rangle$ avec $i_j \in \{0, 1\}$. On utilise aussi la notation "décimale" : $|i_{n-1}...i_0\rangle = |k\rangle$ avec $k = i_{n-1}2^{n-1} + ... + i_O 2^0$. Ainsi l'état d'un système à N qubits sera noté $|\psi\rangle = \sum_{k=0}^{N-1} a_k|k\rangle$ avec $N = 2^n$. Avec toujours la convention : $\sum_{n=0}^{n} |a_n|^2 = 1$.

4.2 Portes logiques et circuit quantique

Il existe aussi les analogues des portes logiques dans le monde quantique : les portes logiques quantiques. Ces portes sont plus nombreuses que les portes logiques classiques. Toutes les portes à un qubit sont représentées par des matrices 2×2 et plus généralement les portes à n qubits par des matrices de taille $2^n \times 2^n$. D'après l'axiome 3, pour qu'une matrice U soit une porte, elle doit vérifier cette équation (voir D) :

$$U^\dagger U = I \tag{4.1}$$

C'est à dire des matrices dites unitaires. Il y a une infinité de portes logiques quantiques. Voici une étude des portes les plus utilisées :

Portes de "rotations" Ce sont des portes à un qubit qui portent les noms : X, Y, Z. Tout simplement car si on représente ces transformations sur la sphère de Bloch, chacune de ces matrices correspond à une rotation autour d'un des axes :

- X = $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ Cette porte correspond à la porte NON, elle inverse les coefficient α et β du qubit en entrée. En représentation sur la sphère de Bloch, elle correspond à une rotation de $\pi/2$ sur l'axe X.
- Y = $\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ Cette porte correspond à une rotation de $\pi/2$ sur l'axe Y. Comme la porte X, elle inverse α et β mais en plus elle change le premiers coefficient en sortie.
- Z = $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ Cette porte correspond à la rotation de $\pi/2$ sur l'axe Z. Cette porte laisse inchangée le premier coefficient de l'entrée mais change le signe du second.

Porte d'Hadamard Cette porte est très utilisée lors de calcul quantique. Elle s'écrit :

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Porte S Cette porte est définie selon la matrice $\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. Elle n'est pas souvent utilisée et n'a pas de propriétés particulières néanmoins, elle sera utilisé dans le Chapitre 5.

Porte à multiple qubit Les portes à plusieurs qubits se forme de la même façon que les portes à un seul qubit, par contre elles sont de taille $2^n \times 2^n$. Les mêmes portes que les classiques existent :

- OR (OU)
- XOR (OU exclusif)
- AND (ET)
- NAND (NON-ET)
- NOR (NON-OU)

Et donc toute porte classique est réalisable dans une version quantique [3].

3. Les portes classiques ne sont en général pas inversibles par contre les portes quantiques, qui correspondent à des matrices unitaires, le sont. La reproduction de portes classiques à l'aide de portes quantiques nécessite d'introduire des scratch qubits qui seront effacés à la fin du calcul.

Théorème Toute matrice U_n de taille $2^n \times 2^n$ est réalisable à partir de portes quantique à un qubit et de trois portes quantique multiqubits (CNOT, SWAP, TOF).D

CNOT Voici la porte appelé CNOT (control NOT), qui a l'avantage de garder en mémoire la première entrée.

$$U_{CN} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Cette porte peut être définie de la façon suivante si le qubit de contrôle est à 1, le qubit cible change de valeur :

$$\begin{aligned} |00\rangle &\to |00\rangle \\ |01\rangle &\to |01\rangle \\ |10\rangle &\to |11\rangle \\ |11\rangle &\to |10\rangle \end{aligned}$$

SWAP La porte SWAP (échange) comme son nom l'indique va échanger les coefficients entre les qubits entrés. Les qubits : $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ et $|\psi_2\rangle = \alpha'|0\rangle + \beta'|1\rangle$ donneront en sortie : $|\psi_1\rangle = \alpha'|0\rangle + \beta'|1\rangle$ et $|\psi_2\rangle = \alpha|0\rangle + \beta|1\rangle$. Voici son schéma :

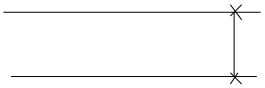

FIGURE 4.2 – Schéma de la porte SWAP

TOF La porte TOF (abréviation de Toffoli nom de son inventeur) est une porte à 3 qubits aussi appelée CCNOT (Control Control NOT) sa matrice est la suivante :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.2}$$

Cette porte est indispensable pour toutes les opérations booléennes. Sur le mêmeprincipe que la porte CNOT, si un des deux qubits de contrôle est égale à 0 rien ne change mais si les deux qubits sont égale à 1, le troisième change de valeur.

4.3 Intrication et téléportation

Cette partie n'est pas nécessaire pour comprendre l'algorithme de Shor. C'est un approfondissement de l'étude de l'informatique quantique et de ses possibilités.

4.3.1 Etat intriqué

L'intrication est un phénomène qui se produit lorsque l'on a un système à plusieurs qubits. Nous prendrons ici un système à deux qubits pour faire simple. On se place dans un espace de Hilbert. Il y a deux possibilités :

- Soit on se retrouve dans un cas comme celui-ci :
  $|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$ on peut factoriser cet état en :
  $|\psi\rangle = (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)$.
  On a factorisé cet état de deux qubits en 2 états d'un qubit. Chaque qubit est ici indépendant l'un de l'autre. Si on mesure l'état d'un des qubit, il n'y aura pas d'influence sur la mesure du deuxième qubit.
- Soit on se retrouve dans une cas comme celui-ci [4] : $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Ici il est impossible de factoriser l'état en deux états en un qubit. C'est ce que l'on appelle un état intriqué. Si l'on mesure l'un des deux qubit, la mesure influencera aussi le deuxième qubit. Par exemple si l'on mesure le premier qubit et que l'on trouve $|0\rangle$, $|\psi\rangle$ sera automatiquement à l'état $|00\rangle$ et donc la deuxième particule à $|0\rangle$.

4.3.2 Téléportation

Cette section s'intéresse à une application intéressante du phénomène d'intrication des qubits. Nous avons deux personnes (Alice et Bob) qui souhaitent communiquer, elles possèdent chacune un qubit d'une paire intriquée. Alice souhaite envoyer à Bob le message codé par un troisième qubit. Nous avons donc un système à 3 qubits :

- le qubit contenant le message, noté A
- le premier qubit de la paire intriquée détenu par Alice, noté B
- le deuxième qubit de la paire intriquée détenu par Bob, noté C

Notre état à trois qubits de départ est défini comme cela [5] :

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}\{(\alpha|0\rangle + \beta|1\rangle)(|00\rangle + |11\rangle)\}$$

La première parenthèse correspond à l'état du qubit inconnu et la deuxième parenthèse correspond au deux qubit de la paire intriquée.
Pour envoyer l'information Anne va d'abord réaliser une porte CNOT sur la paire AB, l'état va devenir :

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\{\alpha(|000\rangle + |011\rangle) + \beta(|110\rangle + |101\rangle)\}$$

Elle envoie ensuite le premier qubit sur une porte Hadamard, l'état devient :

$$|\psi_2\rangle = \frac{1}{2}\{\alpha(|000\rangle + |100\rangle + |011\rangle + |111\rangle) + \beta(|010\rangle - |110\rangle + |001\rangle - |101\rangle)\}$$

4. Les états intriqués ont d'abord été imaginés dans une expérience de pensée par Einstein, Podolsky et Rosen. Le but de cette expérience était d'aboutir à un paradoxe (paradoxe EPR) qui contredirait les fondements de la mécanique quantique. En 1981 une preuve expérimentale (Expérience d'Aspect) a donnéeraison à la mécanique quantique. pour plus d'informations : `http://fr.wikipedia.org/wiki/Paradoxe_EPR#Le_paradoxe`

5. Dans ce calcul comme dans les prochains où je m'aiderai de schéma, ϕ_0 correspond à l'état initial, ϕ_1 à l'état de système après le passage dans la première porte, ..., jusqu'à l'état final.

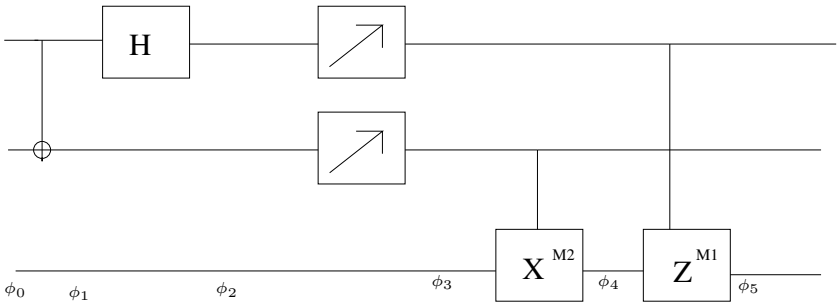

FIGURE 4.3 – Circuit de la téléportation quantique

$$\begin{aligned}|\psi_2\rangle = \frac{1}{2}\{&|00\rangle(\alpha|0\rangle + \beta|1\rangle)\\ &|01\rangle(\alpha|1\rangle + \beta|0\rangle)\\ &|10\rangle(\alpha|0\rangle - \beta|1\rangle)\\ &|11\rangle(\alpha|1\rangle - \beta|0\rangle)\}\end{aligned}$$

Nous voyons maintenant que l'état du qubit C, est totalement déterminé par l'état de la paire AB. Alice va donc maintenant effectuer une mesure $|\alpha'\beta'\rangle$ et l'envoyer à Bob. le qubit se retrouve donc dans un des quatres états du dessus. Il suffit de lui appliquer les portes $Z^{\alpha'}X^{\beta'}$. c'est à dire :

- Rien si $|\alpha'\beta'\rangle = |00\rangle$
- Une porte X si $|\alpha'\beta'\rangle = |01\rangle$
- Une porte Z si $|\alpha'\beta'\rangle = |10\rangle$
- Une porte ZX si $|\alpha'\beta'\rangle = |11\rangle$

Cela implique donc que l'information voyage plus rapidement que la vitesse de la lumière (ce qui est physiquement impossible) ? En fait il ne faut pas voir les qubits comme des entitées séparées mais voir la paire comme une seule entité. Ce qui explique pourquoi des que l'on mesure une partie de cette entité, l'autre partie se retrouve directement affectée. Une expérience de téléportation utilisant ce principe a été réalisée à travers le Danube (voir article de Nature du 19 Août 2004[6]).

Chapitre 5

L'algorithme de Sohr

Dans ce dernier chapitre nous étudions l'algorithme de Shor. Cet algorithme qui porte le nom de son inventeur, Peter W. Shor professeur de mathématiques au MIT, a été découvert en 1994.

FIGURE 5.1 – Peter Shor

5.1 Transformée de Fourier quantique

L'algorithme de Shor repose essentiellement sur la transformée de Fourier quantique (abrégé TFQ par la suite).

5.1.1 TFQ_8

Pour commencer à étudier la TFQ, considérons un espace à 3 qubits. Nous avons donc N états de bases possibles $N = 2^3$

La porte $\mathbf{R}_n$ C'est une porte à un qubit. Nous l'étudions seulement maintenant car sa seule utilisation est pour la TFQ. Cette porte est définit comme cela :

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

On peut voir que $R_0 = I$, $R_1 = Z$ et $R_2 = S$.
Pour en revenir à notre TFQ à trois qubits. Le cricuit qui définit cette TFQ est :

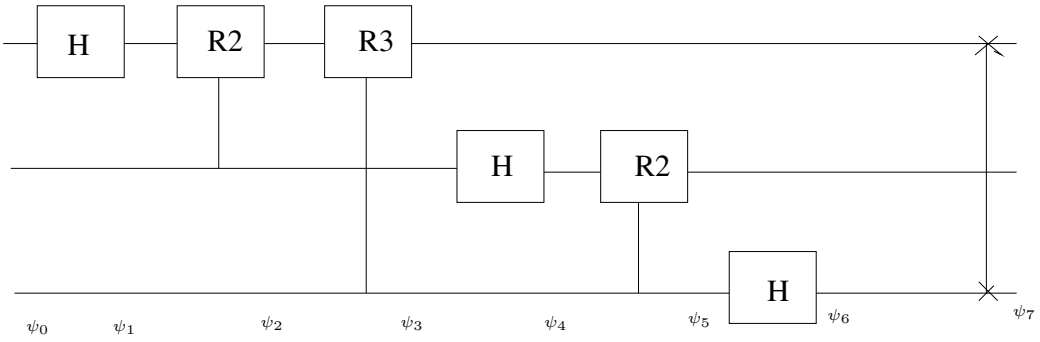

FIGURE 5.2 – Circuit de transformé de qubit à 3 qubits

D'après la remarque plus haut le circuit équivalent est :

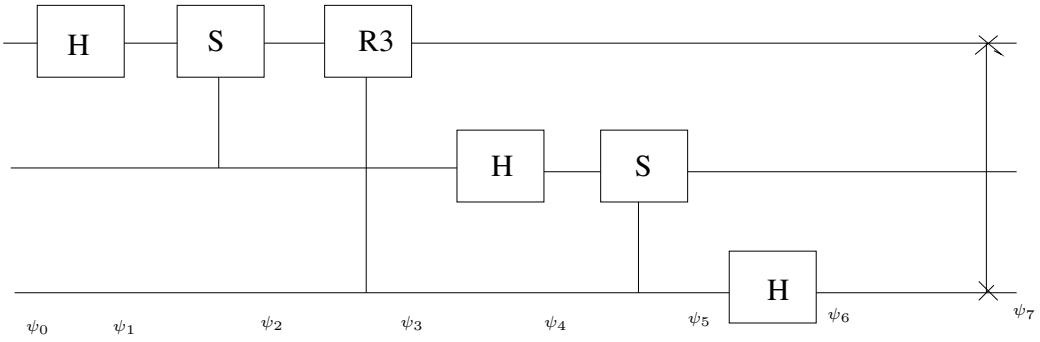

FIGURE 5.3 – Circuit de tranformé de Fourier à 3 qubits équivalent

Posons $\omega = e^{2\pi i/8} = \sqrt{i}$. La matrice qui définit ce circuit est :

$$TFQ_8 = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix}$$

Exemples de la TFQ à 3 qubits Dans ces exemples nous allons calculer notre TFQ à l'aide dans un premier temps du schéma 5.3, puis dans un second temps nous vérifierons

que le résultat est le mêm avec la matrice définie ci dessus.
Prenons l'état $|\phi\rangle = |000\rangle = |0\rangle$ (en décimale). Avec le schéma 5.3 :

$$|\psi_0\rangle = |000\rangle$$
$$|\psi_1\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|100\rangle$$
$$|\psi_2\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|100\rangle$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|100\rangle$$
$$|\psi_4\rangle = \frac{1}{2}|000\rangle + \frac{1}{2}|010\rangle + \frac{1}{2}|100\rangle + \frac{1}{2}|110\rangle$$
$$|\psi_5\rangle = \frac{1}{2}|000\rangle + \frac{1}{2}|010\rangle + \frac{1}{2}|100\rangle + \frac{1}{2}|110\rangle$$
$$|\psi_6\rangle = \frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle + \frac{1}{\sqrt{8}}|010\rangle + \frac{1}{\sqrt{8}}|011\rangle + \frac{1}{\sqrt{8}}|100\rangle + \frac{1}{\sqrt{8}}|101\rangle + \frac{1}{\sqrt{8}}|110\rangle + \frac{1}{\sqrt{8}}|111\rangle$$
$$|\psi_7\rangle = \frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle + \frac{1}{\sqrt{8}}|010\rangle + \frac{1}{\sqrt{8}}|011\rangle + \frac{1}{\sqrt{8}}|100\rangle + \frac{1}{\sqrt{8}}|101\rangle + \frac{1}{\sqrt{8}}|110\rangle + \frac{1}{\sqrt{8}}|111\rangle$$

En utilisant la matrice :

L'état $|\psi\rangle = |000\rangle$ s'écrit comme le vecteur : $\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$ Il suffit donc de le multiplier à la

matrice de la TFQ à 3 qubits :

$$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \times \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} = \frac{1}{\sqrt{8}} \times \begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}$$

Maintenant prenons l'état $|010\rangle$ ou $|2\rangle$ (en décimale), avec le schéma cela nous donne :

$$|\psi_0\rangle = |010\rangle$$
$$|\psi_1\rangle = \frac{1}{\sqrt{2}}|010\rangle + \frac{1}{\sqrt{2}}|110\rangle$$
$$|\psi_2\rangle = \frac{1}{\sqrt{2}}|010\rangle + \frac{i}{\sqrt{2}}|110\rangle$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2}}|010\rangle + \frac{i}{\sqrt{2}}|110\rangle$$
$$|\psi_4\rangle = \frac{1}{2}|000\rangle - \frac{1}{2}|010\rangle + \frac{i}{2}|100\rangle - \frac{i}{2}|110\rangle$$
$$|\psi_5\rangle = \frac{1}{2}|000\rangle - \frac{1}{2}|010\rangle + \frac{i}{2}|100\rangle - \frac{i}{2}|110\rangle$$
$$|\psi_6\rangle = \frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle - \frac{1}{\sqrt{8}}|010\rangle - \frac{1}{\sqrt{8}}|011\rangle + \frac{i}{\sqrt{8}}|100\rangle + \frac{i}{\sqrt{8}}|101\rangle - \frac{i}{\sqrt{8}}|110\rangle - \frac{i}{\sqrt{8}}|111\rangle$$
$$|\psi_7\rangle = \frac{1}{\sqrt{8}}|000\rangle + \frac{i}{\sqrt{8}}|001\rangle - \frac{1}{\sqrt{8}}|010\rangle - \frac{i}{\sqrt{8}}|011\rangle + \frac{1}{\sqrt{8}}|100\rangle + \frac{i}{\sqrt{8}}|101\rangle - \frac{1}{\sqrt{8}}|110\rangle - \frac{i}{\sqrt{8}}|111\rangle$$

Maintenant avec la matrice :

$$\begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix} \times \frac{1}{\sqrt{8}} \begin{pmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^1&\omega^4&\omega^7&\omega^2&\omega^5\\1&\omega^4&1&\omega^4&1&\omega^4&1&\omega^4\\1&\omega^5&\omega^2&\omega^7&\omega^4&\omega^1&\omega^6&\omega^3\\1&\omega^6&\omega^4&\omega^2&1&\omega^6&\omega^4&\omega^2\\1&\omega^7&\omega^6&\omega^5&\omega^4&\omega^3&\omega^2&\omega\end{pmatrix} = \frac{1}{\sqrt{8}} \times \begin{pmatrix}1\\i\\-1\\-i\\1\\i\\-1\\-i\end{pmatrix}$$

Nous allons maintenant prendre un exemple d'état périodique : $\frac{1}{2}(|1\rangle + |3\rangle + |5\rangle + |7\rangle)$(en decimale) $=\frac{1}{2}(|001\rangle + |011\rangle + |101\rangle + |111\rangle)$, avec la matrice cela nous donne : :

$$\frac{1}{2}\begin{pmatrix}0\\1\\0\\1\\0\\1\\0\\1\end{pmatrix} \times \frac{1}{\sqrt{8}} \begin{pmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^1&\omega^4&\omega^7&\omega^2&\omega^5\\1&\omega^4&1&\omega^4&1&\omega^4&1&\omega^4\\1&\omega^5&\omega^2&\omega^7&\omega^4&\omega^1&\omega^6&\omega^3\\1&\omega^6&\omega^4&\omega^2&1&\omega^6&\omega^4&\omega^2\\1&\omega^7&\omega^6&\omega^5&\omega^4&\omega^3&\omega^2&\omega\end{pmatrix} = \frac{1}{2\sqrt{8}} \times \begin{pmatrix}4\\0\\0\\0\\-4\\0\\0\\0\end{pmatrix}$$

Lors des 3 exemples les résultats sont les mêmes que ce soit à l aide du calcul matriciel ou du shéma 5.3. On peut donc dire que la matrice est bien équivalente au schéma 5.3.

Remarque : La TFQ a transformée le signal périodique en un autre signal périodique. De plus la TFQ a initialisé la séquence à $|0\rangle$ ce qui n'était pas le cas dans l'état de départ.

5.1.2 TFQ$_n$

Définition La transformé de Fourier qunatique tout comme la classique change un vecteur de base $\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ ... \\ \alpha_{n-1} \end{pmatrix}$ en sa transformée $\begin{pmatrix} \beta_0 \\ \beta_1 \\ ... \\ \beta_{n-1} \end{pmatrix}$ les coefficients β définissent le nouvelle état de $|\psi\rangle$ transformé par la TFQ dont les propriétés sont :

Propriétés de la TFQ$_n$:
- TFQ$_n$est unitaire
- TFQ$_n^2 = Id$
- TFQ$_n$ transforme un signal périodique en un autre signal périodique en éliminant le point de base, c'est à dire la TFQ transforme un état de la forme : $TFQ_n|\psi\rangle = \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} |x_0 + kr\rangle$ en un état de la forme : $|F\psi\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{\frac{2i\pi}{r} j x_0} |j\frac{2^n}{r}\rangle$ avec la $B_j = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2i\pi}{N} kj} x_j$

Propriété Le circuit général de la Transformée de Fourier quantique est celui là :

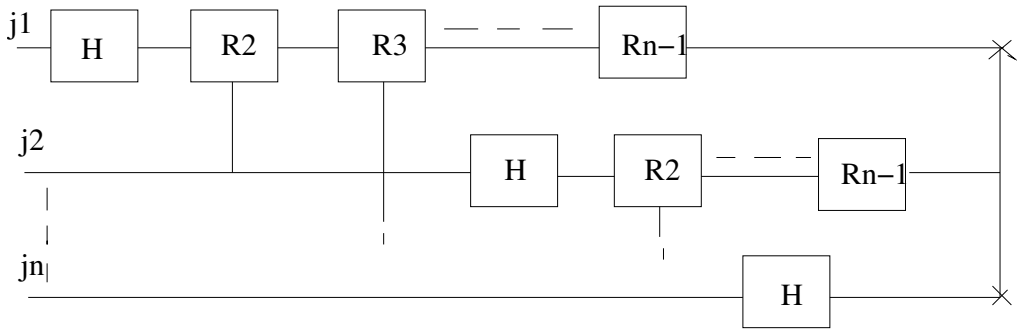

FIGURE 5.4 – Circuit général de la TFQ

Complexité Nous avons :
- sur la première ligne : une porte Hadamard + $(n-1)R_k$ opérations, soient n opérations.
- sur la deuxième ligne : —n-1— opérations
- ...
- Sur la dernière ligne 1 opérations

Il y a en tout n + (n - 1) + ... + 1 = $\frac{n(n+1)}{2}$ opérations. La complexité de la TFQ est donc de O(n^2) (celle de la TF classique est de O($n2^n$)). Nous avons donc un gain qui est exponentielle.

5.2 Algorithme de Shor

5.2.1 Le problème

Dans cet algorithme la factorisation de N, revient à trouver un x tel que $x^2 = 1[N]$. On peut écrire aussi : $x^2 - 1 = 0[N]$ qui se facorise en (x−1)(x+1) = 0[N]. Ce qui implique

que $(x-1)$ et $(x+1)$ sont divisibles par des facteurs de N.
L'algorithme de Sohr va prendre un x au hasard est chercher un r tel que $x^r = 1[N]$ [1]. Si r est pair on aura $y = x^{r/2}$ qui est une racine carré de 1 *mod* N et $pgcd(N, (x^{r/2}-1))$ ou $PGCD(N, (x^{r/2}+1))$ est un diviseur non trivial de N. On est donc ramené à la recherche de la fonction de la période de la fonction $f(k) = x^k$

5.2.2 Recherche de la période de la fonction $f(k) = x^k$

La recherche de la période de la fonction s'effectue avec le circuit quantique suivant : Avec les boîtes TFQ qui représente le circuit de la transformée de Fourier quantique et

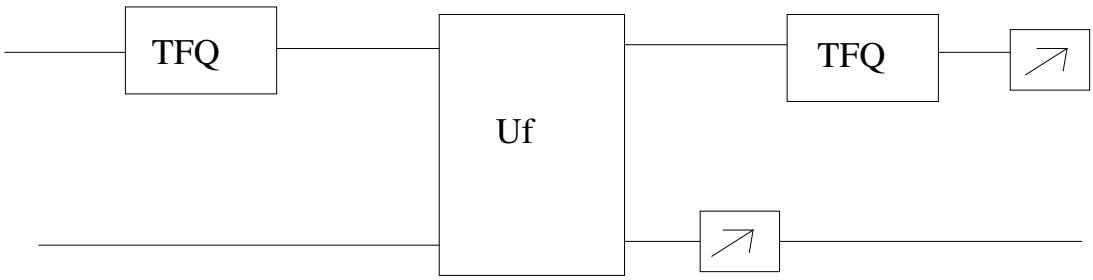

FIGURE 5.5 – Circuit de la recherche d'une fonction

Uf qui représente la boîte capable d'evaluer la fonction ($f(r) = x^r$ sur un état quantique. les autres boites représentent une mesure produite sur le registre correspondant.Au début du circuit l'état est composé de deux registres initialisés à $|0\rangle$ (décimale), l'état de départ s'écrit : $|\psi\rangle = |0\rangle|0\rangle$ On prépare le premier registre à l'aide de la TFQ, on va avoir en sorti un état parallélisé, c'est ce qu'il nous faut pour l'évaluation de la fonction : $|\phi_1\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle|k\rangle|0\rangle$. On évalue la fonction, le deuxième registre est un état initialisé à 0, il va recevoir les résultats de la fonction $|\psi_2\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle|x^k\rangle$. On opère une première mesure sur le deuxième registre pour avoir tous les états qui ont le même résultat de $f(x)$, ce qui nous permettra de trouver la période de la fonction, car sur le premier registre tout les états seront "espacés" de la même période : $|\psi_3\rangle = |\frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} |k_0+k_r\rangle|x^{k_0}\rangle$. On refait une TFQ sur le premier registre pour pouvoir avoir un signal périodique commençant à l'état 0 : $|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |2^r\rangle|x_0\rangle$. On fait un mesure sur le premier registre, avec le résultat on pourra trouver la période et remonter au diviseurs de N.

5.2.3 Exemple de factorisation : factoriser 15 en 5 et 3

On va dans cet exemple factoriser $N = 15$ (produit de 3 et 5) par l'algorithme de shor, dans un espace à 11 qubits.

- On choisit un x au hasard. On prend x = 7
- x est premier avec 15, on passe à la recherche de la période (circuit du dessus)
  - On applique la transformée de Fourier sur le premier registre on a un nouveau vecteur de la forme $|\psi\rangle = \frac{1}{2048}(|0\rangle + |1\rangle + ... + |2047\rangle)$
  - On fait le calcul de la fonction $f(r) = x^r$ sur tout les états en même temps (état parallélisé). On a donc les deux registres qui donne : $|\psi\rangle = \frac{1}{\sqrt{2048}}(|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle...$

1. Grâce à un algorithme d'Euclide qui dit qu'il existe un r tel que $a^r = 1[N]$

- On mesure le deuxième registre on trouve par exemple 4, ce qui nous donne : $|\psi\rangle = \frac{1}{2048}(|2\rangle|4\rangle + |6\rangle|4\rangle + |10\rangle|4\rangle + ...)$
- On refait une transformée de Fourier sur le premier registre ce qui nous donne : $|\psi\rangle = \frac{1}{\sqrt{4}}(|0\rangle + |512\rangle + |1024\rangle|4\rangle + |1536\rangle|4\rangle) = \sum k|\frac{2^{11}}{r}\rangle|4\rangle$
- On mesure le premier registre. On trouve un multiple de $\frac{2^{11}}{r}$ et on peut en déduire r. par exemple si on trouve par 1536. On fait le PGCD(M, 1536) = 512. C'est un diviseur de M [2]

– r est pair on a donc PGCD($x^{r/2}$-1, N) et PGCD($x^{r/2}$+1, N) qui sont égaux à 3 et 5 et qui divise bien 15. [3]

5.2.4 Algorithme et complexité

En reprenant l'étude menée en 5.2.1, 5.2.2 on peut formaliser l'algorithme de Shor :

Data: Entier N (non premier)
Result: un facteur de N
begin
  if *N % 2 = 0* **then**
    Output: 2
  end
  else
    while *r % 2 != 0* **do**
      Tirer au hasard un $x \in [1, N-1]$
      if *PGCD(x, N) > 1* **then**
        Output: PGCD(x, N)
      end
      recherche de la période pour trouver l'ordre r de x^k modulo N
      if *r est pair* **then**
        Output: PGCD($x^{r/2}$-1, N) et PGCD ($x^{r/2}$-1, N)
        L'un au moins est un facteur de N
      end
    end
  end
end

Algorithm 4: Algorithme de Shor de factorisation de nombres entiers en facteurs premiers

Complexité La complexité de l'algorithme est donc l'addition des opérations faites sur ce shéma : $2 \times n^2$ (une fois par TFQ) + la complexité de Uf (polynomiale) car on calcule tout les états en parallèle, ce qui donne au total une complexité de type polynomiale. Nous avons donc un gain qui est ici exponentielle par rapport à l'ordinateur classique ! (pour un ordre d'idée il faut 100 000 ans pour un ordinateur classique actuel pour factoriser un nombre RSA de 1024 bits alors qu'il faudrait seulement 5 minutes pour un ordinateur quantique de 5100 qubits (l'ordinateur classique en a environ 35 milliards (4 Go de RAM)) et un milliards de portes quantiques ... Ce nombre de qubits parait ridicule face au nombre

2. Si le multiple de $\frac{11}{r}$ est déjà un multiple de M on recommence.
3. Voir l'article de nature (disponible en annexe) du 20 décembre 2001 sur la factorisation experimentale de 15 àl'aide de 7 qubites réalisé expérimentalement avec le spin d'électron

de bits de l'ordinateur classique mais c'est énorme à l'heure actuelle. [4]

4. Dans l'article de Nature sur la factorisation expérimentale de 15 seulement 7 qubits était utilisés

Conclusion

Nous avons voulu mettre en évidence le gain en terme de complexité algorithmique apporté par l'algorithme de Shor dans le problème de la factorisation des entiers et ses conséquences sur le chiffrment RSA. La théorie de l'information quantique paraît très prometteur, mais nous n'en sommes qu'au stade de la recherche, les seuls productions actuelles sont expérimentales. A part peut-être un ordinateur (le D-Wave Two) doté de 512 qubits sorti au printemps 2013, fabriqué par l'entreprise canadienne D-Wave, mais le caractère quantique de la machine est toujours discuté par les spécialiste du sujet. Si cet ordinateur est réellement quantique nous pourrions être au coeur d'un changement majeur de l'informatique, beaucoup d'outils utilisés en ce moment comme RSA deviendrait obsolète.
RSA reste néanmoins en chiffrement sûr pour l'instant, il faut quand même faire attention à régulièrement augmenté la taille de n pour être totalement sûr d'être en sécurité, car la puissance des ordinateurs augmente régulièrement.

Grâce à cette UV, j'ai pu largement enrichir mes connaissances sur le sujet de la cryptographie. Mais ce sujet m'a permis aussi d'élargir mes connasaances à l'information quantique. Un sujet très intérressant autant par ces applications (téléportation, algorithme de Shor, et bien d'autres que je n'ai malheureusement pas pu étudier) que par les connaissances qu'il met en jeu, c'est un domaine qui fait appel à la fois au mathématiques (calcul avec les portes, transformée de Fourier, ...), à l'informatique (algorithme, information, ...), et aussi à la physique (réalisation pratique d'un qubit, réalisation expérimentale des portes et de la téléportation, ...).

Bibliographie

[1] MOOC sur la cryptographie fait par Arnaud Bodin sur la plateforme canvas-network.net

[2] Arithmétique et cryptologie de Gilles Bayle-Maître

[3] Quantum computation and quantum Information, de Michael Nielsen et Isaac Chuang, 2010

[4] Introduction à l'informatique quantique, Y. Leroyer, document de cours de l'école ENSEIRB-MATMECA

[5] Article de Nature sur la factorisation de 15, publié 20 décembre 2001

[6] Article de Nature sur l'expérimentation de la téléportation, publié le 19 Août 2004

Annexe A

Certificat MOOC

Annexe B

Programme de factorisation de RSA

Tout les programmes ont été résalisé à l'aide du langage python, un langage que j'ai choisit pour sa simplicité et son efficacité. Ces programme ont été fait avec la version python3.2 sous un système GNU/Linux (distribution : Debian Wheezy)

```
#!/bin/env python
#utf8
import math
import time

boolean = 1
while boolean:
    t1 = time.clock()
    q = int(input())
    i = int(pow(q, 0.5))
    if(i%2 == 0):
        i = i+1
    boolean = False
    while(boolean == False):
        if ((q % i) == 0):
            print(i)
            print(q / i)
            t2 = time.clock()
            print("temps d'execution", t2-t1)
            boolean = True
        else:
            i = i + 2
    print("continue :")
    boolean = int(input())
```

Annexe C

Programme de créations de clés RSA

```
#!/bin/env python
#utf8

import math
import random
import time

# fonction qui cherche deux nombres p et q premiers

def randome(phi):
    boolean = False
    while(boolean == False):
        e = random.randint(2, phi)
        if pgcd(e, phi) == 1:
            boolean = True
            return e

def pq():
    boolean = False
    p = random.randint(3, 10000)
    boolean = isprime(p)
    while(boolean == False):
        if ( p % 2 == 0):
            p = p + 1
            boolean = isprime(p)
        else:
            p = p + 2
            boolean = isprime(p)

    boolean = False

    q = random.randint(3, 10000)
    boolean = isprime(q)
    while(boolean == False):
        if ( q % 2 == 0):
```

```
            q = q + 1
            boolean = isprime(q)
        else:
            q = q + 2
            boolean = isprime(q)
    return p, q

#fonction qui vérifie la primalité d'un nombre
def isprime (p):
    for i in [2, 3, 5, 7]:
        if (pow(i, p-1)%p == 1):
            return True
    return False

#fonction qui calcule le pgcd de 2 nombres
def pgcd(a,b):
    if b==0:
        return a
    else:
        r=a%b
    return pgcd(b,r)

#fonctions qui calcule les coefficients de Bezout
def bezout(a, b):
    r, u = a, 1
    rp, up = b, 0
    while rp != 0:
        q = r//rp
        rs, us = r, u
        r, u = rp, up
        rp, up = (rs - q*rp), (us - q*up)
    return (u)

def invmod(e, phi):
    d = pow(e, phi-2, phi)
    return d

#
#    FONCTION PRINCIPALE
#

#La fonctions time.clock() donne le temps à l'instant précis ou elle appelé, cela perm
t1 = time.clock()
p, q = pq()
t2 = time.clock()
print ("temps de calcul pour trouver p et q", t2-t1)
print("p =", p)
print("q =", q)
```

```
#calcule des différents nombres nécessaires pour le chiffrement RSA
n = p*q

print("n =", n)

phi = (p-1) * (q-1)

print("phi(n) =", phi)

e = randome(phi)

print("e =", e)

d = invmod(e, phi)

print("d=", d)
```

Annexe D

Compléments mathématiques

D.1 Preuve des théorèmes utilisés pour RSA

D.1.1 Indicatrice d'Euler

L'indicatrice d'Euler $\phi(n)$ est égale au nombre de nombres premiers avec n. Donc si n est premier elle vaut $n-1$ et donc si n est le produit de deux facteurs premiers (p et q), elles est égale à : $\phi(n) = \phi(p) et \phi(q) = (p-1) et (q-1)$ Comme p et q sont premiers.

D.1.2 Preuve du théorème de Bezout

La preuve du théorème de Bezout est donné dans le paragraphe 2.1.2 sur l'algorithme d'Euclide qui nous permet de remonter au coefficients u et v.

D.1.3 Preuve du petit théorème de Fermat

Nous allons prouver ce théorème par récurrence pour $a \geq 0$, mais il est nécessaire d'abord d'introduire ce lemme :

lemme $p| \binom{p}{k}$ pour $1 \leq k \leq p-1$, c'est à dire $\binom{p}{k} \equiv 0 (mod$ p$)$

preuve par récurrence du petit théorème de Fermat

- $a = 0$ alors $0 \equiv 0 (mod$ p$)$
- Fixons $a \geq 0$ et supposons que $a^p equiva (modp)$. Nous calculons maintenant $(a+1)^p$ à l'aide de la formule du binôme de Newton :

$(a+1)^p = a^p + \binom{a^{p-1}p}{p-1} \binom{p}{p-2} a^{p-2} \binom{p}{1} + 1$

On réduit maintenant modulo p : $(a+1)^p = a^p + \binom{a^{p-1}p}{p-1} \binom{p}{p-2} a^{p-2} \binom{p}{1} + 1 \text{mod } p$

Grace au Lemme introduit on a :

$$\equiv a^p + 1 (modp) \tag{D.1}$$

L'hypothèse de récurrence nous donne donc :

$$\equiv a + 1 (modp) \tag{D.2}$$

D.1.4 Preuve du lemme de Gauss

Soit a, b, $c \in \mathbb{N}$. Avec a et b premiers entre eux
Si $a|c$ alors

$$c = k \times a \tag{D.3}$$

or $b|c$ donc $b|ka$. a et b étant premiers entre eux, $b|k$. On peut écrire : $k = k'b$ Avec (D.1) on a : $c = ka = k'ba \iff ab|c$

D.2 Espace Hermitien et matrices unitaire

Soit un E=$\mathbb{C}^n$ un espace vectoriel complexe de dimension n.

Definition On appelle produit scalaire hermitien, le corchet $<,>$ tel que pour tout $u, v, w \in E$ et $a, b \in \mathbb{C}$: - $<,>$ est linéaire à droite et semi-linéaire à gauche $< au+v, w >= a < u, w > + < v, w >$ et $< u, av + w >= \overline{a} < u, v > + < u, w >$ - $< u, v >= \overline{< v, u >}$ - $< u, u > \geq 0$ et $< u, u >= 0$ ssi $u = 0$ (défini positif)

Une tranformation unitaire est une application linéaire $f : E \to E$ qui respecte le produit hermitien, c'est-à-dire telle que ¡f(u),f(v)¿=¡u,v¿.

Proposition Soit U la matrice dans une base othonormée pour $<,>$ d'une application unitaire. Alors U vérifie $U^*U = Id$ ou U^* est la transposée de la matrice conjugué de U.