De la sécurité calculatoire des protocoles cryptographiques devant la menace quantique

Mémoire

Louis Fortier-Dubois

Maîtrise en informatique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

De la sécurité calculatoire des protocoles cryptographiques devant la menace quantique

Mémoire

Louis Fortier-Dubois

Sous la direction de:

Pascal Tesson, directeur de recherche

Résumé

On ne s'en inquiète peut-être pas assez, mais toute communication confidentielle sur Internet, dont on prend désormais la sécurité pour acquise, pourrait du jour au lendemain devenir très facile à espionner. Nous savons en effet qu'un ordinateur quantique, s'il en existe un de suffisante envergure, pourra –ou peut déjà, qui sait ?– rendre obsolète les protocoles cryptographiques qui nous permettent de gérer nos comptes utilisateurs, faire des transactions bancaires et simplement d'avoir des conversations privées. Heureusement, une communauté de chercheurs se penche déjà sur des protocoles alternatifs ; cependant chacune des propositions est isolée dans son propre sous-domaine de recherche et il est difficile de faire la lumière sur laquelle est la plus prometteuse. À travers trois horizons, explorant respectivement pourquoi la cryptographie actuelle est considérée sécuritaire, comment l'arrivée d'un seul ordinateur quantique sur la planète changera toute la cryptographie, et que faire pour communiquer confidentiellement dans un monde où l'informatique quantique est omniprésente, nous développons un cadre uniforme pour analyser lesquels de ces nouveaux protocoles cryptographiques sont assis sur les bases théoriques présageant la plus grande sécurité.

Table des matières

Liste des figures

Quand nous défendons le français chez nous, ce sont toutes les langues du monde que nous défendons contre l'hégémonie d'une seule.

Pierre Bourgault

Introduction

La pluralité des langues est ce qui permet encore aujourd'hui à des communautés de partager et célébrer leur culture simplement en communiquant, malgré la progression d'un mode de pensée uniforme conséquent à la mise en place de technologies de l'information à l'échelle mondiale. Paradoxalement, si l'Internet est un danger pour la diversité des langues naturelles, il supporte aussi la possibilité de générer des langages uniques entre toute paire d'individus souhaitant échanger des idées de façon privée, à l'abri des oreilles indiscrètes. Mais à l'instar des langues locales face à la croissance de l'anglais, la cryptographie, science permettant de créer ces façons de communiquer confidentiellement, est aujourd'hui en péril. En fait, une menace pèse sur elle depuis déjà plus de 20 ans.

En 1994, le mathématicien Peter Shor invente un algorithme permettant de comprendre toute communication utilisant un protocole cryptographique conventionnel. Cet algorithme n'a toutefois pas encore eu d'impact grave puisqu'il ne peut être exécuté que sur un ordinateur quantique. Or, ce type de machine est encore à l'étape de prototype ; on sait qu'un tel ordinateur *peut* exister, mais une foule de difficultés techniques ont empêché les ingénieurs d'en déployer un de suffisamment grande envergure pour appliquer l'algorithme de Shor. Mais cela n'est possiblement qu'une question de temps !

Pourtant, encore aujourd'hui, on s'entête à utiliser les mêmes protocoles cryptographiques que l'on sait vulnérables au calcul quantique. Bien des experts se sont penchés sur des protocoles alternatifs, mais ces derniers sont encore très peu, pour ne pas dire pas du tout, utilisés, principalement car ils sont beaucoup plus lourds à exécuter. Nous croyons que cette lourdeur pourrait être fortement diminuée si la communauté scientifique se concentrait sur les protocoles les plus prometteurs au lieu d'explorer des avenues dont la sécurité est plus douteuse, comme c'est le cas actuellement. Nous avons donc développé un *cadre uniforme* pour analyser la sécurité d'un protocole cryptographique, qu'il soit parmi les protocoles contemporains (en cours d'utilisation), post-quantiques (conçus pour fonctionner sur nos ordinateurs actuels, mais résistants aux attaques quantiques) ou quantiques (nécessitant des appareils quantiques pour la communication), dans le but que le lecteur sache où concentrer ses efforts pour qu'ils portent fruit.

À cette fin, nous tentons de répondre à trois objectifs distincts :

1. Dresser un portrait du futur de la cryptographie à clé publique dans un scénario où

l'arrivée des ordinateurs quantiques est imminente, de façon complète et accessible.

2. Identifier les différences fondamentales entre les problèmes résolubles efficacement par des ordinateurs classiques, les problèmes résolubles efficacement par des ordinateurs quantiques seulement, et les problèmes demeurant hors de portée même après l'arrivée du paradigme quantique, en analysant la complexité des problèmes algébriques sous-jacents.
3. Proposer des mesures à prendre pour que le design de protocoles cryptographiques soit toujours en avance sur le domaine de la cryptanalyse, sachant que l'ordinateur quantique sera un outil de cryptanalyse très puissant.

Nous étudions donc plusieurs protocoles cryptographiques provenant de sous-domaines hétéroclites et mettons en évidence leurs ressemblances et différences pour obtenir une compréhension profonde de pourquoi certains cèdent aux attaques quantiques et d'autres pas. Nous faisons notre analyse selon le cadre théorique de la complexité de calcul, en étant le plus général possible, c'est-à-dire que nous concentrons notre attention sur les protocoles les plus primitifs possibles en faisant très peu d'hypothèses sur les capacités de l'attaquant. Après avoir cerné en détails le danger que court la cryptographie (sections 1.1 à 2.2) et présenté une foule de propositions alternatives à ce qui est utilisé actuellement (sections 2.3 à 3.2), nous comparons les solutions proposées par la communauté à l'aide de critères objectifs et uniformes (section 3.3).

Nous commençons le tout par un court chapitre résumant les notions préalables afin d'épurer les chapitres suivants et ainsi en faciliter la lecture. Nous avons divisé ces chapitres subséquents en horizons, représentant chacun une période dans laquelle le monde de la cryptographie se retrouvera dans le futur, advenant l'apparition d'ordinateurs quantiques d'envergure :

— **Horizon I : L'époque contemporaine.** Il n'existe aucun ordinateur quantique d'envergure. Les protocoles cryptographiques en place ne sont pas encore obsolètes mais on peut d'ores et déjà analyser leur sécurité pour comprendre pourquoi, en termes de complexité de calcul, un ordinateur quantique pourra les briser. Ce chapitre explore la théorie mathématique supportant les protocoles cryptographiques actuels (théories des nombres et théorie des groupes) et la base de leur sécurité selon la théorie de la complexité.

— **Horizon II : La période de transition.** Supposons qu'un bon ordinateur quantique est développé quelque part dans le monde. Son détenteur peut briser les systèmes actuels, alors il est important que cet horizon ait été envisagé auparavant et que des systèmes plus résistants, dits « post-quantiques », aient été mis en place. Ces systèmes ont la particularité d'être utilisables par des ordinateurs standards mais impossibles à briser avec des ordinateurs quantiques. Ce chapitre explore en détails le paradigme du calcul quantique, les algorithmes quantiques brisant les protocoles actuels, et les alternatives post-quantiques proposées.

— **Horizon III : L'ère quantique.** Quelques années après la réalisation d'un premier ordinateur quantique d'envergure, il devrait être possible de miniaturiser et commercialiser le prototype, de façon à ce que du matériel capable de calcul quantique soit commun dans les ordinateurs personnels. On pourra alors utiliser la cryptographie « quantique », où les propriétés quantiques de la matière sont utilisées par les entités communicantes et pas seulement par un espion. Ce chapitre explore la théorie de l'information quantique, les protocoles de cryptographie quantique proposés jusqu'à maintenant, puis compare ces protocoles quantiques aux protocoles présentés précédemment en terme de sécurité.

Préalables

Cet ouvrage nécessite plusieurs notions mathématiques relativement avancées, notamment en **théorie des ensembles**, **probabilités**, **analyse d'algorithmes**, **complexité de calcul**, **théorie des groupes**, **théorie des nombres** et **algèbre linéaire**. Nous assumons uniquement de la part du lecteur des connaissances élémentaires en théorie des ensembles et en probabilités ; les autres sont révisées dans ce court chapitre [1].

Toutes les notions de **cryptographie** et de **mécanique quantique** dont nous nous servirons seront définies au fur et à mesure dans les chapitres suivants.

1. Cette révision est toutefois rapide et va droit au but ; le lecteur néophyte est encouragé à s'introduire à ces notions à l'aide d'autres ouvrages.

0.1 Complexité de calcul

La complexité de calcul est un cadre théorique pour hiérarchiser les problèmes algorithmiques selon la quantité de ressources nécessaires à leur résolution. Un problème algorithmique est une question s'appliquant à diverses entrées, dont la réponse pour une entrée donnée se calcule à l'aide d'un algorithme.

Exemple 0.1 (Problème algorithmique)**.**

Primalité	
Entrée :	Un nombre $n \in \mathbb{N}$
Question :	n est-il premier ?

Ici, l'entrée n peut être n'importe quel nombre naturel. Si l'on demande plus précisément si $n = 97$ est premier (la réponse est oui), il s'agit d'une *instance* de Primalité, i.e. un problème algorithmique dont l'entrée est fixée.

Le calcul de la réponse peut être fait de différentes façons : on pourrait essayer de diviser 97 par tous les nombres de 2 à 96 et répondre oui si aucun d'entre eux ne fonctionne. Plus intelligemment, on pourrait cesser la recherche lorsqu'on dépasse $\sqrt{97}$, c'est-à-dire à 10. Une autre amélioration serait de n'essayer que les nombres impairs, puisque 97 est impair. Cela ne demanderait que de vérifier pour $3, 5, 7$ et 9, pour un total de 4 essais (comparé à 95 avec la première méthode !). Bien d'autres améliorations existent, et sont nécessaires pour arriver à vérifier la primalité de très grands nombres en un temps raisonnable.

Cependant, vient un temps où l'on n'arrive plus à améliorer la méthode de résolution du problème. Il semblerait qu'il faut toujours un nombre minimal d'étapes pour pouvoir calculer la réponse. Cela vient de la complexité inhérente au problème, définie par la complexité du meilleur algorithme le résolvant.

0.1.1 Analyse d'algorithmes

Un algorithme est une séquence d'opérations menant à la solution relative à une instance, et fonctionnant peu importe l'instance. La complexité d'un algorithme est une mesure du nombre d'opérations effectuées en fonction de la taille de cette instance. Par exemple, l'algorithme 1 ci-dessous s'exécute en $\mathcal{O}(n^2)$ car on passe sur chaque élément et, pour chacun, on repasse une autre fois sur tous les éléments. Cette notation signifie que, en pire cas, un nombre de l'ordre de n^2 d'opérations devront être effectuées. Ainsi, une entrée seulement 2 fois plus longue prendra environ 4 fois plus de temps à s'exécuter !

Le cas de l'algorithme 2 est plus dramatique : le nombre en entrée étant sur n bits, il faut passer sur tous les nombres de n bits, qui sont au nombre de 2^n. Cette fonction augmente

Algorithme 1 Boucle imbriquée

```
1: Entrée : Une liste l de n éléments
2: pour chaque élément de l faire
3:     pour chaque élément de l faire
4:         ...
5:     fin pour
6: fin pour
```

beaucoup plus rapidement que n^2 à mesure que n grandit. Un algorithme s'exécutant en $\mathcal{O}(2^n)$ fonctionne sur une petite instance, mais devient rapidement impossible : un nombre de 20 bits requiert déjà 1 million d'étapes, et un nombre de 273 bits requerrait davantage d'étapes qu'il n'y a d'atomes dans l'univers observable !

Algorithme 2 Incrémentation d'entiers

```
1: Entrée : Un nombre x décrit sur n bits
2: i := 0
3: tant que i < x faire
4:     ...
5:     i := i + 1
6: fin tant que
```

Cette explosion combinatoire fait en sorte que l'on considère généralement seulement les algorithmes prenant en pire cas un temps polynomial, c'est-à-dire en $\mathcal{O}(n^k)$ pour un k fixe, comme réalistement exécutables.

Énoncé 0.2. Un algorithme est considéré prendre un temps raisonnable si et seulement s'il s'exécute en **temps polynomial**. Conséquemment, un problème algorithmique est considéré soluble si et seulement s'il existe un algorithme polynomial y répondant.

Nous considérons donc que, en pratique, tout algorithme prenant un temps de calcul polynomial peut être exécuté sans problème. Cela inclut les algorithmes *probabilistes*, qui incorporent des bits de hasard dans leurs prises de décision pour accélérer le temps de calcul moyen. Par exemple, le test de Fermat (algorithme 3) tire des nombres aléatoirement pour calculer en moyenne beaucoup plus rapidement si un nombre est premier. En supposant un accès à une source aléatoire, ces algorithmes peuvent très bien exister en pratique.

Un autre type d'algorithme, de nature théorique seulement cette fois, est l'algorithme *non déterministe*. Comme pour les algorithmes probabilistes, des bits régissant certaines prises de décision sont incorporés dans le calcul pour l'accélérer ; la différence est qu'on considère que ces bits permettent de prendre exactement toutes les bonnes décisions. C'est donc comme un algorithme probabiliste qui serait très chanceux et tomberait toujours sur la bonne solution.

Algorithme 3 Test de Fermat

Entrée : Un nombre x à tester et un paramètre k de taille polynomiale en le nombre de bits de x
$i := 0$
pour $i < k$ **faire**
    Tirer un nombre a aléatoirement dans $\{2, ..., x-2\}$
    si $a^{x-1} \neq 1 \pmod{n}$ **alors**
        Retourner *composé*
    fin si
fin pour
Retourner *(probablement) premier*

0.1.2 Classes de complexité

Une classe de complexité est un ensemble de problèmes algorithmiques nécessitant les mêmes ressources. La classe de complexité P (pour *Polynomial-Time*) contient tous les problèmes résolubles grâce à un algorithme polynomial déterministe (donc n'utilisant pas le hasard). Plus formellement, un problème est dans P s'il existe un algorithme s'exécutant en au plus $\mathcal{O}(n^k)$ étapes. Suivant l'énoncé 0.2, on peut considérer tout élément de P comme calculable en pratique. Par exemple, le problème de trouver le plus grand commun diviseur est dans P grâce à l'algorithme 8 (section 0.2.2).

La classe NP (pour *Nondeterministic Polynomial-Time*) regroupe tous les problèmes résolubles grâce à un algorithme non déterministe polynomial. En d'autres mots, il s'agit des problèmes où un algorithme peut prendre plusieurs chemins de taille polynomiale. Chaque chemin peut le mener à accepter l'instance ou la rejeter. Une instance est positive si au moins un de ces chemins mène à l'acceptation ; sinon l'instance est négative. Pour s'exécuter en temps polynomial, l'algorithme ne peut pas essayer tous les chemins, car leur nombre est exponentiel ; sans perte de généralité, l'algorithme n'explore qu'un seul chemin, lequel peut être choisi de façon non déterministe. L'algorithme non déterministe doit avoir ces propriétés :

— Si l'algorithme s'exécute sur une instance négative, alors tous les chemins mènent au rejet et donc l'algorithme peut choisir n'importe quel chemin.

— Si l'algorithme s'exécute sur une instance positive, alors l'algorithme choisit exactement un des chemins qui mènent à l'acceptation.

Un tel algorithme *connaît* le chemin à suivre grâce au non-déterminisme, et ne peut donc pas exister en pratique. Cependant, cela indique qu'il est possible, bien qu'improbable, de trouver la solution en un temps polynomial. On ne peut être certain qu'une instance est une instance négative qu'en vérifiant que tous les chemins rejettent, tandis qu'il suffit de trouver un seul chemin acceptant pour être certain qu'il s'agit d'une instance positive. Ce chemin constitue donc une preuve que l'instance est positive. On peut donc définir NP comme l'ensemble des problèmes pour lesquels il existe un algorithme qui prend en entrée une instance et une preuve

et qui détermine en temps polynomial si la preuve permet d'affirmer que l'instance est une instance positive. À titre d'exemple, le problème SOMME DE SOUS-ENSEMBLES (définition 0.3) est dans NP grâce à l'algorithme 4.

Définition 0.3.

SOMME DE SOUS-ENSEMBLES	
Entrée :	Des entiers $a_1, a_2, ..., a_n, s \in \mathbb{N}$
Question :	Existe-t-il un sous-ensemble $A \subseteq \{a_1, ..., a_n\}$ tel que $\sum_{a \in A} a = s$?

Algorithme 4 Algorithme non déterministe pour SOMME DE SOUS-ENSEMBLES

1: **Entrée :** $\mathbf{a} = (a_1, a_2, ..., a_n) \in \mathbb{N}^n, s \in \mathbb{N}$
2: Sélectionner $\mathbf{x} \in \{0, 1\}^n$ de façon non déterministe
3: Vérifier que $\mathbf{a} \cdot \mathbf{x} = s$, où $\cdot$ représente le produit scalaire.
4: **si** c'est le cas **alors**
5: **Retourner** *oui*
6: **sinon**
7: **Retourner** *non*
8: **fin si**

0.1.3 Réductions

Une réduction d'un problème A vers un problème B est une fonction f qui transforme une instance de A en une instance de B de sorte que l'instance obtenue est positive si et seulement si l'instance en entrée l'était aussi.

Supposons que l'on connaisse un algorithme g résolvant le problème B, mais initialement aucun pour le problème A. Alors en transformant l'instance de A en une instance de B et en résolvant cette dernière, on obtient la solution de l'instance de A. On a donc indirectement un algorithme pour le problème A :

Algorithme 5 Résolution du problème A

1: **Entrée :** I_A, une instance du problème A
2: $I_B = f(I_A)$
3: **Retourner** $g(I_B)$

Soient $t_f(n)$ et $t_g(n)$ les temps de calcul respectifs de f et g en fonction de la taille n d'une instance. Alors en prenant la taille $|I_A|$ de l'instance, l'algorithme pour A est en $\mathcal{O}(t_f(|I_A|) + t_g(|f(I_A)|)) = \mathcal{O}(\max(t_f(|I_A|), t_g(|f(I_A)|)))$. Ainsi, si la réduction est suffisamment simple (si $f(I_A)$ donne une instance de taille raisonnable rapidement), A ne peut pas être plus difficile que B. Une façon intuitive de montrer que A n'est pas plus difficile que B est de supposer l'existence d'un algorithme efficace pour B et de l'utiliser comme une boîte noire dans un algorithme pour A. Nous utiliserons cette stratégie fréquemment par la suite.

Par exemple, on peut dire que SOMME DE SOUS-ENSEMBLES $\leq$ SAC À DOS (i.e. SOMME DE SOUS-ENSEMBLES se réduit au problème du SAC À DOS de la définition 0.4) grâce à l'algorithme 6. SAC À DOS est donc au moins aussi difficile que SOMME DE SOUS-ENSEMBLES.

Définition 0.4.

SAC À DOS	
Entrée :	Des entiers $a_1, a_2, ..., a_n, s, w_1, w_2, ..., w_n, c \in \mathbb{N}$
Question :	Existe-t-il un sous-ensemble d'index $I \subseteq \{1, ..., n\}$ tel que $\sum_{i \in I} a_i \geq s$ et $\sum_{i \in I} w_i \leq c$?

Intuitivement, il s'agit de maximiser le total des a_i (qui représentent la valeur des objets) sans que le total des w_i correspondants (qui représentent leur poids) ne dépasse la capacité c d'un sac à dos.

Algorithme 6 Réduction de SOMME DE SOUS-ENSEMBLES à SAC À DOS

1: **Entrée :** Une instance de SOMME DE SOUS-ENSEMBLES, i.e. des entiers $a_1, a_2, ..., a_n, s \in \mathbb{N}$, et un algorithme boîte noire pour SAC À DOS
2: Poser $w_i := a_i \ \forall i \in \{1, ..., n\}$
3: Poser $c := s$
4: **Retourner** Algorithme pour SAC À DOS $(a_1, ..., a_n, s, w_1, ...w_n, c)$

0.1.4 NP-complétude

Au sein d'une même classe, des problèmes peuvent être plus faciles que d'autres : en particulier, tous les problèmes de P sont aussi dans NP puisqu'un algorithme déterministe peut être vu comme un algorithme non déterministe n'ayant qu'un seul chemin. Pourtant, nous n'avons jamais découvert d'algorithme polynomial pour certains problèmes de NP comme SOMME DE SOUS-ENSEMBLES.

Ce dernier est en fait un exemple typique de la classe bien spéciale des problèmes NP-complets. Ces éléments sont les plus difficiles de NP car tout autre problème de cette classe s'y réduit. N'étant pas résolubles de façon déterministe et polynomiale selon nos connaissances, on les considère trop complexes à résoudre avec nos moyens actuels. La réduction 6 a montré que SAC À DOS n'est pas plus facile, pourtant il s'agit aussi d'un problème de NP (étant donné un ensemble d'éléments optimal, on peut vérifier facilement s'il satisfait aux conditions). Cela implique que SAC À DOS doit lui aussi être un problème NP-complet.

Il existe en fait une foule de problèmes de ce type, tous aussi difficiles car on peut les transformer entre eux. La découverte d'un algorithme efficace pour un seul d'entre eux impliquerait que tous ces problèmes seraient en fait faciles, et les classes P et NP seraient en fait une seule et même classe. Bien que nous n'ayons jamais pu montrer l'absence d'un tel algorithme, son

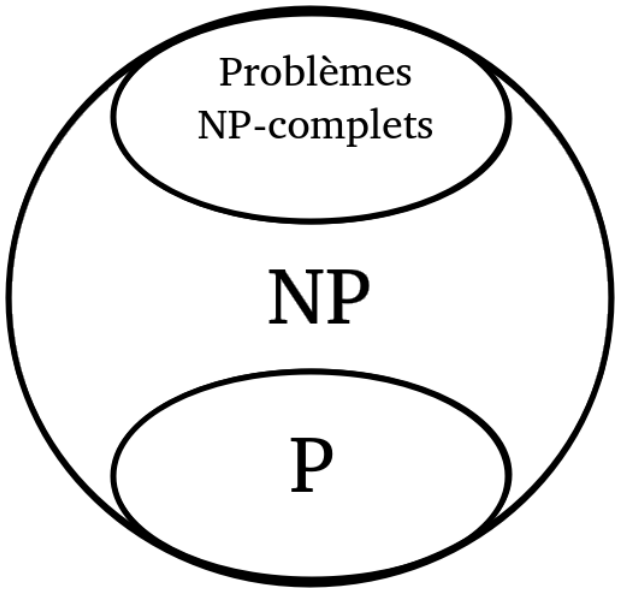


Figure 0.1 – Relations entre les classes P, NP et les problèmes NP-complets

existence est improbable étant donnée la quantité de recherche infructueuse des experts. Nous considérerons donc la conjecture suivante pour la suite :

Conjecture 0.5. $\mathsf{P} \neq \mathsf{NP}$

Nous considérons donc qu'il existe des problèmes dans $\mathsf{NP} \setminus \mathsf{P}$; toutefois un problème peut être plus difficile que ceux de P sans être pour autant NP-complet. En effet, rien n'empêche que des problèmes aient une complexité se situant quelque part entre les deux (voir la figure 0.1) [2].

2. À vrai dire, en supposant $\mathsf{P} \neq \mathsf{NP}$, nous *savons* que la classe NP-intermediate, contenant les problèmes de NP n'étant ni NP-complets ni dans P, n'est pas vide [42].

0.2 Algèbre

0.2.1 Théorie des groupes

Une algèbre est un ensemble muni d'opérations agissant sur cet ensemble. On caractérise différentes structures algébriques selon le nombre d'opérations et les différentes propriétés qui régissent ces opérations. Le type d'algèbre dont nous nous servirons le plus est le **groupe**, qui consiste en un ensemble G (dont la taille peut être finie ou infinie, mais nous nous servirons presque uniquement de groupes finis) muni d'une opération $\cdot$ satisfaisant les propriétés suivantes :

— L'opération $\cdot$ est une loi interne, i.e. $x, y \in G \Rightarrow x \cdot y \in G$;

— L'opération $\cdot$ est associative, i.e. $\forall x, y, z \in G, x \cdot (y \cdot z) = (x \cdot y) \cdot z$;

— L'opération $\cdot$ possède un élément neutre, i.e. $\exists e \in G, \forall x \in G, x \cdot e = e \cdot x = x$;

— Chaque élément a un inverse par rapport à $\cdot$, i.e. $\forall x \in G, \exists x^{-1} \in G, x \cdot x^{-1} = e$.

Exemple 0.6 ($\mathbb{Z}_7^*$)**.** L'ensemble $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$ des entiers modulo 7 sauf 0 est, lorsque muni de la multiplication, un groupe. Son élément neutre est 1 et l'inverse d'un élément x est le y tel que $x \cdot y = 1$ [3]. Par exemple, l'inverse de 5 est 3 car $5 \cdot 3 = 15 = 1 \pmod 7$ et $5 \cdot x \neq 1$ si $x \neq 3$.

Si on ajoute la propriété de commutativité, c'est-à-dire $\forall x, y \in G, x \cdot y = y \cdot x$, alors on dit que le groupe est **abélien**.

L'**ordre d'un élément** $g \in G$ est le plus petit entier r tel que $g^r = 1$. L'**ordre du groupe** est quant à lui la cardinalité $|G|$.

Prenons un sous-ensemble $S \subseteq G$, et notons $\langle S \rangle$ l'ensemble des éléments que l'on peut obtenir à partir de S en appliquant $\cdot$ arbitrairement à ses éléments. Deux cas peuvent survenir.

Si $\langle S \rangle = G$, alors on dit que S est un **ensemble générateur** de G. Si $|S| = 1$ alors l'élément g de S est appelé le **générateur** de G. On a alors que $\forall x \in G, \exists k \in \{0, ..., |G|\}, x = g^k$. G est alors appelé un **groupe cyclique** puisque tout élément peut être obtenu en poursuivant un cycle parmi les puissances de g. L'ordre de g est alors $|G|$ puisque cela nous fait passer par exactement $|G|$ éléments avant de revenir à g. Si G est cyclique alors il est aussi abélien puisque :

$$x \cdot y = g^{k_1} \cdot g^{k_2} = g^{k_1 k_2} = g^{k_2 k_1} = g^{k_2} \cdot g^{k_1} = y \cdot x$$

3. Un tel y existe et est unique, à condition que x et le modulo soient premiers entre eux, ce qui est nécessairement le cas pour le modulo 7.

Exemple 0.7 (Groupe cyclique)**.** Le groupe $\mathbb{Z}_7^*$ est cyclique puisqu'il est généré par 3 :

$$\begin{aligned} 3^1 &= 3 = 3 \pmod 7 \\ 3^2 &= 3 \cdot 3 = 9 = 2 \pmod 7 \\ 3^3 &= 2 \cdot 3 = 6 = 6 \pmod 7 \\ 3^4 &= 6 \cdot 3 = 18 = 4 \pmod 7 \\ 3^5 &= 4 \cdot 3 = 12 = 5 \pmod 7 \\ 3^6 &= 5 \cdot 3 = 15 = 1 \pmod 7 \\ 3^7 &= 1 \cdot 3 = 3 = 3 \pmod 7 \end{aligned}$$

Il est aussi possible que $\langle S \rangle \subset G$, auquel cas $\langle S \rangle$ est un **sous-groupe** de G, i.e. un autre groupe muni de la même opération mais ne contenant pas tous les éléments de G, à condition qu'il inclue l'élément neutre e de G. En appliquant un élément de G différent de e à tous les éléments du sous-groupe, on obtient un ensemble distinct de même taille ne contenant pas l'élément neutre, qu'on appelle **coset**.

Exemple 0.8 (Sous-groupe et coset)**.** Restons avec les entiers modulo 7 mais prenons $S = \{2\}$:

$$\begin{aligned} 2^1 &= 2 = 2 \pmod 7 \\ 2^2 &= 2 \cdot 2 = 4 = 4 \pmod 7 \\ 2^3 &= 4 \cdot 2 = 8 = 1 \pmod 7 \\ 2^4 &= 1 \cdot 2 = 2 = 2 \pmod 7 \end{aligned}$$

On est revenu à 2 avant d'être passé par tous les éléments. L'ensemble $\{1, 2, 4\}$ est donc un sous-groupe généré par 2. En multipliant par 3 chacun de ces éléments, on obtient le coset $\{3, 6, 5\}$.

0.2.2 Théorie des nombres

Nous utilisons la notation $\mathbb{Z}_n$ pour représenter l'ensemble des entiers modulo n, formant un groupe lorsque muni de l'*addition*, tandis que $\mathbb{Z}_n^*$ représente la restriction de $\mathbb{Z}_n$ aux éléments *premiers avec* n (c'est-à-dire ne partageant avec n aucun autre facteur que 1), ensemble qui forme un groupe lorsque muni de la *multiplication*. La cardinalité de ce dernier groupe est donnée par l'indicatrice d'Euler. Soit $N = \{p \in \mathbb{P} \mid p \text{ divise } n\}$, où $\mathbb{P}$ dénote l'ensemble de tous les nombres premiers (i.e. divisibles uniquement par 1 et eux-mêmes), alors l'indicatrice

d'Euler est :

$$\varphi(n) = n \prod_{p \in N} (1 - \frac{1}{p})$$

La taille du groupe est maximale, et le groupe cyclique, lorsque $n \in \mathbb{P}$, car on a $\varphi(n) = n(1 - \frac{1}{n}) = n - 1$, donc tous les éléments de $\mathbb{Z}_n$ sauf 0.

Notation. Un nombre premier est généralement noté p, donc on utilise la notation $\mathbb{Z}_p^*$ pour signifier le groupe multiplicatif modulo un nombre premier.

Ce groupe est très utile en cryptographie, mais nécessite d'identifier de très grands nombres premiers. Or, on peut trouver un nombre premier arbitrairement grand en temps polynomial avec haute probabilité grâce à l'algorithme suivant :

Algorithme 7 Trouver un nombre premier

Entrée : Un nombre de bits n
tant que rien n'est retourné **faire**
    Sélectionner un nombre p impair de n bits
    Tester la primalité de p
    si p est premier **alors**
        Retourner p
    fin si
fin tant que

Le théorème des nombres premiers indique qu'un nombre de n bits tiré au hasard a une probabilité d'environ $\frac{1}{n}$ d'être premier [4], et il est possible de vérifier qu'il est probablement premier rapidement avec un test comme celui de Fermat (algorithme 3).

On peut vérifier que deux nombres x et y sont premier entre eux simplement en vérifiant que leur *plus grand commun diviseur* est 1. Ce nombre, noté $pgcd(x, y)$, peut être calculé en temps polynomial grâce à l'algorithme récursif 8.

Algorithme 8 Algorithme d'Euclide

Entrée : Deux entiers $x, y \in \mathbb{N}$
si $y := 0$ **alors**
    Retourner x
sinon
    Retourner Plus grand commun diviseur ($x := y$, $y := x \mod y$)
fin si

Des opérations essentielles sur le groupe $\mathbb{Z}_p^*$ peuvent elles aussi être calculées en temps polynomial :

— l'inverse modulaire, i.e. trouver, pour un $x \in \mathbb{Z}_p^*$, l'élément $x^{-1} \in \mathbb{Z}_p^*$ tel que $xx^{-1} = 1 \pmod p$, peut être calculé avec le célèbre algorithme d'Euclide étendu (voir [60] par exemple) ;

— l'exponentiation modulaire (algorithme 9), i.e. trouver $y \in \mathbb{Z}_p^*$ tel que $y = x^k$, où $x \in \mathbb{Z}_p^*, k \in \mathbb{N}$. Comme k est divisé par deux à chaque étape, l'algorithme est en $\mathcal{O}(\log k)$, donc polynomial en le nombre de bits de k.

Algorithme 9 Exponentiation modulaire rapide

Entrée : Des entiers $p \in \mathbb{P}$, $x \in \mathbb{Z}_p^*$, $k \in \mathbb{N}$
$y := 1$
tant que $k > 0$ **faire**
    si $k = 1 \pmod 2$ **alors**
        $y := xy \pmod p$
    fin si
    $k := \lfloor \frac{k}{2} \rfloor$
    $x := x^2 \pmod p$
fin tant que

0.2.3 Théorie des corps

Un **corps** $\mathbb{K}$ est une structure algébrique munie de deux opérations, que nous noterons comme l'addition et la multiplication, car bien qu'elles puissent être plutôt différentes de ces opérations conventionnelles, elles doivent respecter des contraintes similaires. En fait, un corps est une structure qui agit essentiellement comme un groupe abélien à la fois sous l'addition et sous la multiplication, et où la multiplication est distributive par rapport à l'addition :

— L'opération $+$ est une loi interne, associative et commutative ;

— L'opération $+$ possède un élément neutre, noté 0, et chaque élément a un inverse par rapport à $+$, i.e. $\forall x \in \mathbb{K}, \exists (-x) \in \mathbb{K}, x + (-x) = 0$;

— L'opération $\cdot$ est une loi interne, associative et commutative ;

— L'opération $\cdot$ possède un élément neutre, noté 1, et chaque élément, à l'exception de 0, a un inverse par rapport à $\cdot$, i.e. $\forall x \in \mathbb{K} \setminus \{0\}, \exists x^{-1} \in \mathbb{K}, x \cdot x^{-1} = 1$;

— L'opération $\cdot$ est **distributive** par rapport à $+$, i.e.

$$\forall x, y, z \in \mathbb{K}, x \cdot y + x \cdot z = x \cdot (y + z)$$

L'exemple le plus typique de corps est l'ensemble des réels $\mathbb{R}$, muni de l'addition et de la multiplication habituelles. Un autre exemple est celui des **nombres complexes**,

$$\mathbb{C} = \{a + bi \mid a, b \in \mathbb{R}, i = \sqrt{-1}\}$$

dont l'addition est définie par

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

et la multiplication par

$$(a + bi) \cdot (c + di) = (ac - bd) + (bc + ad)i$$

la soustraction dans cette dernière équation venant du fait que $i^2 = (\sqrt{-1})^2 = -1$.

La **valeur absolue** d'un nombre complexe $a + bi$ est donnée par

$$|a + bi| = \sqrt{a^2 + b^2}$$

et son **conjugué** par

$$(a + bi)^* = (a - bi)$$

On appelle **racine de l'unité** de n tout nombre complexe z tel que

$$z^n = 1$$

donc l'un des n éléments de l'ensemble

$$\{e^{\frac{2\pi ik}{n}} \mid n \in \mathbb{N} \setminus \{0\}, k \in \{0, 1, ..., n-1\}\}$$

Les nombres complexes nous seront nécessaires lorsque nous traiterons de mécanique quantique. Pour le reste, nous utiliserons un corps discret et fini : $\mathbb{Z}_p$, contenant les entiers de 0 à $p - 1$, où $p \in \mathbb{P}$, muni de l'addition et de la multiplication modulo p.

0.2.4 Algèbre linéaire

Comme le corps, l'espace vectoriel est lui aussi une structure algébrique définie par un ensemble et deux opérations. La différence majeure est qu'une de ses opérations est une loi de composition externe, c'est-à-dire qu'un de ses deux arguments provient d'une structure algébrique sous-jacente. Cette structure est un corps dont les éléments sont nommés *scalaires*.

Un **espace vectoriel** sur un corps $\mathbb{K}$ est un ensemble V dont les éléments sont nommés *vecteurs* et qui est muni des opérations + (somme vectorielle) et $\cdot$ (multiplication par un scalaire). Nous verrons occasionnellement un vecteur comme un n-uplet de scalaires : $\mathbf{v} = (v_1, v_2, ..., v_n)$.

L'ensemble des vecteurs V forme un groupe abélien sous l'addition, ce qui signifie que :

— L'opération + est une loi de composition interne, i.e. elle est du type $V \times V \to V$;

— L'opération + est associative ;

— L'opération + possède un élément neutre, le vecteur nul, noté $\mathbf{0}$;

— Chaque élément $\mathbf{v} \in V$ a un inverse par rapport à +, noté $-\mathbf{v}$;

— L'opération + est commutative.

L'opération $\cdot$ est une loi de composition externe faisant intervenir un scalaire, c'est-à-dire un élément du corps $\mathbb{K}$. Elle possède les propriétés suivantes :

— L'opération $\cdot$ est une loi de composition externe du type $\mathbb{K} \times V \to V$;

— L'opération $\cdot$ est distributive à gauche par rapport à $+$, i.e. $\forall a \in \mathbb{K}, \mathbf{u}, \mathbf{v} \in V, a \cdot (\mathbf{u}+\mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v}$;

— L'opération $\cdot$ est distributive à droite par rapport à l'opération d'addition de $\mathbb{K}$, i.e. $\forall a, b \in \mathbb{K}, \mathbf{u} \in V, (a +_{\mathbb{K}} b) \cdot \mathbf{u} = a \cdot \mathbf{u} + b \cdot \mathbf{u}$;

— L'élément neutre de l'opération de multiplication du corps $\mathbb{K}$, noté 1, est aussi neutre à gauche pour l'opération $\cdot$, i.e. $\forall \mathbf{u} \in V, 1 \cdot \mathbf{u} = \mathbf{u}$.

Remarque. On parle de distributivité et de neutralité à gauche ou à droite, car l'opération a le type $\mathbb{K} \times V \to V$ et n'est pas commutative. Donc si $\mathbf{v}$ est un vecteur et a un scalaire, l'opération $\mathbf{v} \cdot a$ n'a simplement pas de sens dans un espace vectoriel. Notons aussi que selon sa définition, un espace vectoriel ne peut pas être vide car il contient toujours le vecteur nul.

Comme un espace vectoriel V sur $\mathbb{K}$ est fermé sous la somme vectorielle et sous la multiplication par un scalaire, il est aussi fermé sous la composition de ces opérations. En d'autres termes,

$$a_1, a_2, ..., a_n \in \mathbb{K}, \mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n} \in V \Rightarrow a_1\mathbf{v_1} + a_2\mathbf{v_2} + ... + a_n\mathbf{v_n} \in V$$

On dit alors que $\mathbf{w} = a_1\mathbf{v_1} + a_2\mathbf{v_2} + ... + a_n\mathbf{v_n}$ est une **combinaison linéaire** de $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}$.

Pour un espace au nombre de dimensions fini, il existe toujours un ensemble minimal de vecteurs $\mathcal{B} = \{\mathbf{b_1}, \mathbf{b_2}, ..., \mathbf{b_n}\}$ tel que tout vecteur dans V est une combinaison linéaire des éléments de $\mathcal{B}$. $\mathcal{B}$ est appelé la **base** de V et sa cardinalité correspond à la dimension de l'espace vectoriel.

Un **espace de Hilbert complexe** $\mathcal{H}$ est un espace vectoriel sur le corps $\mathbb{C}$ et muni d'une autre opération : le produit scalaire[4], de type $V \times V \to \mathbb{C}$ et défini ainsi :

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i^*$$

où n est la dimension de l'espace. L'astérisque indique qu'il faut prendre le conjugué complexe de v_i.

La **norme** $\|\mathbf{u}\|$ d'un vecteur $\mathbf{u}$ est obtenue en appliquant le produit scalaire sur lui-même ; elle est égale à $\sqrt{\mathbf{u} \cdot \mathbf{u}}$. Un vecteur est dit **unitaire** si cette valeur vaut 1. Deux vecteurs $\mathbf{u}, \mathbf{v}$ sont **orthogonaux** si $\mathbf{u} \cdot \mathbf{v} = 0$, et sont **orthonormaux** s'ils sont orthogonaux entre eux et que chacun est unitaire. Une **base orthonormale** est une base dont tous les éléments sont orthonormaux deux à deux.

4. Un espace de Hilbert doit aussi être *complet*, notion dont nous ne nous servirons cependant pas et ne définissons donc pas.

On peut aussi définir des opérations qui agissent sur différents espaces vectoriels. Une **transformation linéaire** $T : V \to W$ prend un vecteur de l'espace vectoriel V et le transforme en un vecteur d'un autre autre espace W, d'une façon qui préserve les combinaisons linéaires : si $\mathbf{v} = a_1\mathbf{v_1} + ... + a_n\mathbf{v_n} \in V$ alors $T(\mathbf{v}) = a_1T(\mathbf{v_1}) + ... + a_nT(\mathbf{v_n}) \in W$.

Une telle transformation peut être décrite par une **matrice**. Il s'agit d'un tableau en deux dimensions dont chaque entrée est un scalaire. Pour des bases de cardinalité m pour V et n pour W, T peut être décrite avec une matrice $\mathbf{M}_T$ de dimension $m \times n$:

$$\mathbf{M}_T = \begin{bmatrix} t_{1,1} & t_{1,2} & ... & t_{1,n} \\ t_{2,1} & t_{2,2} & ... & t_{2,n} \\ \vdots & \vdots & & \vdots \\ t_{m,1} & t_{m,2} & ... & t_{m,n} \end{bmatrix}, t_{i,j} \in \mathbb{K}\ \forall i, j$$

Une matrice peut être multipliée par un scalaire à sa gauche ainsi :

$$r\mathbf{M}_T = \begin{bmatrix} rt_{1,1} & rt_{1,2} & ... & rt_{1,n} \\ rt_{2,1} & rt_{2,2} & ... & rt_{2,n} \\ \vdots & \vdots & & \vdots \\ rt_{m,1} & rt_{m,2} & ... & rt_{m,n} \end{bmatrix}, r, t_{i,j} \in \mathbb{K}\ \forall i, j$$

Deux matrices peuvent être multipliées ensemble, à condition que la deuxième dimension de la matrice de gauche soit égale à la première de celle de droite. Soit

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & ... & a_{1,l} \\ a_{2,1} & a_{2,2} & ... & a_{2,l} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & ... & a_{n,l} \end{bmatrix}$$

de dimension $n \times l$. Alors $\mathbf{A} \cdot \mathbf{M}_T$ est indéfini si $l \neq m$, mais $\mathbf{M}_T \cdot \mathbf{A}$ donne une matrice de taille $m \times l$ où un élément à la position i, j vaut $\sum_{k=1}^{n} t_{ik}a_{kj}$. Cela vaut aussi pour les vecteurs, car ceux-ci sont un cas particulier de matrice où une dimension est de 1.

Certaines matrices sont dites carrées, signifiant que leurs deux dimensions sont les mêmes. La matrice identité,

$$\begin{bmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & ... & 1 \end{bmatrix}$$

où 0 est l'élément neutre de l'addition du corps et 1 l'élément neutre de la multiplication, est une matrice carrée agissant comme élément neutre dans la multiplication entre matrices. On la note simplement $\mathbb{1}$ (sa taille étant généralement déductible à partir du contexte).

Une matrice carrée ayant un seul 1 par rangée et par colonne, tout autre élément étant 0, est une **matrice de permutation** et a pour effet d'intervertir les éléments d'un vecteur entre eux.

Exemple 0.9 (Matrice de permutation)**.**

$$\begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} v_3 & v_4 & v_2 & v_1 \end{bmatrix}$$

Lorsque le corps sous-jacent est $\mathbb{C}$, la transposée conjuguée de $\mathbf{A}$ est

$$\mathbf{A}^\dagger = \begin{bmatrix} a^*_{1,1} & a^*_{2,1} & ... & a^*_{l,1} \\ a^*_{1,2} & a^*_{2,2} & ... & a^*_{l,2} \\ \vdots & \vdots & & \vdots \\ a^*_{1,n} & a^*_{2,n} & ... & a^*_{l,n} \end{bmatrix}$$

Par rapport à $\mathbf{A}$, on a transposé la matrice (l'élément en position i, j est celui qui était en position j, i) et on a changé chaque élément par son conjugué complexe.

On dit qu'une matrice carrée $\mathbf{U}$ est **unitaire** si

$$\mathbf{U}\mathbf{U}^\dagger = \mathbf{U}^\dagger\mathbf{U} = \mathbb{1}$$

Une telle matrice a la particularité qu'elle préserve l'unitarité des vecteurs. Si $\|\mathbf{v}\| = 1$, alors $\|\mathbf{U} \cdot \mathbf{v}\| = 1$.

Nous aurons finalement besoin de combiner des espaces vectoriels à l'aide du **produit tensoriel**, noté $\otimes$. Soient $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \in V, \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \in F$. Alors

$$\mathbf{v} \otimes \mathbf{w} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \otimes \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 w_1 \\ v_1 w_2 \\ \vdots \\ v_1 w_n \\ v_2 w_1 \\ v_2 w_2 \\ \vdots \\ v_2 w_n \\ \vdots \\ v_m w_n \end{bmatrix}$$

La dimension du **tenseur**[5] résultant est alors mn. Lorsqu'appliqué sur des espaces vectoriels, $\mathcal{B}_{V\otimes W}$ contient tous les $\mathbf{v}\otimes\mathbf{w}$ tels que $\mathbf{v}\in\mathcal{B}_V, \mathbf{w}\in\mathcal{B}_W$.

Exemple 0.10 (Produit tensoriel)**.** Soient V de dimension 2 engendré par $\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\}$ et W de dimension 3 engendré par $\{\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\}$. Alors $V\otimes W$ est engendré par

$$\{\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}0\\0\\1\end{bmatrix}\}$$

ou

$$\{\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}\}$$

Toute combinaison linéaire de cette base fait ainsi partie de $V\otimes W$.

Remarquons que tout produit tensoriel entre un vecteur de V et un de W peut être décrit facilement comme un vecteur de $V\otimes W$. Par exemple

$$\begin{bmatrix}5\\3\end{bmatrix}\otimes\begin{bmatrix}-1\\4\\0\end{bmatrix} = \begin{bmatrix}-5\\20\\0\\-3\\12\\0\end{bmatrix} = -5\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix} + 20\begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix} - 3\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix} + 12\begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix}$$

On dit alors que ce tenseur est **séparable**, voulant dire qu'on peut le décrire en prenant V et W séparément. En effet, il se factorise bien en le produit de $\begin{bmatrix}5\\3\end{bmatrix}$ et $\begin{bmatrix}-1\\4\\0\end{bmatrix}$:

$$-5(\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}1\\0\\0\end{bmatrix}) + 20(\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}0\\1\\0\end{bmatrix}) - 3(\begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}1\\0\\0\end{bmatrix}) + 12(\begin{bmatrix}0\\1\end{bmatrix}\otimes\begin{bmatrix}0\\1\\0\end{bmatrix})$$

5. On lui donne un nouveau nom, même s'il s'agit toujours d'un vecteur simplement dans un plus grand espace.

$$= 5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes (- \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}) + 3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes (- \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix})$$

$$= (5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 1 \end{bmatrix}) \otimes (- \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix})$$

$$= \begin{bmatrix} 5 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} -1 \\ 4 \\ 0 \end{bmatrix}$$

L'inverse n'est pas toujours vrai : une combinaison linéaire dans $V \otimes W$ n'est pas forcément descriptible à l'aide de V et W isolément. Le tenseur suivant est **non séparable** :

$$\begin{bmatrix} 0 \\ 2 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 2(\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}) + (\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}) + (\begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix})$$

Si on tente de faire une mise en évidence, on obtient seulement

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes (2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Cela ne se simplifie pas davantage. Ce tenseur existe donc uniquement dans $V \otimes W$, et non séparément dans V et W.

En terminant, notons que le produit tensoriel se généralise à des matrices : Soient

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{m,1} & a_{m,2} & ... & a_{m,n} \end{bmatrix}$$

de taille $m \times n$ et

$$\mathbf{B} = \begin{bmatrix} b_{1,1} & b_{1,2} & ... & b_{1,l} \\ b_{2,1} & b_{2,2} & ... & b_{2,l} \\ \vdots & \vdots & & \vdots \\ b_{k,1} & b_{k,2} & ... & b_{k,l} \end{bmatrix}$$

de taille $k \times l$. Alors $\mathbf{A} \otimes \mathbf{B}$ est la matrice de taille $mk \times nl$ suivante :

$$\begin{bmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,n}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{m,1}\mathbf{B} & a_{m,2}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}$$

Horizon I :
L'époque contemporaine

Au moment d'écrire ces lignes, nous sommes en **2018**. Depuis une vingtaine d'années, nous utilisons l'Internet à outrance, toujours en assumant que nos communications sont privées, c'est-à-dire à l'abri d'espions qui voudraient lire ou modifier nos propos (ou nos transactions bancaires). C'est généralement avec raison que nous pouvons avoir confiance en la confidentialité et l'intégrité des données que nous partageons sur le réseau, car avant de les envoyer, nos ordinateurs effectuent des opérations de *chiffrement*, où les données sont transformées en un charabia que seul le destinataire légitime sera en mesure de comprendre.

Les algorithmes utilisés pour assurer une communication sécurisée ont très peu évolué depuis les débuts de l'Internet. À titre d'exemple, le protocole de chiffrement RSA a été inventé en 1977 et demeure aujourd'hui l'un des algorithmes de chiffrement les plus répandus. Pourquoi changerions-nous une recette gagnante, sachant que RSA permet des opérations très efficaces, est conceptuellement simple et, surtout, est encore sécuritaire ?

Dans cet horizon, qui représente la réalité actuelle, nous explorons en détail sur quoi repose la sécurité des protocoles cryptographiques contemporains que nous prenons pour acquise depuis maintenant plusieurs années.

1.1 Bases de la cryptographie

1.1.1 Chiffrement

La cryptographie, l'étude de la communication sécurisée, est avant toute chose basée sur le concept de chiffrement. Supposons qu'Alice veuille envoyer un message confidentiel en français à Bob via Internet. Pour qu'aucune oreille indiscrète ne l'intercepte, il lui faut traduire le message dans un langage que seul Bob est en mesure de comprendre. À des fins pratiques, ce *langage* que Bob est le seul à connaître est généralement la langue de communication standard, par exemple le français, représentée en langage informatique, donc en un nombre, puis transformée par une opération mathématique secrète.

L'acte de chiffrer consiste donc à transformer un *message clair* m, c'est-à-dire écrit dans une langue compréhensible, en un message inintelligible $E(m)$ appelé *message chiffré*, grâce à une *fonction de chiffrement* E. Ce message chiffré peut alors être communiqué sans crainte d'être intercepté, mais doit permettre au récipiendaire légitime de retrouver le message original m avec une *fonction de déchiffrement* D, où $D(E(m)) = m$.

Jusqu'ici, pour que la cryptographie fonctionne, il faudrait que chaque paire d'individus partage secrètement des algorithmes de chiffrement et de déchiffrement uniques. La sécurité d'une telle méthode, dite *sécurité par l'obscurité*, repose sur l'hypothèse que les adversaires ne connaissent pas l'algorithme utilisé et n'ont donc aucune idée de la structure du message chiffré. Comme cela demanderait d'avoir des millions d'algorithmes différents, cette vision est généralement jugée non viable.

Plutôt que de recourir à des algorithmes non divulgables, il est préférable d'utiliser des algorithmes bien connus mais faisant usage d'un secret, ou *clé*, pouvant changer d'une communication à l'autre. De cette façon, les diverses techniques de chiffrement et déchiffrement, qu'on appellera *cryptosystèmes*, présentent plusieurs avantages : ils peuvent être étudiés par la communauté scientifique internationale, ils peuvent être réutilisés à volonté simplement en changeant la clé, mais surtout, leur sécurité peut être quantifiée de façon formelle, grâce aux théories de l'information et de la complexité.

Définition 1.1. Un **cryptosystème** est la formalisation d'une méthode de chiffrement et est mathématiquement défini comme un tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, où :

- $\mathcal{M}$ est l'ensemble des messages clairs possibles ;
- $\mathcal{C}$ est l'ensemble des messages chiffrés possibles ;
- $\mathcal{K}$ est l'ensemble des clés possibles ;
- $\mathcal{E}$ est une famille de fonctions de chiffrement, où $\forall k \in \mathcal{K}, \exists E_k \in \mathcal{E}$ tel que $E_k : \mathcal{M} \to \mathcal{C}$;

Alice	(Ève)	Bob
Au départ : $m \in \mathcal{M}$		
	$\leftarrow k \rightarrow$	
$c = E_k(m)$		
	$c \rightarrow$	
		$m = D_k(c)$

Figure 1.1 – Cryptosystème symétrique utilisé par Alice (émettrice) et Bob (récepteur). Alice et Bob doivent avant tout s'entendre secrètement sur une clé k.

Alice	(Ève)	Bob
Au départ : $m \in \mathcal{M}$		Au départ : $k_{priv} \in \mathcal{K}_{priv}, k_{pub} \in \mathcal{K}_{pub}$
	$\leftarrow k_{pub}$	
$c = E_{k_{pub}}(m)$		
	$c \rightarrow$	
		$D_{k_{priv}}(c) = m$

Figure 1.2 – Cryptosystème asymétrique utilisé par Alice (émettrice) et Bob (récepteur). Ils n'ont pas à échanger de clé secrète au préalable.

— $\mathcal{D}$ est une famille de fonctions de déchiffrement, où $\forall k \in \mathcal{K}, \exists D_k \in \mathcal{D}$ tel que $D_k : \mathcal{C} \rightarrow \mathcal{M}$ [14].

Historiquement, les premières techniques étudiées en cryptographie étaient des méthodes de *chiffrement symétrique* (figure 1.1), où Alice et Bob partagent une clé commune pour chiffrer et déchiffrer un message.

Définition 1.2. Un **cryptosystème symétrique** est un cryptosystème tel que $\forall k \in \mathcal{K}, D_k = E_k^{-1}$.

Partager un secret au préalable est néanmoins impraticable dans certains contextes, par exemple pour envoyer un numéro de carte de crédit à un site web pour la première fois. Mais grâce à une percée réalisée dans les années 1970 [19], il est devenu possible pour deux entités de partager de l'information de façon confidentielle sans disposer d'un secret commun avant la communication. Dans le *chiffrement asymétrique* (figure 1.2), le récepteur génère deux clés, l'une qu'il garde privée et l'autre qu'il publie. Un émetteur peut ainsi chiffrer des messages avec la clé publique, de façon à ce que seul le détenteur de la clé privée puisse les déchiffrer.

Définition 1.3. Un **cryptosystème asymétrique** est un cryptosystème avec $\mathcal{K} = \mathcal{K}_{pub} \cup \mathcal{K}_{priv}$, où $\forall k_{pub} \in \mathcal{K}_{pub}, \exists k_{priv} \in \mathcal{K}_{priv}$ tel que $D_{k_{priv}} = E_{k_{pub}}^{-1}$.

Pour qu'un cryptosystème soit sécuritaire, il doit être impossible pour une espionne –appelons-la Ève– de découvrir les informations privées à partir des données échangées sur le fil, i.e. de

cryptanalyser le système. Comme la connaissance de la clé secrète implique celle du message grâce à l'algorithme de déchiffrement, on peut définir la sécurité d'un cryptosystème ainsi :

Définition 1.4. Un **cryptosystème symétrique est sécuritaire** s'il est impossible de déduire m connaissant $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ et $E_k(m)$.

Définition 1.5. Un **cryptosystème asymétrique est sécuritaire** s'il est impossible de déduire m connaissant $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, $E_{k_{pub}}(m)$ et k_{pub}.

Remarque. Il s'agit d'une notion de sécurité parmi plusieurs, mais nous choisissons celle-ci car elle fait peu d'hypothèses sur les capacités d'Ève.

D'une part, il peut être impossible de déduire le message car l'adversaire manque d'*information* par rapport à la clé. Dans ce cas, plusieurs clés peuvent mener à des messages clairs différents mais équiprobables et l'attaquant n'a aucun moyen de distinguer le bon message des autres messages possibles. D'autre part, l'attaquant peut avoir suffisamment d'information pour qu'un seul message clair soit possible, mais être incapable de déduire ce message *en un temps raisonnable*.

1.1.2 Sécurité inconditionnelle

Un chiffrement est inconditionnellement sécuritaire selon la théorie de l'information si, même en disposant d'un temps infini, l'espionne ne peut deviner le message ayant été chiffré, car plusieurs messages clairs sont possibles et elle manque de données pour distinguer le bon.

À la base, certains messages ont de plus grandes chances d'être communiqués que d'autres. Par exemple, un vrai mot comme *crypto* est plus probable que la séquence de lettres aléatoire *gtsnvj*. En d'autres termes, si M est une variable aléatoire suivant la distribution des messages possibles, $\Pr(M = \text{“}crypto\text{”}) > \Pr(M = \text{“}gtsnvj\text{”})$. Cette distribution de probabilité dite *a priori* est inhérente au langage naturel et un attaquant disposera nécessairement au moins de ces informations. S'il est possible que la valeur c du message chiffré C révèle de l'information sur M (i.e. rende certains messages clairs plus probables), il est toutefois impossible de rendre le message plus incertain que dicté par la probabilité a priori :

$$\forall m \in \mathcal{M}, c \in \mathcal{C}, \quad \Pr(M = m | C = c) \leq \Pr(M = m)$$

En meilleur cas, la connaissance de c ne devrait pas modifier la distribution de probabilités de M. On dit d'un cryptosystème atteignant ce critère qu'il fournit le secret parfait.

Définition 1.6. Un cryptosystème fournit le **secret parfait** lorsque les textes chiffrés ne présentent aucune information supplémentaire à l'attaquant [63] :

$$\forall m \in \mathcal{M}, c \in \mathcal{C}, \quad \Pr(M = m | C = c) = \Pr(M = m)$$

On peut facilement voir que cette équation a lieu lorsque M et C sont des variables indépendantes :

$$\begin{aligned}\Pr(M = m|C = c) &= \frac{\Pr(M = m \cap C = c)}{\Pr(C = c)} \\ &= \frac{\Pr(M = m)\Pr(C = c)}{\Pr(C = c)} \\ &= \Pr(M = m)\end{aligned}$$

Le *masque jetable* est un exemple de cryptosystème où le message chiffré est toujours indépendant du message clair ; il fournit donc le secret parfait.

Définition 1.7. Le **masque jetable** est un cryptosystème symétrique tel que

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- $E_k(x) = D_k(x) = x \oplus k$, où $\oplus$ représente le *ou exclusif* bit à bit.
- Chaque clé k est sélectionnée avec probabilité $\frac{1}{|\mathcal{K}|} = \frac{1}{2^n}$ [29].

Exemple 1.8.

- Supposons qu'Alice veuille envoyer la chaîne binaire $m = 01110001$ à Bob et qu'ils partagent la clé $k = 11001001$. Elle envoie à Bob la chaîne $c = m \oplus k = 01110001 \oplus 11001001 = 10111000$.
- Bob retrouve le message $m = c \oplus k = 10111000 \oplus 11001001 = 01110001$.

Théorème 1.9. *Le masque jetable fournit le secret parfait.*

Démonstration. Soit M, C et K des variables aléatoires représentant respectivement le message clair, le message chiffré et la clé. On montre que $\forall m \in \mathcal{M}, c \in \mathcal{C}, \Pr(M = m|C = c) = \Pr(M = m)$.

- Par l'utilisation du *ou exclusif*, on sait que pour une paire $(m \in \mathcal{M}, c \in \mathcal{C})$, il existe toujours une unique clé k telle que $c = m \oplus k$. Donc $\Pr(C = c|M = m) = \Pr(K = m \oplus c) = \frac{1}{2^n}$ car toute clé est obtenue avec probabilité $\frac{1}{2^n}$.
- On a que $\Pr(C = c) = \sum_{k\in\mathcal{K}} \Pr(K = k)\Pr(M = c \oplus k) = \sum_{k\in\mathcal{K}} \frac{1}{2^n}\Pr(M = c \oplus k)$. Puisque pour une paire (m, c), il existe toujours une unique clé k telle que $m = c \oplus k$, on a que $\sum_{k\in\mathcal{K}} \Pr(M = c \oplus k) = 1$, et ainsi $\Pr(C = c) = \frac{1}{2^n}$.
- Par la formule de Bayes, $\Pr(M = m|C = c) = \frac{\Pr(C=c|M=m)\Pr(M=m)}{\Pr(C=c)} = \frac{\frac{1}{2^n}\Pr(M=m)}{\frac{1}{2^n}} = \Pr(M = m)$.

Une démonstration plus détaillée est présentée dans [29]. □

Le masque jetable vient malheureusement avec un coût qui le rend très difficile à utiliser en pratique : la clé doit être aussi longue que le message et doit être transmise via un canal sécuritaire. Or, pour une majorité d'applications, s'il était possible de transmettre une clé de n bits de façon sécuritaire, il aurait été plus simple de transmettre directement le message de n bits par ce canal. De plus, il est montré dans [63] que, si l'on souhaite obtenir le secret parfait, alors il est toujours obligatoire que la taille de la clé soit égale à celle du message. C'est pourquoi il convient de s'intéresser à des cryptosystèmes symétriques qui ne fournissent pas le secret parfait.

Dans ces systèmes, l'obtention d'un message chiffré par un attaquant révèle généralement un peu d'information sur le message clair. Dans ce cas, la distribution de probabilité *a posteriori* favorise certains messages par rapport à d'autres et la distribution de probabilité du message clair M, sachant la valeur c du message chiffré C, devient inférieure à la distribution a priori du message, i.e. $\Pr(M = m|C = c) < \Pr(M = m)$. L'obtention de plusieurs couples de messages clairs et chiffrés peut alors gravement faire chuter l'incertitude par rapport à la clé.

La conception de cryptosystèmes symétriques à la fois sécuritaires et utilisables est un domaine de recherche en soi ; pour la suite nous assumerons que de tels systèmes existent et nous nous concentrerons sur la cryptographie asymétrique.

1.1.3 Sécurité calculatoire

Dans un cryptosystème asymétrique, afin d'éviter une ambiguïté lors du déchiffrement, une clé publique donnée est associée à une et une seule clé privée. Les conséquences de ce lien étroit entre clés publique et privée semblent graves : comme l'espionne connaît la clé publique, elle détient suffisamment d'informations pour qu'il n'y ait aucun doute sur la clé privée utilisée. Cependant, cette clé lui échappe malgré tout, car le passage de ces informations à la clé est un problème trop difficile au sens de la complexité de calcul.

Les problèmes algorithmiques complexes (section 0.1) ont cette particularité qu'à partir de l'entrée du problème, il n'y a qu'une seule réponse possible, et pourtant celle-ci demeure malgré tout hors de portée. Typiquement, un cryptosystème asymétrique est bâti à partir d'un tel problème algorithmique : la clé publique est l'entrée du problème et la clé privée en est la sortie. L'idée est que la découverte de la clé privée impliquerait la résolution d'un problème supposé trop difficile et serait donc impossible. L'*hypothèse calculatoire* du cryptosystème est la supposition que le problème algorithmique sous-jacent est impossible à résoudre efficacement.

1.2 Hypothèses calculatoires

1.2.1 NP \ BPP

En cryptographie, un problème est suffisamment difficile s'il est impossible, ou du moins très peu probable, d'obtenir sa solution en un temps raisonnable. Nous avons établi précédemment que, en algorithmique, *temps raisonnable* signifie *temps polynomial*, donc il est évident qu'aucun élément de P n'est considéré difficile. On ne peut toutefois ignorer la possibilité qu'un attaquant emploie un algorithme probabiliste polynomial, par exemple un algorithme BPP, pour résoudre le problème.

Définition 1.10. La classe de complexité BPP (pour *Bounded-error Probabilistic Polynomial time*) contient les problèmes pour lesquels il existe un algorithme probabiliste *polynomial* dont la sortie est correcte avec une probabilité d'au moins $\frac{2}{3}$ [26].

La particularité de BPP est que la probabilité d'obtenir la bonne réponse est une constante strictement plus grande que $\frac{1}{2}$. En fait, le choix de $\frac{2}{3}$ est totalement arbitraire. Comme l'algorithme s'exécute en temps polynomial, on peut répéter son exécution un nombre polynomial de fois tout en restant polynomial. Chaque exécution augmente considérablement les probabilités d'obtenir la bonne réponse globalement, jusqu'à un pourcentage de confiance aussi près que l'on souhaite de 100% [27]. Il va sans dire qu'un problème admettant un tel algorithme ne peut être considéré comme difficile.

Heureusement pour la cryptographie, la notion de calcul efficace s'arrête ici. En effet, la classe BPP est considérée par les experts comme la classe contenant tous les problèmes résolubles efficacement par un ordinateur classique [2] [6]. En conséquence, tout problème qui n'est pas dans BPP est intéressant dans la perspective de bâtir un cryptosystème résistant aux attaques par un ordinateur standard.

Ceci dit, si le déchiffrement illégitime doit être difficile, le problème inverse qu'est le chiffrement doit pour sa part admettre un algorithme déterministe polynomial, c'est-à-dire être dans P. Ainsi, le problème de calcul difficile est nécessairement dans NP [19] : en effet, rappelons qu'un problème est dans NP si sa solution est vérifiable en temps polynomial. Or, on peut vérifier qu'un message a été bien déchiffré en exécutant l'algorithme efficace de chiffrement sur la solution obtenue pour vérifier que cela donne le même message chiffré. Cela nous mène à l'énoncé suivant :

Énoncé 1.11. Tout problème algorithmique servant d'hypothèse calculatoire doit se trouver dans NP \ BPP.

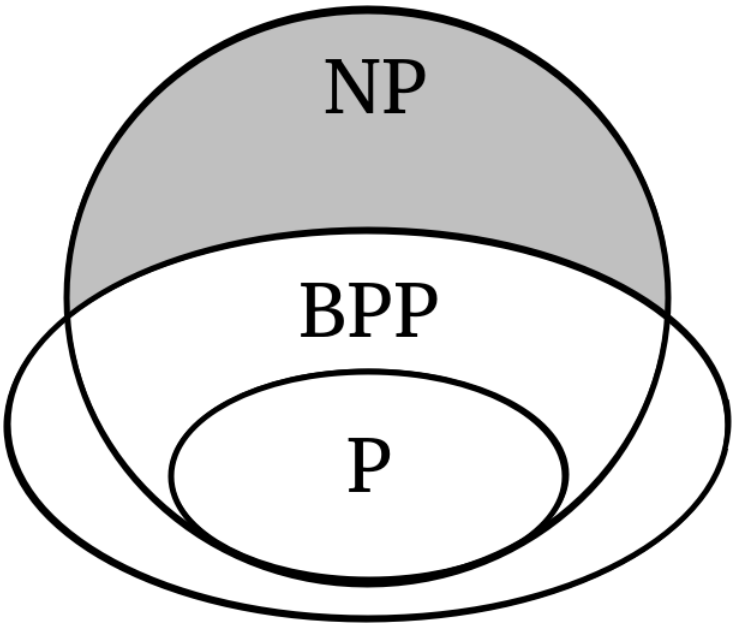


Figure 1.3 – Relations entre les classes P, BPP et NP avec en gris la classe NP \ BPP.

1.2.2 Fonction à sens unique

Les problèmes formant les classes de complexité usuelles sont des problèmes décisionnels, c'est-à-dire que leur question se répond par *oui* ou par *non*. Dans NP, bien souvent, le problème décisionnel consiste à découvrir *s'il existe* une solution. Cependant, pour des classes permettant un temps de calcul supérieur à la taille de l'entrée, comme c'est le cas pour P, BPP et NP, passer du problème décisionnel au problème de véritablement trouver la solution prend un temps négligeable et la complexité demeure la même [32]. Nous nous permettons donc un léger abus de terminologie en disant que le calcul du résultat d'une fonction fait partie d'une classe de complexité.

Définition 1.12. Formellement, soit une fonction $f : X \to Y$ définie par extension comme suit :

$$f = \{(x_1, y_1), (x_2, y_2), ...\} \text{ où } x_i \in X, y_i \in Y \ \forall i \in \mathbb{N}$$

Alors on note

- $f \in \mathsf{P}$ pour signifier que, pour tout i, il existe un algorithme déterministe polynomial identifiant y_i étant donné x_i ;
- $f \in \mathsf{BPP}$ pour signifier que, pour tout i, il existe un algorithme probabiliste polynomial identifiant y_i étant donné x_i, avec une probabilité d'au moins $\frac{2}{3}$;
- $f \in \mathsf{NP}$ pour signifier que, pour tout i, il existe un algorithme déterministe polynomial vérifiant l'appartenance de (x_i, y_i) à f.

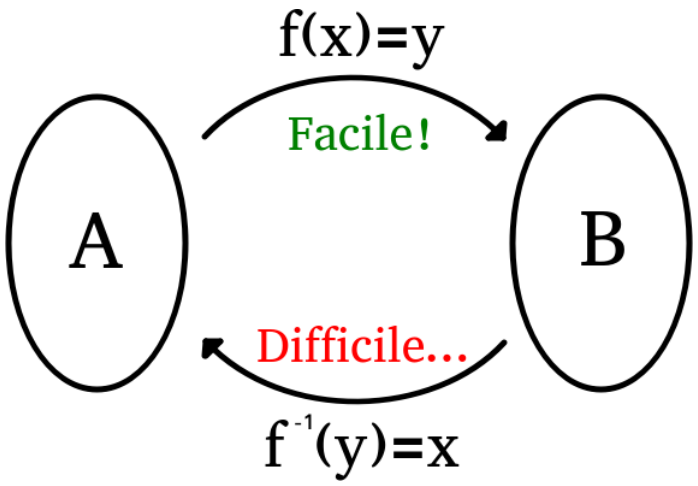


Figure 1.4 – Schéma d'une fonction à sens unique $f : A \to B$

Nous pouvons désormais définir succinctement le concept de fonction à sens unique :

Définition 1.13. Une **fonction à sens unique** f est une fonction telle que le calcul de $f(x) = y$ est dans P tandis que le calcul inverse, $f^{-1}(y) = x$ n'est pas dans BPP [5].

Quelques remarques :

- Comme $f \in \mathsf{P}$, on a par définition que $f^{-1} \in \mathsf{NP}$, et donc $f^{-1} \in \mathsf{NP} \setminus \mathsf{BPP}$;
- La définition suppose que $\mathsf{NP} \not\subseteq \mathsf{BPP}$, ce qui n'a jamais été prouvé. Si $\mathsf{NP} \subseteq \mathsf{BPP}$, les fonctions à sens unique ne peuvent exister. Pour la suite, on assumera toujours que $\mathsf{NP} \not\subseteq \mathsf{BPP}$;
- La fonction n'a pas à être injective, auquel cas il doit être difficile de trouver n'importe quel x tel que $f(x) = y$.

Cet outil permet de calculer rapidement une clé publique à partir d'une clé privée, sans que cette dernière ne soit compromise par la publication de la première. Par exemple, supposons que $k_{priv} = \mathbf{x}$, où $\mathbf{x}$ est un vecteur de n bits, et soit la fonction $f(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}$, où $\mathbf{a}$ est un vecteur de n entiers public. Le calcul de f est très simple : comme $\mathbf{x}$ est binaire, $f(\mathbf{x})$ est la somme de tous les a_i tels que $x_i = 1$.

Supposons maintenant que l'on connaisse $f(\mathbf{x})$ (et $\mathbf{a}$, qui est bien sûr public) et que l'on cherche $\mathbf{x}$. Il faut trouver un sous-ensemble des éléments de $\mathbf{a}$ dont la somme donne l'entier $\mathbf{a} \cdot \mathbf{x}$. Ce défi est exactement le problème Somme de sous-ensembles que nous avons mentionné être NP-complet à la section 0.1.4 [19].

Si l'exemple de Somme de sous-ensembles montre bien la notion de sens unique, il n'a cependant pas toutes les qualités nécessaires pour être utilisé en cryptographie. La NP-complétude garantit que le problème est trop difficile *en pire cas*, mais le simple problème de choisir un $\mathbf{x}$ qui ne mène pas à une instance facile est lui-même difficile [33].

Pour être utilisable en cryptographie, une fonction à sens unique doit être telle que le calcul de f^{-1} est un problème difficile en moyenne. La *factorisation* et le *logarithme discret* sont deux problèmes pour lesquels il semble être facile de générer des instances difficiles, c'est pourquoi ils servent souvent d'hypothèses calculatoires.

1.2.3 Factorisation

Même s'il est conceptuellement très simple, le problème de trouver les facteurs premiers d'un entier est, selon les connaissances actuelles, intraitable par nos ordinateurs. En effet, faute d'algorithme BPP connu, il semblerait que le problème de déterminer *les facteurs* d'un entier composé soit trop difficile [1] [64].

Définition 1.14.

FACTORISATION	
Entrée :	Un nombre $n \in \mathbb{N}$
Question :	Que sont les $p_1, p_2, ..., p_m \in \mathbb{P}$ tel que $\prod_{i=1}^{m} p_i = n$?

Conjecture 1.15. FACTORISATION $\notin$ BPP.

Pour garantir que l'instance de factorisation obtenue soit véritablement difficile, il convient d'utiliser deux nombres premiers p et q de taille similaire, et cette taille doit être suffisamment grande : à cette date, le plus grand entier pq à avoir été factorisé avec succès possédait 768 bits, pour des facteurs de 384 bits chacun [41].

Proposition 1.16. *En supposant la conjecture 1.15, la multiplication $f(p,q) = pq$, avec $p, q \in \mathbb{P}$, est une fonction à sens unique.*

Démonstration.

- On peut obtenir des nombres premiers de n bits en temps polynomial en n, avec probabilité très élevée, grâce à l'algorithme 7 (section 0.2.2). Un nombre de bits au-delà de 384 est donc tout à fait gérable.
- La multiplication des entiers p et q est dans P (même une simple multiplication à la main est en $\mathcal{O}(n^2)$).
- Trouver p et q à partir de pq est une instance de FACTORISATION.

□

1. Ne pas confondre avec le problème de déterminer simplement *si* un nombre est composé ou premier, qui est dans P [4].

1.2.4 Logarithme discret

Dans un groupe cyclique, un générateur est un élément g tel que tout élément du groupe peut être écrit comme g^k pour un $k \in \mathbb{N}$. Le problème du logarithme discret consiste à trouver à quelle puissance il faut élever un générateur pour obtenir un élément précis. Tout comme la factorisation, ce problème semble trop difficile pour nos ordinateurs [64].

Définition 1.17.

LOGARITHME DISCRET	
Entrée :	Un groupe cyclique G, un générateur g de G et un élément $c \in G$
Question :	Quel est l'entier k tel que $g^k = c$?

On dit d'un problème qu'il est difficile en général s'il existe des instances intraitables. Dans le cas du logarithme discret, seules certaines instances du problème sont réellement à l'abri des algorithmes polynomiaux et il est donc important de sélectionner une bonne représentation du groupe G. $\mathbb{Z}_p^*$, le groupe multiplicatif des entiers modulo un nombre premier p, est très répandu. Pour obtenir une instance du logarithme discret, il s'agit de choisir un générateur g de $\mathbb{Z}_p^*$ et le multiplier (modulo p) par lui-même k fois. Pour que le problème soit intraitable, le nombre premier p choisi doit être suffisamment grand, sachant que le record de résolution du logarithme discret est pour un nombre p de 768 bits [40]. Une autre représentation de groupe, les courbes elliptiques, sera présenté à la section 1.3.4.

La difficulté de ce problème dépend directement du groupe G sélectionné, mais comme il *existe* certains groupes pour lesquels il est en apparence difficile, le problème est, dans sa forme générale, difficile. Nous posons donc la conjecture suivante :

Conjecture 1.18. LOGARITHME DISCRET $\notin BPP$.

Proposition 1.19. *En supposant la conjecture 1.18 et l'utilisation d'un groupe dans lequel le problème est difficile, si l'opération de groupe $\cdot$ est calculable en temps polynomial en la taille de g, alors l'exponentiation $f(k) = g^k$, où g est un générateur de G et $k \in \mathbb{N}$, est une fonction à sens unique.*

Démonstration.

- Grâce à l'algorithme d'exponentiation rapide (algorithme 9, section 0.2.2), le calcul de g^k se fait en $\mathcal{O}(\log k)$, donc demeure polynomial même si k est de taille exponentielle.
- Trouver k sachant $g, g^k \in G$ est une instance du logarithme discret.

□

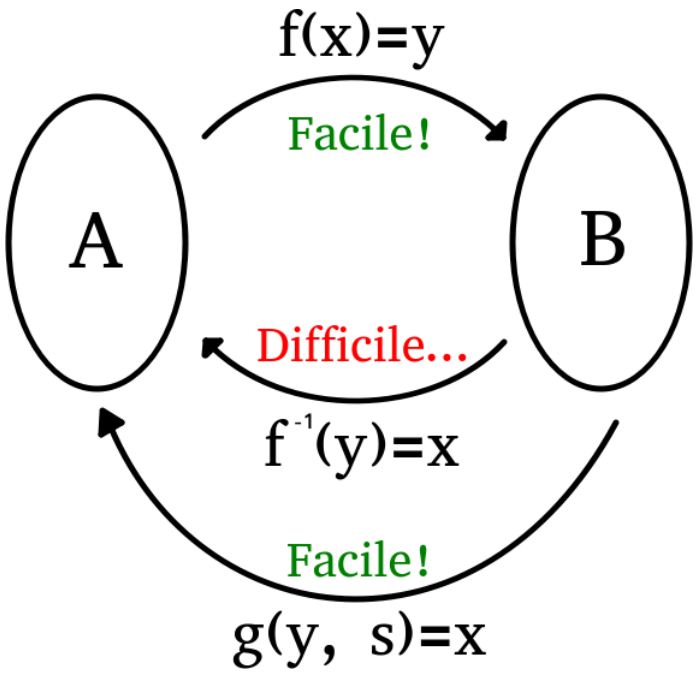


Figure 1.5 – Schéma d'une fonction à porte dérobée $f : A \to B$

1.2.5 Fonction à porte dérobée

Jusqu'ici, nous avons justifié l'utilisation de fonctions à sens unique en cryptographie asymétrique en disant que le chiffrement devrait pouvoir être exécuté de façon efficace, alors que le déchiffrement devrait être difficile afin de garantir la confidentialité du message. Mais un message chiffré de cette manière est inutile puisqu'il ne peut être lu par personne.

Contrairement aux espions, le destinataire légitime devrait être en mesure de déchiffrer le message efficacement, ce qui peut être rendu possible s'il possède une information privilégiée sur la structure du problème servant d'hypothèse calculatoire.

Définition 1.20. Une **fonction à porte dérobée** est une famille de fonctions $\{f_k\}$ ayant les particularités suivantes :

- $f_k(x) = y$, où $f_k \in \mathsf{P}$;
- $f_k^{-1}(y) = x$, où $f_k^{-1} \in \mathsf{NP} \setminus \mathsf{BPP}$;
- $\exists g, s_k$ tels que $g(y, s_k) = x$, où $g \in \mathsf{P}$.

Intuitivement, il s'agit donc d'une fonction qui est en apparence à sens unique. Par contre, il existe une façon secrète de résoudre le problème, disponible uniquement à qui connaît le secret s_k.

Exemple 1.21 (Somme de sous-ensembles)**.** Un exemple intuitif –mais qui s'est avéré non sécuritaire– basé sur Somme de sous-ensembles est présenté dans [47]. Rappelons-nous que la fonction à sens unique pour ce problème consiste en un produit scalaire entre le vecteur binaire de n éléments en entrée et un vecteur d'entiers fixe $\mathbf{a} = (a_1, a_2, ..., a_n)$. La porte dérobée vient du fait qu'une certaine structure est imposée audit vecteur fixe.

Lorsque les éléments de $\mathbf{a}$ sont en ordre *super-croissant*, c'est-à-dire lorsque $a_i > \sum_{j=1}^{i-1} a_j$, l'algorithme polynomial 10 mène à la bonne solution et il s'agit donc d'une instance facile de Somme de sous-ensembles.

Algorithme 10 Somme de sous-ensembles avec super-croissance

Entrée : Un vecteur $\mathbf{a} = (a_1, a_2, ...a_n)$ avec $a_i > \sum_{j=1}^{i-1} a_j$ et un entier w
$i := n$, $S := \emptyset$, $w' := w$
tant que $i > 0$ **faire**
    si $a_i < w'$ **alors**
        $S := S \cup \{i\}$
        $w' := w' - a_i$
    fin si
    $i := i - 1$
fin tant que
Retourner S

Évidemment, de cette façon, le problème est facile pour tous. L'idée est de transformer $\mathbf{a}$ de façon à ce qu'il s'agisse en apparence d'une instance générique, tout en gardant la possibilité d'appliquer la transformation inverse, laquelle doit demeurer secrète. En sélectionnant un nombre premier $p > \sum_{i=1}^{n} a_i$ et un entier $t < p$, on peut calculer $\mathbf{a'} = (a_1 t \mod p, a_2 t \mod p, ..., a_n t \mod p)$, un vecteur qui semble contenir des éléments de $\mathbb{Z}_p$ tirés aléatoirement.

La fonction à sens unique est $f(\mathbf{x}) = \mathbf{a'} \cdot \mathbf{x} = \sum_{i=1}^{n} x_i a_i t \mod p$ et est ainsi dans P. Comme Ève ne peut reconnaître la structure de $\mathbf{a'}$, elle semble faire face à un problème tiré de $\mathsf{NP} \setminus \mathsf{BPP}$. Toutefois, Bob, qui connaît la porte dérobée t, peut résoudre le problème efficacement grâce à l'algorithme 11.

Algorithme 11 Somme de sous-ensembles avec super-croissance cachée

Entrée : Un nombre premier p, un entier $t < p$, un vecteur $\mathbf{a'} = (a'_1, a'_2, ...a'_n)$ avec $a'_i t^{-1} \mod p > \sum_{j=1}^{i-1} (a'_j t^{-1} \mod p)$ et un entier w
Calculer $t^{-1} \mod p$
Calculer $\mathbf{a} = (a'_1 t^{-1} \mod p, a'_2 t^{-1} \mod p, ..., a'_n t^{-1} \mod p) = (a_1, a_2, ..., a_n)$
Retourner Algorithme 10 $(\mathbf{a}, w)$

Rappelons que cette fonction laisse place à une cryptanalyse efficace [62] et n'est donc pas une véritable fonction à porte dérobée. Pour la suite, nous présenterons plusieurs fonctions de ce

type qui résistent toujours à la cryptanalyse, car à partir de toute véritable fonction à porte dérobée, il est possible de définir un protocole asymétrique sécuritaire [68].

1.3 Cryptosystèmes asymétriques

Avec la notion de porte dérobée, nous pouvons définir formellement le type de cryptosystème asymétrique que nous utiliserons par la suite.

Définition 1.22. Un **cryptosystème asymétrique à porte dérobée** est un tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, \mathcal{D})$, où :

- $\mathcal{M}$ est l'ensemble des messages clairs et $\mathcal{C}$ l'ensemble des messages chiffrés ;
- $\mathcal{K}_{pub}$ est l'ensemble des clés publiques et $\mathcal{K}_{priv}$ l'ensemble des clés privées ;
- $gen : \mathcal{K}_{priv} \to \mathcal{K}_{pub}$ est une fonction à sens unique ;
- $\mathcal{E} = \{E_{k_{pub}}\}$ est une famille de fonctions de chiffrement telle que $\forall k_{pub}$,

$$E_{k_{pub}}(m) = c \text{ et } E_{k_{pub}} \in \mathsf{P}$$

$$E^{-1}_{k_{pub}}(c) = m \text{ et } E^{-1}_{k_{pub}} \in \mathsf{NP} \setminus \mathsf{BPP}$$

- $\mathcal{D} = \{D_{k_{pub}}\}$ est une famille de fonctions de déchiffrement telle que $\forall k_{priv}$,

$$D_{gen(k_{priv})}(E_{gen(k_{priv})}(m), k_{priv}) = m \text{ et } D_{gen(k_{priv})} \in \mathsf{P}$$

En soi, un cryptosystème asymétrique permet à Alice d'envoyer à Bob un message m de façon à ce qu'Ève ne puisse le comprendre. La cryptographie moderne va toutefois bien au-delà du simple chiffrement ; sont étudiées de nos jours une foule de situations où diverses parties doivent communiquer ensemble pour atteindre un but commun, mais où certaines parties sont potentiellement malveillantes. Un protocole cryptographique est une série d'actions qu'entreprennent les parties pour arriver à ce but. Ces protocoles, parfois très complexes, font généralement appel à des sous-protocoles plus simples ou directement à des algorithmes de chiffrement, qu'on appelle alors *primitives cryptographiques*.

Si les cryptosystèmes asymétriques sont peu utilisés isolément, la majorité des protocoles de sécurité plus ou moins complexes utilisés de nos jours en font intervenir régulièrement. La signature électronique, le calcul multipartite ou même le *blockchain* sont tous des exemples d'applications cryptographiques qui dépendent de la sûreté du chiffrement asymétrique car elles en utilisent comme primitive.

Ainsi, avant de présenter les cryptosystèmes les plus répandus, voyons comment on peut analyser leur sécurité.

1.3.1 Modèle de l'attaquant

Supposons le scénario typique de communication chiffrée d'Alice à Bob présenté à la figure 1.6. Alice et Bob sont des machines polynomiales, donc pouvant calculer des fonctions de P.

Alice	(Ève)	Bob
Au départ : m		Au départ : k_{priv} **Génération** de k_{pub}
	$\leftarrow k_{pub}$	
Chiffrement de m en c avec k_{pub}		
	$c \rightarrow$	
		Déchiffrement de c en m avec k_{priv}

Figure 1.6 – Exécution d'un protocole de chiffrement asymétrique par Alice (émettrice) et Bob (récepteur)

On suppose aussi qu'ils ont accès à un oracle aléatoire, un modèle répandu en cryptographie théorique et permettant de générer des nombres de façon parfaitement aléatoire [7].

Notation. Nous utiliserons régulièrement la notation $x \in_{\mathcal{U}} X$ pour signifier que l'élément x est tiré aléatoirement dans X suivant une distribution *uniforme* sur les éléments de X.

Dans ce scénario, l'objectif de l'espionne Ève est de découvrir la valeur du message m correspondant au message chiffré c qu'elle observe. Notons que, puisque $D(c, k_{priv}) = m$, la découverte de la clé k_{priv} est parmi les moyens d'arriver à cette fin.

Définition 1.23. Une **attaque est réussie** si Ève découvre la valeur de m ou k_{priv} à partir des valeurs interceptées k_{pub} et $E_{k_{pub}}(m)$ et du système connu $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, \mathcal{D})$.

Deux variables majeures peuvent être modulées pour évaluer le niveau de sécurité d'un protocole cryptographique : le modèle de calcul de l'adversaire et son mode d'interaction.

Modèle de calcul : Tout d'abord, un adversaire peut évidemment calculer uniquement ce que l'ordinateur qu'il utilise lui permet. Il s'agit donc d'identifier la classe de complexité correspondant aux problèmes que l'adversaire peut résoudre. Pour les protocoles présentés concernant la cryptographie contemporaine (horizon I), on assumera bien sûr que l'adversaire peut résoudre tous, mais uniquement, les problèmes de BPP. Dans les horizons II et III, toutefois, nous analyserons les cryptosystèmes en suivant une toute autre hypothèse.

Notons que si on a besoin de supposer que la puissance de calcul de l'adversaire est bornée pour montrer qu'un cryptosystème est sécuritaire, cela mène à un résultat *moins fort* que si on suppose une puissance de calcul infinie, comme c'est le cas pour la sécurité inconditionnelle (section 1.1.2). Une telle hypothèse est toutefois obligatoire pour arriver à des cryptosystèmes asymétriques sécuritaires utilisables avec des ordinateurs classiques.

Mode d'interaction : Dans le scénario que nous analysons, la présence d'Ève n'a aucun impact sur l'interaction d'Alice et Bob, elle ne fait qu'écouter les informations et exécuter

des calculs de son côté. Il s'agit ainsi d'une simple *oreille indiscrète*, ou adversaire *passif*, par opposition à un adversaire *actif* qui pourrait s'insérer entre Alice et Bob et corrompre les données passant sur le fil à son avantage.

Un adversaire actif a évidemment davantage de possibilités envisageables pour arriver à une attaque réussie, incluant celles d'un adversaire passif, donc prouver la sécurité dans ce cadre d'analyse serait un résultat bien plus fort. Par contre, la proposition converse nous indique que si l'on peut montrer qu'un protocole *ne résiste pas* à un adversaire passif, c'est qu'il ne résiste pas non plus à un adversaire actif.

Les cryptosystèmes asymétriques des prochaines sections sont les plus répandus dans le monde. À ce titre, ils sont conçus (généralement avec de mineures adaptations) pour être résistants à des adversaires BPP *actifs*, un résultat bien plus fort que ce que nous allons montrer. Ceci dit, à l'horizon II, nous montrerons qu'un changement du modèle de calcul permet tout à fait à un adversaire *passif* de réussir son attaque, et qu'il aurait donc été inutile, dans cette perspective, de considérer un mode d'interaction plus complexe.

1.3.2 RSA

Le protocole RSA, probablement le cryptosystème asymétrique le plus célèbre, est basé sur le fait qu'on peut multiplier deux nombres premiers entre eux facilement, mais qu'une fois multipliés il est trop difficile de retrouver les nombres initiaux.

Définition 1.24. RSA est un cryptosystème asymétrique à porte dérobée défini par le tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

- $\mathcal{K}_{priv} = \mathbb{P} \times \mathbb{P}$;
- $\mathcal{K}_{pub} = \mathbb{N} \times \mathbb{N}$;
- $gen((p,q)) = (n,e)$, où $n = pq$, $e < \varphi(n)$ et $pgcd(e, \varphi(n)) = 1$;
- $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n$;
- $\mathcal{E} = \{E_{(n,e)} | E_{(n,e)}(m) = m^e \mod n\}$;
- $\mathcal{D} = \{D_{(n,e)} | D_{(n,e)}(m^e \mod n, (p,q)) = m\}$.

Les fonctions gen, E et D sont respectivement calculées à l'aide des algorithmes 12 (exécuté par Bob), 13 (Alice) et 14 (Bob) ci-dessous. Ces algorithmes et les exemples proviennent de l'article original de RSA [60].

Exemple 1.25 (Utilisation de RSA)**.**

- Supposons la clé privée $(p = 47, q = 59)$, ce qui implique $n = pq = 47 \cdot 59 = 2773$ et $\varphi(n) = (p-1)(q-1) = 46 \cdot 58 = 2668$. En prenant $e = 17$, qui est premier et donc assurément premier avec 2668, la clé publique est $(2773, 17)$.

Algorithme 12 Génération de la clé publique RSA

1: **Entrée :** La clé privée $(p, q) \in \mathbb{P} \times \mathbb{P}$
2: Calculer $n = pq$ et $\varphi(n) = (p-1)(q-1)$
3: Sélectionner $e \in \mathbb{N}$ tel que $e < \varphi(n)$ et $pgcd(e, \varphi(n)) = 1$.
4: **Retourner** (n, e)

Algorithme 13 Chiffrement RSA

1: **Entrée :** La clé publique $(n, e) \in \mathbb{N} \times \mathbb{N}$, le message $m \in \mathbb{Z}_n$
2: **Retourner** $c = m^e \mod n$

Algorithme 14 Déchiffrement RSA

1: **Entrée :** La clé publique $(n, e) \in \mathbb{N} \times \mathbb{N}$, la clé privée $(p, q) \in \mathbb{P} \times \mathbb{P}$ et le message chiffré $c \in \mathbb{Z}_n$
2: Calculer $d = e^{-1} \mod \varphi(n) = e^{-1} \mod (p-1)(q-1)$
3: **Retourner** $m = c^d \mod n$

— Supposons que le message est $m = 920$. Alors $c = m^e \mod n = 920^{17} \mod 2773 = 948$.

— Connaissant e, p et q, on peut calculer en temps polynomial $d = 17^{-1} \mod 2668 = 157$. Le déchiffrement donne finalement $m = c^d \mod n = 948^{157} \mod 2773 = 920$.

Remarque. En pratique, la valeur de d est calculée une seule fois lors de la génération de clés et sert de clé privée. Nous avons toutefois préféré garder (p, q) comme clé privée pour mettre l'accent sur son utilisation comme porte dérobée.

Proposition 1.26. *En supposant la proposition 1.16, la génération de la clé publique RSA est une fonction à sens unique.*

Démonstration. Selon la proposition 1.16, passer de p, q à pq est déjà une fonction à sens unique. On peut générer e en temps polynomial car le calcul de $\varphi(pq)$ est trivial et le calcul du *pgcd* est polynomial (algorithme 8, section 0.2.2). Le nombre e ne divulgue cependant pas suffisamment d'information pour aider au calcul des facteurs de pq (e peut être premier et ainsi $\varphi(pq)$ peut être absolument n'importe quoi). □

Pour montrer que le chiffrement est une fonction à porte dérobée, nous avons besoin de faire l'hypothèse qu'un nouveau problème, le problème RSA, est trop difficile. Ce problème se réduit directement à la factorisation, mais pourrait s'avérer plus facile.

Définition 1.27.

Problème RSA	
Entrée :	$n, e, c \in \mathbb{N}$, où $pgcd(e, \varphi(n) = 1)$ et $n = pq$ avec $p, q \in \mathbb{P}$
Question :	Quel est l'entier m tel que $m^e \mod n = c$?

Conjecture 1.28. Problème RSA $\in \mathsf{NP} \setminus \mathsf{BPP}$ [60].

Proposition 1.29. *En supposant la conjecture 1.28, le chiffrement RSA est une fonction à porte dérobée.*

Démonstration.

- $E(m) = m^e \mod n$ est une exponentiation modulaire, pouvant être calculée par l'algorithme polynomial 9 (section 0.2.2) ;
- la conjecture 1.28 nous indique qu'il n'y a aucun algorithme polynomial pour le problème inverse ;
- le déchiffrement est polynomial grâce à la porte dérobée $(p, q) : d = e^{-1} \mod (p-1)(q-1)$ est un calcul d'inverse modulaire, calculable avec l'algorithme d'Euclide étendu, et $m = c^d \mod n$ est une autre exponentiation modulaire.

□

Théorème 1.30. Briser RSA $\leq$ Factorisation.

Démonstration. Supposons que l'on peut factoriser n en ses facteurs p et q. Alors on peut obtenir la clé privée (p, q) à partir de (pq, e), utiliser cette clé comme porte dérobée pour l'algorithme de déchiffrement et ainsi trouver m en temps polynomial. □

1.3.3 El Gamal sur $\mathbb{Z}_p^*$

Alors que RSA se réduit au problème de factorisation, le cryptosystème El Gamal est quant à lui basé sur le logarithme discret. Nous présentons d'abord sa définition sur un groupe générique, puis nous le spécialisons sur le groupe $\mathbb{Z}_p^*$ ainsi que sur une courbe elliptique.

Définition 1.31. Étant donné un groupe cyclique $G = \langle g \rangle$ d'ordre q, **El Gamal** est un cryptosystème asymétrique à porte dérobée défini par le tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

- $\mathcal{K}_{priv} = \{1, ..., q-1\}$;
- $\mathcal{K}_{pub} = G$;
- $gen(x) = g^x$;
- $\mathcal{M} = \mathcal{C} = G$;
- $\mathcal{E} = \{E_{g^x} | E_{g^x}(m) = (g^y, mg^{xy})$, où $y \in_{\mathcal{U}} \{1, ..., q-1\}\}$
- $\mathcal{D} = \{D_{g^x} | D_{g^x}((g^y, mg^{xy}), x) = m\}$ [21].

Pour $G = \mathbb{Z}_p^*$, où $p \in \mathbb{P}$, toutes les opérations sont effectuées modulo p, et on a $q = p - 1$.

Exemple 1.32 (Utilisation de El Gamal sur $\mathbb{Z}_p^*$)**.** Supposons $G = \mathbb{Z}_{17}^*$ d'ordre $q = 16$ généré par $g = 3$.

Algorithme 15 Génération de la clé publique El Gamal sur $\mathbb{Z}_p^*$

1: **Entrée :** La clé privée $x \in \{1, ..., q-1\}$ et les informations publiques $p \in \mathbb{P}$, $g \in \mathbb{Z}_p^*$
2: **Retourner** $g^x \mod p$

Algorithme 16 Chiffrement El Gamal sur $\mathbb{Z}_p^*$

1: **Entrée :** La clé publique $g^x \mod p$, le message $m \in \mathbb{Z}_p^*$ et les informations publiques $p \in \mathbb{P}$, $g \in \mathbb{Z}_p^*$
2: Sélectionner un $y \in \{1, ..., q-1\}$ suivant une distribution uniforme
3: Calculer $g^y \mod p$ et $g^{xy} \mod p = (g^x)^y \mod p$
4: **Retourner** $c = (g^y \mod p, mg^{xy} \mod p)$

Algorithme 17 Déchiffrement El Gamal sur $\mathbb{Z}_p^*$

1: **Entrée :** La clé publique $g^x \mod p$, la clé privée $x \in \{1, ..., p-2\}$, le message chiffré $c = (c_1, c_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ et les informations publiques $p \in \mathbb{P}$, $g \in \mathbb{Z}_p^*$
2: Calculer $g^{xy} \mod p = (g^y)^x \mod p = c_1^x \mod p$
3: Calculer $g^{-xy} \mod p = (g^{xy})^{-1} \mod p$
4: **Retourner** $m = mg^{xy}g^{-xy} \mod p = c_2(g^{-xy}) \mod p$

— Supposons la clé privée $x = 7$. Alors la clé publique est $k_{pub} = 3^7 \mod 17 = 11$.

— Supposons que le message est $m = 15$. On sélectionne (au hasard) $y = 9$, puis on calcule $g^y = 3^9 \mod 17 = 14$ ainsi que $m \cdot k_{pub}^y = 15 \cdot 11^9 \mod 17 = 15 \cdot 6 \mod 17 = 5$. On génère donc le message chiffré $(14, 5)$.

— On a en possession $c_1 = 14$ et $c_2 = 5$. On calcule $g^{xy} = 14^7 \mod 17 = 6$, puis son inverse $6^{-1} \mod 17 = 3$, pour finalement obtenir $m = c_2 \cdot 3 \mod 17 = 5 \cdot 3 \mod 17 = 15$, comme désiré.

Proposition 1.33. *En supposant la proposition 1.19, la génération de la clé publique El Gamal sur le groupe $\mathbb{Z}_p^*$ est une fonction à sens unique.*

Démonstration. L'exponentiation modulaire $g^x \mod p$ se fait en temps polynomial (voir l'algorithme 9). De plus, quiconque est en possession uniquement des informations publiques g et p et de la clé publique $g^x \mod p$ doit résoudre le logarithme discret (qui n'est pas dans BPP, selon la conjecture 1.18) pour trouver x. □

Encore une fois, la fonction à porte dérobée demande que l'on définisse un nouveau problème, semblable au logarithme discret mais potentiellement plus facile.

Définition 1.34.

Problème de Diffie-Hellman	
Entrée :	$g, g^x, g^y \in \mathbb{Z}_p^*$
Question :	Que vaut g^{xy} ?

Conjecture 1.35. Problème de Diffie-Hellman $\in \mathsf{NP} \setminus \mathsf{BPP}$ [19].

Proposition 1.36. *Le chiffrement El Gamal sur le groupe $\mathbb{Z}_p^*$ est une fonction à porte dérobée.*

Démonstration.

- $E(m) = (g^y \mod p, m(g^x)^y \mod p)$ consiste en deux exponentiations modulaires et une multiplication, donc son calcul est clairement polynomial ;
- pour retrouver m à partir de ces informations, il faut calculer $g^{-xy} \mod p$ et donc $g^{xy} \mod p$, connaissant toutefois $k_{pub} = g^x \mod p$ et $g^y \mod p$ ce qui revient exactement au Problème de Diffie-Hellman ;
- le déchiffrement est polynomial grâce à la porte dérobée x : $g^{xy} \mod p = (g^y)^x \mod p$ est une autre exponentiation modulaire, $(g^{-xy}) = (g^{xy})^{-1} \mod p$ est un inverse modulaire et $m = c_2(g^{-xy}) \mod p$ est une simple multiplication.

□

Théorème 1.37. Briser El Gamal sur $\mathbb{Z}_p^* \leq$ Logarithme discret.

Démonstration. Comme $\mathbb{Z}_p^*$ est un groupe cyclique, le problème de trouver x étant donné $g^x \mod p$, qui permet d'inverser la fonction à porte dérobée, est un cas particulier du Logarithme discret. La réduction découle directement de ce fait. □

1.3.4 El Gamal sur une courbe elliptique

La figure 1.7 montre en rouge l'ensemble des points $(x, y) \in \mathbb{R}^2$ correspondant à une équation bien spécifique qu'on nomme courbe elliptique. Cette équation, toujours de la forme $y^2 = x^3 + ax + b$, a la particularité que lorsqu'elle est définie sur un corps ($\mathbb{R}$ dans la figure), alors l'ensemble des points sur la courbe, munis d'une opération d'addition particulière, forme un groupe abélien. On pourra ainsi, en utilisant un corps fini comme $\mathbb{Z}_p$ plutôt que $\mathbb{R}$, se servir du logarithme discret pour bâtir un cryptosystème avec les éléments de la courbe.

Définition 1.38. Étant donné un corps $\mathbb{K}$, une **courbe elliptique** $\mathbf{E}(\mathbb{K})$ est l'ensemble des points $(x, y) \in \mathbb{K} \times \mathbb{K}$ satisfaisant l'équation [2] $y^2 = x^3 + ax + b$, pour $a, b \in \mathbb{K}$, agrémenté d'un point abstrait, le *point à l'infini* P_∞. En d'autres termes,

$$\mathbf{E}(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} \mid y^2 = x^3 + ax + b\} \cup \{P_\infty\}$$

L'opération du groupe est une addition bien particulière : elle consiste à prendre la droite entre les deux points additionnés, trouver l'autre point de la courbe elliptique passant par

2. a et b doivent être tels que $4a^3 + 27b^2 \neq 0$, condition assurant que la tangente en chaque point est bien définie.

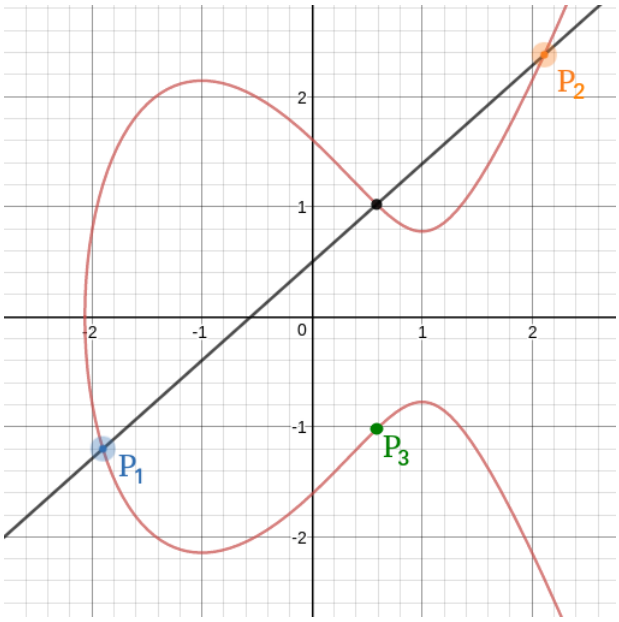


Figure 1.7 – Une courbe elliptique (en rouge) sur $\mathbb{R}$ et l'addition $P_1 + P_2 = P_3$ (image générée avec le calculateur Desmos : https://www.desmos.com/calculator/ialhd71we3)

cette droite, puis, si on dénote le point ainsi obtenu par (x, y), prendre finalement $(x, -y)$ (voir figure 1.7).

Définition 1.39. L'**addition** $+_{\mathbf{E}}$ de points $P_1 = (x_1, y_1), P2 = (x_2, y_2)$ dans une courbe elliptique est définie par les propriétés suivantes :

1. $P_1 +_{\mathbf{E}} P_\infty = P_\infty +_{\mathbf{E}} P_1 = P_1$
2. Si $x_1 = x_2$ et $y_1 = -y_2$, $P_1 +_{\mathbf{E}} P_2 = P_\infty$
3. Si $P_1 = P_2$ alors $P_1 +_{\mathbf{E}} P_2 = P_3 = (x_3, y_3)$ où

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \text{ (note : si } y_1 = 0 \text{ on est plutôt à la condition 2)}$$

4. Sinon $P_1 +_{\mathbf{E}} P_2 = P_3 = (x_3, y_3)$ où

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ (note : si } x_1 = x_2 \text{ on est aussi à la condition 2)}$$

[69]

On remarque de la première propriété que P_∞ est un élément neutre de l'addition ; de la deuxième, que P_1 et P_2 sont des inverses si $x_1 = x_2$ et $y_1 = -y_2$. Pour les propriétés 3 et 4 (figure 1.7), λ est la tangente ou pente obtenue grâce aux deux points, et le reste du calcul permet de trouver l'inverse en y de l'intersection avec la courbe.

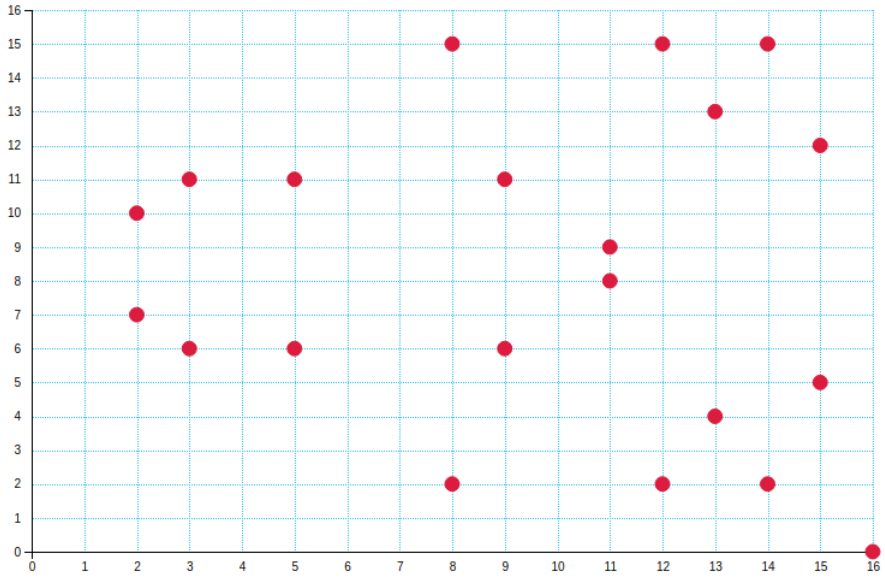


Figure 1.8 – Une courbe elliptique sur $\mathbb{Z}_{17}$ (image générée par le calculateur de Sascha Grau : http://www.graui.de/code/elliptic2/)

Théorème 1.40. *L'ensemble* $\mathbf{E}(\mathbb{K})$ *muni de l'addition dans une courbe elliptique est un groupe abélien.*

Démonstration. Comme observé, l'élément neutre est P_∞ et l'inverse de $P = (x_1, y_1)$ est $-_{\mathbf{E}}P = (x_1, -y_1)$. La commutativité est triviale car la pente entre deux points ne dépend pas de l'ordre de ces deux points. Montrer l'associativité est pour sa part beaucoup plus complexe ; nous référons le lecteur intéressé à [22]. □

Si les figures présentées jusqu'à maintenant montrent une courbe continue et lisse, c'est parce que nous avons utilisé les réels $\mathbb{R}$ comme corps sous-jacent à la courbe. En cryptographie, toutefois, nous avons besoin d'un ensemble discret et fini comme $\mathbb{Z}_p$. La figure 1.8 obtenue pour une courbe $\mathbb{Z}_{17}$ est en apparence bien différente d'une courbe sur les réels, mais comme il s'agit tout autant d'un corps, toutes les définitions sont valables et toutes les propriétés montrées demeurent.

Lorsque nous avons présenté le logarithme discret, nous avons supposé un groupe dont l'opération était la multiplication. L'application de plusieurs multiplications était donc une exponentiation et l'inverse de l'exponentiation prenait donc le nom de logarithme. Dans le cas des courbes elliptiques, l'opération de groupe est dénotée comme une addition. L'homologue de l'exponentiation est donc plutôt une multiplication par un entier :

$$iP = \underbrace{P +_{\mathbf{E}} P +_{\mathbf{E}} \ldots +_{\mathbf{E}} P}_{i \text{ fois}}$$

ce qui fait que la recherche du « logarithme » consiste à inverser une multiplication. Nous pouvons donc adapter El Gamal de façon à ce que l'hypothèse calculatoire sous-jacente soit la difficulté d'inverser une multiplication dans une courbe elliptique.

Le cryptosystème de la définition 1.41 suivante nécessite la connaissance de l'ordre du groupe q, c'est-à-dire le nombre de points sur la courbe $\mathbf{E}(\mathbb{Z}_p)$. Nous assumons qu'il est disponible puisqu'on peut le déterminer en temps polynomial avec l'algorithme de Schoof [61].

Définition 1.41. Étant donné un corps fini $\mathbb{Z}_p$, une courbe elliptique $\mathbf{E}(\mathbb{Z}_p)$ d'ordre q et un point $P \in \mathbf{E}(\mathbb{Z}_p)$, **El Gamal sur une courbe elliptique** est un cryptosystème asymétrique à porte dérobée défini par le tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

— $\mathcal{K}_{priv} = \{1, ..., q-1\}$;

— $\mathcal{K}_{pub} = \mathbf{E}(\mathbb{Z}_p)$;

— $gen(i) = iP$;

— $\mathcal{M} = \mathcal{C} = \mathbf{E}(\mathbb{Z}_p)$;

— $\mathcal{E} = \{E_{iP} | E_{iP}(M) = (jP, M +_{\mathbf{E}} jiP)$, où $j \in_{\mathcal{U}} \{1, ..., q-1\}\}$

— $\mathcal{D} = \{D_{iP} | D_{iP}((jP, M + jiP), i) = M\}$.

Algorithme 18 Génération de la clé publique El Gamal sur $\mathbf{E}(\mathbb{Z}_p)$

1: **Entrée :** La clé privée $i \in \{1, ..., q-1\}$ et les informations publiques $\mathbf{E}(\mathbb{Z}_p), P \in \mathbf{E}(\mathbb{Z}_p)$
2: **Retourner** iP

Algorithme 19 Chiffrement El Gamal sur $\mathbf{E}(\mathbb{Z}_p)$

1: **Entrée :** La clé publique $iP \in \mathbf{E}(\mathbb{Z}_p)$, le message $M \in \mathbf{E}(\mathbb{Z}_p)$ et les informations publiques $\mathbf{E}(\mathbb{Z}_p), P \in \mathbf{E}(\mathbb{Z}_p)$
2: Sélectionner un $j \in \{1, ..., q-1\}$ suivant une distribution uniforme
3: Calculer jP et $jiP = j(iP)$
4: **Retourner** $C = (jP, M +_{\mathbf{E}} jiP)$

Algorithme 20 Déchiffrement El Gamal sur $\mathbf{E}(\mathbb{Z}_p)$

1: **Entrée :** La clé publique $iP \in \mathbf{E}(\mathbb{Z}_p)$, la clé privée $i \in \{1, ..., q-1\}$, le message chiffré $C = (C_1, C_2) \in \mathbf{E}(\mathbb{Z}_p) \times \mathbf{E}(\mathbb{Z}_p)$ et les informations publiques $\mathbf{E}(\mathbb{Z}_p), P \in \mathbf{E}(\mathbb{Z}_p)$
2: Calculer $jiP = i(jP) = iC_1$
3: Calculer $-_{\mathbf{E}} jiP$ (si $jiP = (x, y)$, $-_{\mathbf{E}} jiP = (x, -y)$)
4: **Retourner** $M = (M +_{\mathbf{E}} jiP) -_{\mathbf{E}} jiP = C_2 +_{\mathbf{E}} (-_{\mathbf{E}} jiP)$

Exemple 1.42 (Utilisation de El Gamal sur $\mathbf{E}(\mathbb{Z}_p)$)**.** Supposons $\mathbf{E}(\mathbb{Z}_7) = \{(x, y) \in \mathbb{Z}_7 \times \mathbb{Z}_7 | y^2 = x^3 + 2x + 3\}$ et le point $P = (2, 1)$.

— Supposons la clé privée $i = 2$. Alors la clé publique est $2P = P +_{\mathbf{E}} P$. On calcule donc $P +_{\mathbf{E}} P = (3, 6)$ suivant la troisième propriété de l'addition.

— Supposons que le message est $M = (2, 6)$. On sélectionne (au hasard) $j = 3$, puis on calcule $jP = 3P = (3, 6) +_{\mathbf{E}} (2, 1) = (6, 0)$ (quatrième propriété). On calcule également $j(iP) = 3(3, 6)$ comme suit : $(3, 6) +_{\mathbf{E}} (3, 6) = (3, 1)$ (propriété 3), puis, remarquant qu'il s'agit du même x et que $6 = -1 \mod 7$, $jiP = (3, 1) +_{\mathbf{E}} (3, 6) = P_\infty$ (propriété 2). On a donc le message chiffré $(C_1, C_2) = (jP, M +_{\mathbf{E}} jiP) = ((6, 0), (2, 6))$. $M +_{\mathbf{E}} jiP = (2, 6) +_{\mathbf{E}} P_\infty = (2, 6)$ est obtenu grâce à la propriété 1.

— On a en possession $C_1 = (6, 0)$ et $C_2 = (2, 6)$. On calcule $jiP = iC_1 = 2(6, 0) = P_\infty$ (car $0 = -0 \mod 7$), puis son inverse $-_{\mathbf{E}} jiP$ qui demeure P_∞. On obtient $M = C_2 -_{\mathbf{E}} jiP = (2, 6) +_{\mathbf{E}} P_\infty = (2, 6)$, comme désiré.

Proposition 1.43. *La génération de la clé publique El Gamal sur une courbe elliptique est une fonction à sens unique.*

Démonstration. Une addition sur la courbe elliptique consiste en une poignée d'opérations élémentaires sur le corps ; les répéter i fois, cependant, nécessite un algorithme en $\mathcal{O}(\log i)$, i n'étant pas borné polynomialement. Cela peut être réalisé avec l'algorithme *Double-and-add* [31], dont le principe est similaire à celui de l'exponentiation rapide. Cependant, trouver i à partir de iP revient au logarithme discret. □

Définition 1.44.

Problème de Diffie-Hellman sur courbes elliptiques	
Entrée :	$P, iP, jP \in \mathbf{E}(\mathbb{Z}_p)$
Question :	Que vaut jiP ?

La porte dérobée rend le problème potentiellement plus facile. À l'instar de El Gamal sur $\mathbb{Z}_p^*$, il est suffisant de résoudre le problème de Diffie-Hellman, ou plutôt la variante suivante, supposée tout autant difficile :

Conjecture 1.45. Problème de Diffie-Hellman sur courbes elliptiques $\in \mathsf{NP} \backslash \mathsf{BPP}$.

Proposition 1.46. *La chiffrement El Gamal sur une courbe elliptique est une fonction à porte dérobée.*

Démonstration.

— Le chiffrement consiste en deux multiplications sur la courbe elliptique, ce qui se fait en temps polynomial.

— Retrouver M sans connaître jiP mais connaissant jP et iP revient au Problème de Diffie-Hellman sur courbes elliptiques.

— Retrouver M sachant i consiste en une multiplication sur la courbe, une inversion (une soustraction sur le corps) et une addition sur la courbe, opérations qui prennent toutes un temps polynomial.

□

Théorème 1.47. Briser El Gamal sur $\mathbf{E}(\mathbb{Z}_p) \leq$ Logarithme discret.

Démonstration. Si l'on peut résoudre le logarithme discret, alors on peut calculer i à partir de iP. L'entier i peut alors être utilisé comme porte dérobée pour déchiffrer M. □

1.4 Un cadre général pour la Factorisation et le Logarithme discret

1.4.1 Le problème du sous-groupe caché

Les cryptosystèmes les plus répandus se basent sur la factorisation et le logarithme discret car ce sont deux problèmes de nature discrète, on pense qu'il est facile de générer des instances difficiles et, surtout, il s'agit de problèmes algébriques qui semblent *trop complexes* pour nos ordinateurs. Les théorèmes 1.30, 1.37 et 1.47 indiquent que la capacité de résoudre ces deux problèmes implique la capacité de briser à peu près tout protocole cryptographique utilisé de nos jours.

En fait, la sécurité des protocoles actuels dépend de la difficulté d'un seul et même problème algébrique : le problème du sous-groupe caché pour les groupes abéliens (qu'on nommera HSP pour *Hidden Subgroup Problem*). En effet, comme il sera démontré dans les prochaines sections, la factorisation de même que le logarithme discret se réduisent en temps polynomial à HSP.

Définition 1.48.

Problème du sous-groupe caché pour les groupes abéliens	
Entrée :	Un groupe abélien G ayant un sous-groupe H, et une fonction $f : G \to X$, où X est un ensemble fini quelconque
Promesse :	f est constante sur chaque coset de H et différente d'un coset à l'autre
Question :	Quel est un ensemble générateur du sous-groupe H ? [50]

1.4.2 De Factorisation à HSP

La factorisation et HSP sont en apparence très différents, c'est pourquoi Factorisation est d'abord réduit (de façon probabiliste) au problème intermédiaire Recherche d'ordre, lequel est par la suite réduit (déterministement) à HSP.

Définition 1.49.

Recherche d'ordre	
Entrée :	Un entier n et un $a \in \mathbb{Z}_n^*$
Question :	Quel est le plus petit $r \in \mathbb{N}$ tel que $a^r = 1 \mod n$?

Dans le groupe multiplicatif $\mathbb{Z}_n^*$, l'ordre d'un élément $a \in \mathbb{Z}_n^*$ est le plus petit entier $r > 0$ tel que $a^r = 1 \mod n$. Malgré la nature différente des deux problèmes, on a que s'il est possible de trouver efficacement r pour tout élément a, alors il est possible de déduire un facteur non trivial (différent de 1 et de n) de n efficacement avec très haute probabilité.

Lemme 1.50. Factorisation $\leq$ Recherche d'ordre

Démonstration. Soit $n = \prod_{i=1}^{m} p_i^{k_i}$, où $m, k_1, k_2, ..., k_m \in \mathbb{N}^+, p_1, p_2, ..., p_m \in \mathbb{P}$. On montre d'abord que pour un x tel que $1 < x < n-1$ et $x^2 = 1 \mod n$, il est certain que $pgcd(x-1, n)$ ou $pgcd(x+1, n)$ est un facteur non trivial de n.

Soit $x \in \mathbb{Z}_n^*$ puis supposons que $x^2 = 1 \mod n$ et que $1 < x < n-1$. Alors $(x-1)(x+1) = x^2 - 1 = 0 \mod n$, c'est-à-dire $(x-1)(x+1) = in$ pour un entier i. Notons que l'égalité fait en sorte que les facteurs premiers de $(x-1)(x+1)$ sont exactement les mêmes que ceux de in, donc un facteur premier de n est aussi un facteur de $x-1$ ou de $x+1$. Il existe donc au moins un facteur commun entre n et $x-1$ ou $x+1$, lequel peut être calculé en temps polynomial en trouvant $pgcd(x-1, n)$ ou $pgcd(x+1, n)$. Ce facteur ne peut être égal à n car $1 < x < n-1$ par hypothèse, ce qui implique que $x - 1 < x + 1 < n$ et donc n ne peut être un facteur ni de $x-1$ ni $x+1$. Le facteur donné par $\max(pgcd(x-1, n), pgcd(x+1, n))$ est donc assurément un facteur non trivial de n.

Soit un quelconque $y \in \mathbb{Z}_n^*$, et soit r_y l'ordre de y modulo n. Si r_y est pair, alors en posant $x = y^{\frac{r_y}{2}}$, on a nécessairement $x^2 = y^{r_y} = 1 \mod n$. Il suffit alors de vérifier que $x \neq 1$ et $x \neq n-1$, auquel cas on obtient un x valable.

Il est montré dans [49] que la probabilité de piger un y donnant un tel x est élevée, c'est-à-dire qu'étant donné $y \in \mathbb{Z}_n^*$ pris aléatoirement selon une distribution uniforme, la probabilité que son ordre r_y soit impair ou que $y^{\frac{r_y}{2}} = \pm 1 \mod n$ est négligeable, en l'occurrence inférieure à $\frac{1}{2^m}$, où m est le nombre de facteurs premiers différents de n :

$$\Pr\Big(r_y \bmod 2 = 1 \vee y^{\frac{r_y}{2}} = \pm 1 \bmod n\Big) \leq \frac{1}{2^m}$$

Ainsi, en supposant l'existence d'un algorithme polynomial probabiliste pour Recherche d'ordre, l'algorithme suivant est un algorithme probabiliste polynomial pour Factorisation.

Algorithme 21 Trouver un facteur

Entrée : Un entier à factoriser $n > 1$
Tirer un $y \in \mathbb{Z}_n$ au hasard selon une distribution uniforme.
Calculer r_y, l'ordre de y modulo n
si r_y est impair **alors**
    L'algorithme **échoue**
sinon si $y^{\frac{r_y}{2}} = \pm 1 \mod n$ **alors**
    L'algorithme **échoue**
sinon
    Retourner $\max(pgcd(x-1, n), pgcd(x+1, n))$
fin si

À noter que l'algorithme réussit avec probabilité supérieure à $\frac{1}{2}$ seulement si n a au moins 2 facteurs premiers différents. Cependant, il est possible de déterminer au préalable si $n = a^b$ pour $a, b \in \mathbb{N}$ en temps polynomial sur un ordinateur classique [49]. De plus, bien que

l'algorithme ne retourne qu'un seul facteur et ne réussit qu'avec une certaine probabilité, on peut l'exécuter un nombre polynomial de fois pour obtenir presque assurément tous les facteurs. □

Lemme 1.51. Recherche d'ordre ≤ HSP

Démonstration. Le problème de recherche d'ordre (définition 1.49) peut être réduit au problème du sous-groupe caché. Dans l'instance de HSP que nous créerons, la fonction f a comme ensemble de départ le groupe additif $\mathbb{Z}_n$ (dont il existe un sous-groupe), et comme ensemble d'arrivée le groupe multiplicatif $\mathbb{Z}_n^*$ (dans lequel se trouve l'élément a en entrée à Recherche d'ordre).

Nous prenons la fonction $f(x) = a^x \mod n$. Pour la période (inconnue) r de a, on a $f(x) = a^x 1 = a^x a^{ir} = a^{x+ir} \mod n$ pour tout $i \in \mathbb{N}$. Donc, f est constante sur les sous-ensembles donnés par $\{x + ir + y : i \in \mathbb{N}\}$ pour un $y < r$ fixe, mais différente lorsque y n'est pas le même. Donc le sous-groupe caché par f est $H_0 = \{rm : m \in \mathbb{Z}_n\}$ et ses cosets sont $H_y = \{rm + y : m \in \mathbb{Z}_n\}$, où $1 \leq y < r$.

S'il est possible de résoudre le problème du sous-groupe caché pour les groupes abéliens, alors on peut trouver le générateur de H_0, qui est précisément la période r [51]. □

Théorème 1.52. Factorisation ≤ HSP

Démonstration. Par la transitivité de la réduction, les lemmes 1.50 et 1.51 impliquent que Factorisation ≤ HSP. □

1.4.3 De Logarithme discret à HSP

S'agissant déjà d'un problème de groupe, la réduction du logarithme discret à HSP est considérablement plus simple que pour la factorisation.

Théorème 1.53. Logarithme discret ≤ HSP

Démonstration. Rappelons que le problème du logarithme discret consiste à trouver l'entier k sachant g, le générateur d'un groupe abélien X, et $c = g^k$, un autre élément du groupe. Pour HSP, nous nous intéressons surtout au groupe $\mathbb{Z}_r \times \mathbb{Z}_r$, où $\mathbb{Z}_r$ est le groupe additif modulo l'ordre r de X. Nous présentons ici une fonction $f : \mathbb{Z}_r \times \mathbb{Z}_r \to X$ constante sur les cosets d'un sous-groupe dont la description nous informe directement sur la valeur de x.

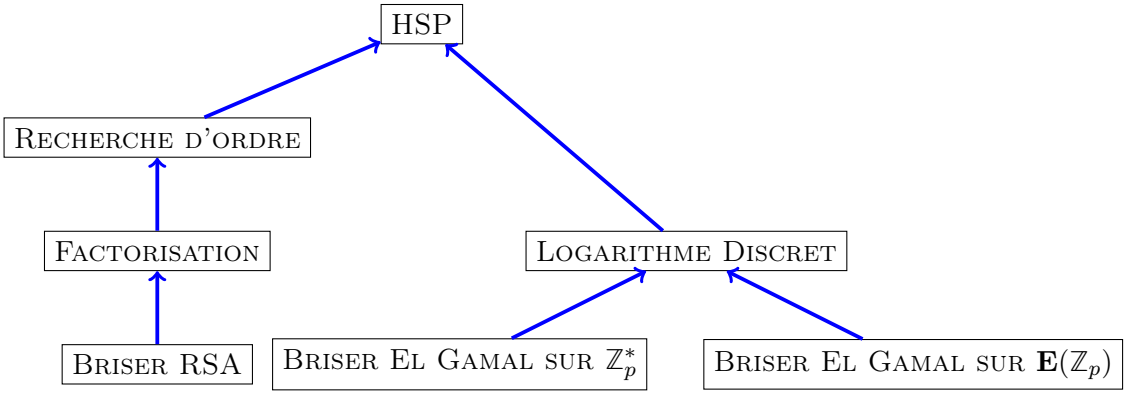


Figure 1.9 – Résumé des réductions présentées. Une flèche $A \to B$ signifie $A \leq B$.

Soit f telle que $f(\alpha, \beta) = c^\alpha g^{-\beta}$. Notons que

$$\begin{aligned} f(\alpha, \beta) &= c^\alpha g^{-\beta} \\ &= (g^k)^\alpha g^{-\beta} \\ &= g^{\alpha k} g^{-\beta} \\ &= g^{\alpha k - \beta} \\ &= g^y \end{aligned}$$

où $y = \alpha k - \beta$. On peut obtenir le même y pour des entrées différentes, et donc la fonction est constante sur certaines entrées : en l'occurence quand $\beta = \alpha k - y$. Considérons le sous-ensemble $H_y = \{(\alpha, \alpha k - y) : \alpha \in \mathbb{Z}_r\} \subseteq \mathbb{Z}_r \times \mathbb{Z}_r$. Pour $y = 0$, on a que $H_0 = \{(\alpha, \alpha k) : \alpha \in \mathbb{Z}_r\}$ est un sous-groupe de $\mathbb{Z}_r \times \mathbb{Z}_r$, et les cosets sont définis par les H_y où $y \neq 0$.

Supposons qu'on peut résoudre le problème du sous-groupe caché. Alors, étant donné $g, g^x \in X$ on peut définir la fonction f, qui est constante sur les cosets de H_0. En résolvant le problème du sous-groupe caché, on obtient le générateur $(1, k)$ de $H_0 = \{(\alpha, \alpha k) : \alpha \in \mathbb{Z}_r\}$, dans lequel k est la réponse au logarithme discret [50].

□

1.4.4 Et si HSP était facile ?

La figure 1.9 résume les réductions que nous avons présentées jusqu'à maintenant. La conséquence est bien visible : il suffit d'un seul algorithme efficace pour résoudre le problème du sous-groupe caché pour les groupes abéliens pour que l'ensemble des protocoles cryptographiques contemporains soient obsolètes.

Chaque réduction avait pour but d'aboutir à un seul problème général sur lequel nous nous concentrerons pour la suite. Même si l'on base les cryptosystèmes sur des problèmes difficiles

afin qu'ils soient impossibles à briser, réduire un cryptosystème à un problème n'est jamais un gage de sécurité en soi.

Pour obtenir un véritable indice de sécurité, il faudrait plutôt montrer la réduction inverse, par exemple montrer qu'un algorithme capable de briser RSA peut être utilisé pour factoriser n'importe quel nombre. Ce genre de preuve est très difficile à accomplir, car la nécessité d'avoir une porte dérobée donne une attaque potentielle supplémentaire, et oblige les cryptographes à recourir à davantage de conjectures ; par exemple la difficulté du problème RSA (conjecture 1.28).

L'existence de problèmes dans $\mathsf{NP} \setminus \mathsf{BPP}$ demeure une question ouverte ; même s'il en existe, la présence de Factorisation et Logarithme discret dans cette classe de complexité n'a jamais été montrée ; et le fait que RSA et El Gamal sont aussi difficiles que ces problèmes est une conjecture. Si la démonstration formelle de la sécurité d'un cryptosystème est un problème excessivement difficile, l'inverse ne l'est toutefois pas : il suffit d'un seul algorithme polynomial pour HSP et nous saurons *assurément* que ces protocoles ne sont pas sécuritaires.

Horizon II :
La période de transition

Transportons-nous plusieurs années dans le futur, par exemple en **2038** [1]. Une équipe de scientifiques et d'ingénieurs vient, après des décennies d'efforts, de finaliser un ordinateur quantique d'envergure, c'est-à-dire pouvant fonctionner de façon stable avec autant de *qubits* qu'ils le souhaitent.

À première vue, la prouesse scientifique qu'est la construction d'un tel ordinateur devrait être matière à réjouissance, puisque sa façon nouvelle d'effectuer des calculs permet vraisemblablement de résoudre des problèmes qui nous étaient trop difficiles. Ne célébrons pas trop vite ; la portée supplémentaire des ordinateurs quantiques, bien que limitée, rend calculables les quelques problèmes dont nous nous servions en cryptographie et que nous espérions être trop difficiles.

Un seul ordinateur quantique entre mauvaises mains suffit donc pour qu'il ne soit plus sécuritaire d'utiliser les bon vieux protocoles comme RSA et El Gamal. Peut-on continuer à communiquer de façon sécurisée avec nos simples ordinateurs classiques, sachant qu'un espion plus puissant que nous essaie peut-être d'écouter notre conversation ?

Nous supposons dans cet horizon l'existence d'un ordinateur quantique dans les mains d'une organisation d'espionnage malveillante. Nous explorons en détail en quoi consiste le calcul quantique et ses impacts en cryptographie, puis survolons des stratégies que pourraient utiliser Alice et Bob pour continuer à communiquer de façon sécurisée malgré tout.

1. 20 années est un intervalle assez arbitraire, mais souvent choisi par les futurologues puisqu'il laisse suffisamment de temps pour que l'avènement d'un évènement rare soit plausible, mais est suffisamment court pour que l'on se sente interpellé.

2.1 Algorithme de Shor

2.1.1 Vue d'ensemble du circuit quantique

Nous avons répété maintes fois que la factorisation, le logarithme discret, et par conséquent HSP, sont hors de portée des ordinateurs classiques, les ordinateurs répandus de par le monde et fonctionnant à l'aide de simples bits et portes logiques. Cependant, il est possible d'imaginer des modèles de calcul différents et possiblement plus puissants que les ordinateurs actuels. De tels modèles dépassent rarement le cadre théorique, mais l'un deux, le paradigme quantique, semble être tout à fait applicable en pratique.

Un algorithme quantique peut utiliser les propriétés contre-intuitives de la physique qui régissent l'échelle atomique, dont :

— La **superposition**, qui permet à un système quantique, et en particulier à un **qubit** (l'équivalent quantique du bit), de posséder plusieurs valeurs à la fois. Un qubit peut ainsi prendre les valeurs 0, 1, ou alors une superposition de 0 et 1. Avec n qubits, il devient alors possible de superposer 2^n chaînes binaires pour les traiter toutes en même temps.

— La **mesure** d'un système quantique rend toutefois la superposition beaucoup moins miraculeuse en stipulant que l'observation d'un système superposé s'effondre et prend au hasard une seule des 2^n valeurs possibles. A priori, cela ne vaut pas mieux qu'un algorithme probabiliste...

— Mais pas tout à fait, grâce à l'**intrication** quantique. L'intrication est une corrélation qui se crée entre qubits lors d'un calcul, faisant par exemple que si l'on mesure la sortie d'un algorithme quantique, alors on sait sans mesurer l'entrée qu'elle ne contient qu'une superposition de toutes les entrées ayant mené à cette sortie précise.

Toutes ces propriétés permettent l'existence d'un *algorithme quantique polynomial* pour HSP. Cet algorithme est dû principalement à Shor [64], qui le décrivit d'abord pour les cas particuliers Factorisation et Logarithme discret et à Jozsa [35] pour la généralisation au problème du sous-groupe caché pour les groupes abéliens (définition 1.48).

Cet algorithme menace de rendre RSA, El Gamal et plusieurs autres cryptosystèmes obsolètes. Quiconque arrive à l'exécuter pourra faire fi de la majorité des mesures de sécurité employées sur Internet. Heureusement –d'un point de vue cryptographique–, la création d'un ordinateur quantique d'envergure est encore un défi technique trop difficile et on est encore loin de pouvoir factoriser des nombres de 1024 bits [1].

Voici donc le terrible circuit quantique permettant de résoudre HSP.

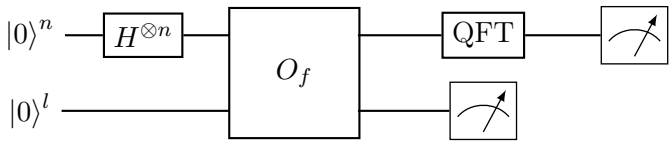


Figure 2.1 – Circuit quantique pour HSP

Dans ce circuit, que nous analyserons en détails dans les sections suivantes, les n qubits du premier registre (la ligne du haut) sont mis en superposition par des portes quantiques H puis sont donnés en entrée à un circuit calculant la fonction f de HSP. Le second registre (la ligne du bas) contient alors une superposition de toutes les images de f, qu'on mesure pour obtenir, par intrication quantique, une superposition de toutes les pré-images correspondantes dans le premier registre, donc tous les éléments d'un même coset. On utilise par la suite une *transformée de Fourier quantique* pour passer de l'ensemble des éléments à la quantité d'éléments superposés, ce qui nous informe sur la nature du sous-groupe caché.

Maintenant que nous avons une intuition du fonctionnement du circuit, nous pouvons nous lancer dans le formalisme mathématique pour nous convaincre que ce circuit résout véritablement HSP. Pour simplifier l'analyse du circuit quantique, nous assumons que le groupe représenté est $\mathbb{Z}_{2^n}$, mais l'analyse se fait également pour un groupe générique (voir [18]).

Remarque. Bien qu'il s'agisse d'une introduction complète à l'algorithme de Shor, incluant toutes les notions nécessaires pour le comprendre, cette section n'est pas une introduction au calcul quantique en général, et nous laissons de côté plusieurs détails qui seraient intéressants pour concevoir d'autres types d'algorithmes quantiques. Pour une introduction plus complète au domaine, nous conseillons [53].

2.1.2 Le qubit

Le calcul est avant tout un phénomène physique. Mais lorsqu'il est question de concevoir des algorithmes, on abstrait les appareils électriques sous-jacents pour travailler uniquement avec des objets mathématiques abstraits, dont l'unité d'information fondamentale, le bit. De même, le calcul quantique doit être basé sur de véritables systèmes quantiques : le qubit peut par exemple être défini par le *spin* d'un électron, qui peut être *up*, *down*, ou une superposition des deux. Cependant, le travail du concepteur d'algorithmes quantiques n'est pas de s'attarder à la façon de créer un qubit, mais plutôt de réfléchir à ce qu'on peut faire avec eux, connaissant leurs propriétés.

Nous verrons donc un qubit non pas comme un *spin* d'électron, mais plutôt comme l'abstraction mathématique décrite par les postulats de la mécanique quantique [44].

Définition 2.1. Un **qubit** est un espace de Hilbert complexe à deux dimensions $\mathcal{H}$.

Définition 2.2. L'**état** d'un qubit $\mathcal{H}$ est un vecteur unitaire

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \in \mathcal{H}, \text{ où } \alpha, \beta \in \mathbb{C} \text{ et } |\alpha|^2 + |\beta|^2 = 1$$

Notation. On représente généralement l'état d'un qubit avec la notation *ket* : $|\psi\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$. Le symbole ψ ne signifie rien en soi, il s'agit d'une étiquette pour identifier le vecteur.

Énoncé 2.3. Les vecteurs $|0\rangle = \begin{bmatrix}1\\ 0\end{bmatrix}$ et $|1\rangle = \begin{bmatrix}0\\ 1\end{bmatrix}$ forment une base orthonormale pour $\mathcal{H}$.

Il s'agit d'une base orthonormale parmi tant d'autres, mais nous l'utilisons pour sa simplicité. $|0\rangle$ et $|1\rangle$ n'ont pas de lien fondamental avec les 0 et 1 classiques, mais nous les étiquetons de cette façon pour établir une correspondance : lorsque mesurés, $\begin{bmatrix}1\\ 0\end{bmatrix}$ sera identifié comme un 0 classique et $\begin{bmatrix}0\\ 1\end{bmatrix}$ comme un 1.

Tout autre état est défini comme une combinaison linéaire de $|0\rangle$ et $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

où α et β sont des nombres complexes tels que $|\alpha|^2 + |\beta|^2 = 1$ afin d'assurer l'unitarité du vecteur. Si $\alpha > 0$ et $\beta > 0$, l'état est une superposition de $|0\rangle$ et $|1\rangle$.

Alors qu'un bit classique peut uniquement prendre une de deux valeurs, le qubit peut être décrit par tout élément de $\{(\alpha, \beta) \in \mathbb{C}^2 \mid |\alpha|^2 + |\beta|^2 = 1\}$, dont la cardinalité est *infinie*. Un qubit peut donc, *théoriquement*, contenir une quantité infinie d'information [53], une propriété qui peut être utilisée au sein d'un système fermé comme un circuit quantique et qui permet un gain significatif.

Par contre, pour déduire quelque chose d'un qubit au niveau macroscopique, il faut le mesurer à un certain point. La mesure d'un système quantique doit être faite par rapport à une certaine base orthonormale, et renvoie toujours un seul vecteur de cette base. On suppose toujours que la mesure est selon $\{|0\rangle, |1\rangle\}$, puisqu'il existe des opérations unitaires permettant de passer arbitrairement d'une base orthonormale à l'autre. Les portes H et QFT des sections 2.1.3 et 2.1.8 peuvent par ailleurs être vues comme des changement de bases.

Énoncé 2.4. La mesure d'un qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ selon la base $\{|0\rangle, |1\rangle\}$ le fait s'effondrer en

- $|0\rangle$ avec probabilité $|\alpha|^2$;
- $|1\rangle$ avec probabilité $|\beta|^2$.

Définition 2.5. Un qubit est en **superposition parfaite** lorsque la probabilité de mesurer $|0\rangle$ ou $|1\rangle$ est identique.

Exemple 2.6. L'état

$$|\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

est en superposition parfaite car $|\alpha|^2 = |\beta|^2 = (\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$.

2.1.3 Opérations sur un qubit

Nous pouvons transformer l'état d'un qubit à l'aide d'opérations, que l'on nomme portes quantiques par analogie avec les portes logiques à la base du calcul classique. Les portes logiques usuelles ont la particularité qu'elles peuvent détruire de l'information : sachant que $x_1 \vee x_2$ donne *vrai*, il n'y a aucun moyen de retrouver si x_1, x_2 ou tous deux étaient *vrais*. La destruction d'information vient avec une libération d'énergie, ce qui est possible à l'échelle macroscopique mais toutefois impossible à l'échelle quantique.

Cela force les portes quantiques à être toutes inversibles ; c'est-à-dire que si l'entrée d'une opération résulte en une sortie précise, cette dernière peut être utilisée sur l'opération inverse pour retrouver l'entrée initiale.

Comme l'état d'un qubit est un vecteur, une opération est non surprenamment une transformation linéaire. La linéarité est ce qui permet d'effectuer plusieurs calculs simultanément : une fonction $f : X \to Y$ est linéaire si

$$\sum_{x \in X} f(x) = f(\sum_{x \in X} x)$$

Or, la superposition est une somme de vecteurs de base, donc nous pourrons calculer uniquement le résultat de la somme, ce qui préserve l'information relative au calcul de f sur chaque vecteur de base.

L'unique contrainte sur cette transformation est qu'elle soit *unitaire*, afin de préserver la taille du vecteur tout en étant inversible.

Définition 2.7. Une **porte quantique** G est une transformation linéaire dont la matrice $\mathbf{M}$ est telle que $\mathbf{M}^\dagger\mathbf{M} = \mathbf{M}\mathbf{M}^\dagger = \mathbb{1}$, où $\mathbb{1}$ est la matrice identité et $\mathbf{M}^\dagger$ est la conjuguée transposée de $\mathbf{M}$. Comme nous travaillerons toujours dans la même base, nous utiliserons directement la porte quantique comme une matrice.

Exemple 2.8. Une porte $G_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ inversant les $|0\rangle$ et les $|1\rangle$ peut être réalisée puisqu'elle est unitaire :

$$G_1 G_1^\dagger = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{1}$$

L'application de G_1 sur un état générique donne

$$G_1 |\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Exemple 2.9. Une porte $G_2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$ transformant les $|0\rangle$ et les $|1\rangle$ en $|1\rangle$ ne peut être réalisée car elle n'est pas unitaire :

$$G_2 G_2^\dagger = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \neq \mathbb{1}$$

La porte d'Hadamard permet de créer une superposition parfaite à partir d'un état de base. À ce titre, il s'agit d'une des portes les plus importantes en calcul quantique.

Définition 2.10. La **porte d'Hadamard** est donnée par la matrice $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Proposition 2.11. *H est unitaire.*

Démonstration. Comme H est symétrique et ne contient aucune partie imaginaire, $H^\dagger = H$ alors il suffit de montrer $H^2 = \mathbb{1}$.

$$H^2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{1}$$

□

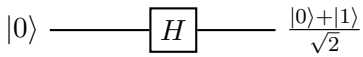
L'application de H sur un état générique donne

$$H |\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix}$$

En particulier, si $\alpha = 1$ et $\beta = 0$, on a un état parfaitement superposé :

$$H |0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 + 0 \\ 1 - 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

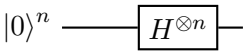
Notation. Cette dernière équation correspond au circuit suivant :



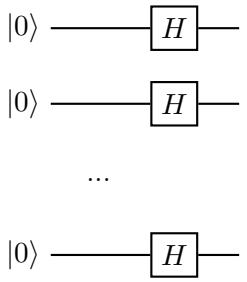
L'application de plusieurs portes en série résulte en une composition de transformations. La lecture d'un circuit se fait de gauche à droite, mais la composition de droite à gauche et ainsi l'ordre des matrices doit être l'inverse de celui des portes :

2.1.4 Plusieurs qubits

Le circuit quantique pour HSP contient entre autres le schéma



qui n'est rien de plus que l'application de la porte d'Hadamard sur n qubits en parallèle. En d'autres termes, c'est équivalent à



La mécanique quantique nous indique qu'un système composé est défini par le produit tensoriel des espaces d'états de ses sous-systèmes. Lorsque placés en parallèle, n qubits sont donc représentés par $\mathcal{H}^n = \underbrace{\mathcal{H} \otimes \mathcal{H} \otimes ... \otimes \mathcal{H}}_{n \text{ fois}}$.

Notation. Un élément générique de $\mathcal{H}^n$ est noté $\left|\psi^{(n)}\right\rangle$ pour dénoter qu'il s'agit d'un vecteur sur n qubits.

Comme chaque $\mathcal{H}$ a 2 dimensions, l'espace obtenu possède 2^n dimensions et est engendré par les 2^n combinaisons de vecteurs de bases possibles :

$$\mathcal{B} = \{|0\rangle \otimes |0\rangle \otimes ... \otimes |0\rangle, |0\rangle \otimes |0\rangle \otimes ... \otimes |1\rangle, ..., |1\rangle \otimes |1\rangle \otimes ... \otimes |1\rangle\}$$

aussi noté

— $\{|0\rangle |0\rangle ... |0\rangle, |0\rangle |0\rangle ... |1\rangle, ..., |1\rangle |1\rangle ... |1\rangle\}$,

— de façon plus condensée $\{|00...0\rangle, |00...1\rangle, ..., |11...1\rangle\}$,

— en notation décimale $\{|0\rangle, |1\rangle, |2\rangle, ..., |2^n - 1\rangle\}$,

— ou, en représentant les coordonnées, $\{\begin{bmatrix}1\\0\\...\\0\end{bmatrix}, \begin{bmatrix}0\\1\\...\\0\end{bmatrix}, ..., \begin{bmatrix}0\\0\\...\\1\end{bmatrix}\}$.

Toute combinaison linéaire unitaire de la base $\mathcal{B}$ est un état du système composé.

Exemple 2.12. Supposons $n = 2$. Alors $\mathcal{B} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ et un état est donné par $|\psi^{(2)}\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle = \begin{bmatrix}\alpha\\\beta\\\gamma\\\delta\end{bmatrix}$, avec $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$.

Définition 2.13. Un état est **pur** si son tenseur est séparable, c'est-à-dire si on peut le factoriser de la sorte : $|\psi^{(2)}\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle)$.

Exemple 2.14. L'état $\frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$ est pur, puisqu'il se factorise en $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

Un état pur est tel que la mesure d'un qubit n'a aucun impact sur les autres qubits. Si

$$|\psi^{(2)}\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle)$$

alors qu'on mesure $|0\rangle$ ou $|1\rangle$ sur le premier qubit, le second restera dans la superposition $\gamma|0\rangle + \delta|1\rangle$.

Définition 2.15. Un état est **intriqué** si son tenseur est non séparable, c'est-à-dire qu'il ne peut être factorisé complètement.

Si deux qubits sont intriqués, alors la mesure de l'un fera s'effondrer (totalement ou partiellement) la superposition de l'autre.

Exemple 2.16. L'état $\frac{|00\rangle}{\sqrt{2}} + \frac{|10\rangle+|11\rangle}{2}$ est intriqué :

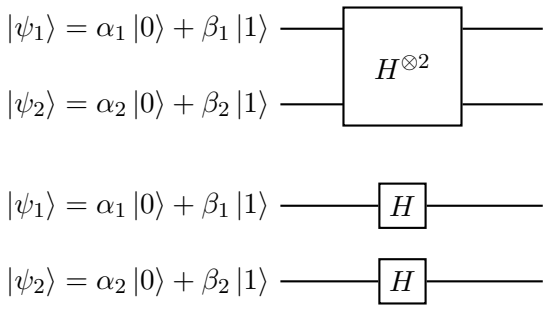
— Si la mesure du premier qubit donne $|0\rangle$, le second s'effondre lui aussi à $|0\rangle$.

— Si la mesure du premier qubit donne $|1\rangle$, le second s'effondre en la superposition $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

2.1.5 Opérations sur plusieurs qubits

Une opération sur n qubits est encore une fois une transformation linéaire unitaire, mais ayant 2^n rangées et colonnes. Elle peut parfois être définie par le produit tensoriel de plus simples opérations, auquel cas elle est équivalente à de simples opérations exécutées en parallèle.

Proposition 2.17. *Les deux circuits suivants sont équivalents :*



Démonstration. Montrons que $H^{\otimes 2}(|\psi_1\rangle |\psi_2\rangle) = H |\psi_1\rangle \otimes H |\psi_2\rangle$.

— Du côté gauche, on a l'état

$$|\psi_1\rangle |\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix} = \begin{bmatrix}\alpha_1\alpha_2\\ \alpha_1\beta_2\\ \beta_1\alpha_2\\ \beta_1\beta_2\end{bmatrix}$$

et l'opération

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}$$

dont l'application à l'état donne

$$H^{\otimes 2}(|\psi_1\rangle |\psi_2\rangle) = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix} \cdot \begin{bmatrix}\alpha_1\alpha_2\\ \alpha_1\beta_2\\ \beta_1\alpha_2\\ \beta_1\beta_2\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\alpha_1\alpha_2 + \alpha_1\beta_2 + \beta_1\alpha_1 + \beta_1\beta_2\\ \alpha_1\alpha_2 - \alpha_1\beta_2 + \beta_1\alpha_1 - \beta_1\beta_2\\ \alpha_1\alpha_2 + \alpha_1\beta_2 - \beta_1\alpha_1 - \beta_1\beta_2\\ \alpha_1\alpha_2 - \alpha_1\beta_2 - \beta_1\alpha_1 + \beta_1\beta_2\end{bmatrix}$$

— Du côté droit, on a les états

$$|\psi_1\rangle = \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix}, |\psi_2\rangle = \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix}$$

qui, passant par l'opération H, deviennent

$$H |\psi_1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \cdot \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_1 + \beta_1\\ \alpha_1 - \beta_1\end{bmatrix}$$

$$H |\psi_2\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \cdot \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_2 + \beta_2\\ \alpha_2 - \beta_2\end{bmatrix}$$

résultant en l'état composé

$$H|\psi_1\rangle \otimes H|\psi_2\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_1+\beta_1\\ \alpha_1-\beta_1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_2+\beta_2\\ \alpha_2-\beta_2\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\alpha_1\alpha_2+\alpha_1\beta_2+\beta_1\alpha_1+\beta_1\beta_2\\ \alpha_1\alpha_2-\alpha_1\beta_2+\beta_1\alpha_1-\beta_1\beta_2\\ \alpha_1\alpha_2+\alpha_1\beta_2-\beta_1\alpha_1-\beta_1\beta_2\\ \alpha_1\alpha_2-\alpha_1\beta_2-\beta_1\alpha_1+\beta_1\beta_2\end{bmatrix}$$

□

En particulier, si $|\psi_1\rangle = |\psi_2\rangle = |0\rangle$ ou, autrement dit, $\alpha_1 = \alpha_2 = 1$, $\beta_1 = \beta_2 = 0$, le circuit donne l'état parfaitement superposé $\frac{|00\rangle+|01\rangle+|10\rangle+|11\rangle}{2}$. La proposition 2.17 se généralise pour $n > 2$:

$$H^{\otimes n}(|0\rangle^n) = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}|j\rangle = \frac{1}{\sqrt{|\mathcal{B}|}}\sum_{|\psi^{(n)}\rangle\in\mathcal{B}}\left|\psi^{(n)}\right\rangle$$

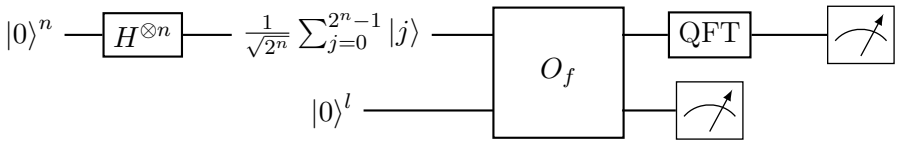
Il s'agit d'une parfaite superposition de chacun des qubits. Intuitivement, la chaîne de n qubits est en superposition des 2^n différentes chaînes de n bits.

Une opération sur n qubits n'est toutefois pas toujours séparable, c'est-à-dire qu'elle ne peut pas toujours être définie comme des portes indépendantes en parallèle. Une telle transformation génère de l'intrication entre les qubits, comme c'est le cas de l'oracle quantique O_f.

2.1.6 Oracle quantique

Jusqu'ici, nous sommes en mesure de comprendre le circuit jusqu'à l'entrée de l'oracle O_f. Avant de passer par O_f, l'état global des qubits est

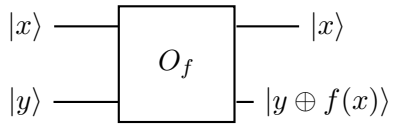
$$(\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}|j\rangle) \otimes |0\rangle^l = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}|j\rangle|0\rangle^l$$



L'oracle O_f est un sous-circuit calculant une fonction f quelconque. L'opérateur prend les n bits du premier registre (la ligne du haut) comme entrée et place le résultat de f dans le second registre (la ligne du bas).

Nous nous en servons comme une boîte noire pour deux raisons :

— Nous savons que, à condition qu'elle soit unitaire, toute transformation est possible. Or, nous pouvons rendre unitaire tout calcul de fonction en conservant l'entrée [64], c'est-à-dire en spécifiant l'oracle ainsi :



où x et y sont des chaînes binaires et $\oplus$ est le *ou exclusif* bit à bit. En prenant $|y\rangle = |0\rangle^l$, la sortie est directement écrite sur le second registre à la sortie de l'oracle. Mais surtout, appliquer $O_f^\dagger$ sur $|x\rangle$ et $|f(x)\rangle$ redonnerait $|0\rangle$ sur le second registre.

— Tout calcul classique prenant un temps polynomial peut être simulé par un calcul quantique en temps polynomial [11]. À condition que f soit calculable en temps polynomial sur un ordinateur classique, nous pouvons assumer que le circuit composant O_f est de taille polynomiale.

Dans notre cas, nous utilisons l'oracle pour calculer la fonction en entrée au problème HSP (définition 1.48). Cette fonction a comme domaine un groupe G et comme co-domaine un ensemble X. Comme nous avons besoin que le premier registre contienne une entrée de f et le second une sortie, nous choisissons $n = \lceil \log_2 |G| \rceil$ et $l = \lceil \log_2 |X| \rceil$.

Si $|x\rangle$ était un état de base de $\mathcal{H}^n$, alors la sortie de O_f serait simplement $|x\rangle\,|f(x)\rangle$. Mais nous envoyons plutôt une superposition des 2^n entrées possibles, ce qui résulte en une superposition de toutes les sorties existantes :

$$\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle\,|0\rangle^l \longmapsto^{O_f} \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle\,|f(j)\rangle$$

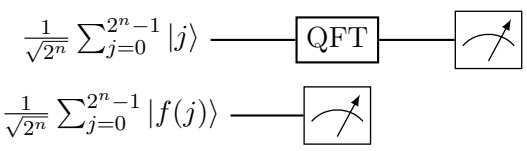
Remarque. Il y a certainement au plus 2^n images possibles étant donné qu'il s'agit du nombre d'entrées possibles. Dans les faits, il se peut très bien que l soit beaucoup plus petit que n, et donc que certaines images soient obtenues plusieurs fois. Le second registre contient dans ce cas une superposition des images pondérée par le nombre de fois qu'on obtient l'image.

2.1.7 Mesure de la sortie de l'oracle

À la sortie de l'oracle, l'état global est

$$\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle\,|f(j)\rangle$$

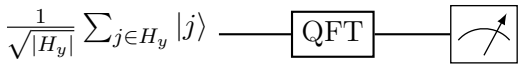
et la suite du circuit est :



Malgré l'indépendance apparente entre les deux registres, ils sont pourtant intimement liés par intrication. En effet, si l'on mesurait le premier registre, on obtiendrait un $|j\rangle$ où $j \in_{\mathcal{U}} \{0, 1, ..., 2^n - 1\}$ qui, par intrication, forcerait le second registre à prendre l'unique état $|f(j)\rangle$ correspondant. Si l'on mesure plutôt le second registre en premier lieu, on obtient un $|f(j)\rangle$ au hasard, ce qui n'est pas très intéressant a priori...

C'est malgré tout la prochaine étape du circuit, puisque le symbole  signifie qu'une mesure est prise. Le résultat de la mesure est pourtant tout à fait inutile et peut être immédiatement jeté ! C'est l'effet indirect de la mesure qui nous intéresse : étant intriquée avec le deuxième registre, le premier registre s'effondre sur une superposition des $|j\rangle$ correspondant au $|f(j)\rangle$ mesuré.

Rappelons-nous des particularités de la fonction f en entrée à HSP. Il s'agit d'une fonction $f : G \to X$, où G contient un sous-groupe H inconnu et X est non pertinent (il n'est pas étonnant qu'on jette le résultat de la mesure du second registre !), qu'on suppose constante sur chaque coset de H et différente d'un coset à l'autre. La fonction f associe donc tous les éléments d'un même coset H_y à un seul élément de X. En mesurant $f(j) \in X$, l'état du premier registre s'effondre sur la superposition parfaite de tous les éléments d'un quelconque coset H_y.



La solution à HSP est une description du sous-groupe H, laquelle est directement obtenue si l'on connaît la taille de H ou de l'un de ses cosets (chaque élément de H est séparé par $\frac{|G|}{|H|}$ éléments de G). Voyons comment on peut isoler $|H_y|$ à partir de $\frac{1}{\sqrt{|H_y|}} \sum_{j \in H_y} |j\rangle$.

2.1.8 Transformée de Fourier quantique

La transformée de Fourier permet de passer d'une fonction périodique compliquée à sa représentation comme une somme pondérée de fonctions périodiques simples. Très utilisée en analyse de signaux, elle permet par exemple de passer d'une fonction décrivant l'amplitude d'une onde par rapport au temps à une fonction donnant la pondération, pour chaque fréquence possible, de la simple fonction sinusoïdale composant la fonction originale.

Chaque fonction périodique peut être décrite comme une somme sur toutes les fréquences possibles d'une certaine quantité de cosinus et d'une certaine quantité de sinus. Cette paire de quantités peut être décrite par un seul nombre complexe, dont les parties réelle et imaginaire représentent respectivement les quantités de cosinus et de sinus.

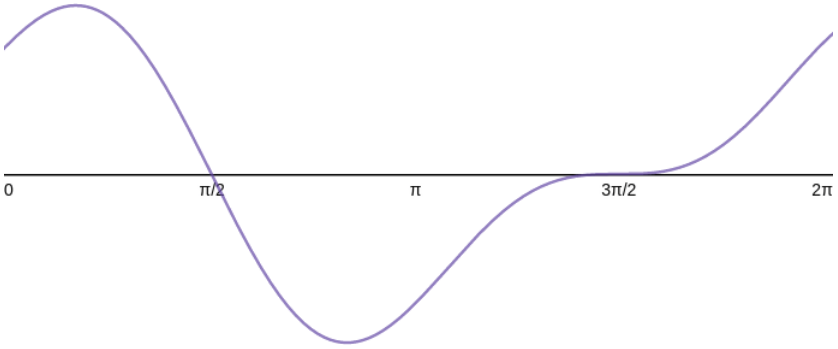


Figure 2.2 – La fonction $f(t) = \cos(t) + \frac{\sin(2t)}{2}$ (image générée avec le calculateur Desmos : https://www.desmos.com/calculator/lab9nylxsi)

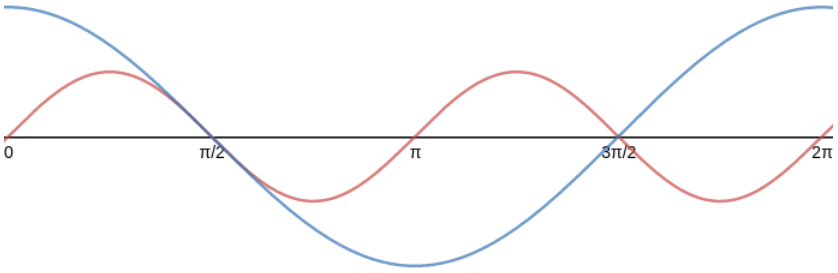


Figure 2.3 – En bleu, la fonction $f_1(t) = \cos(t)$ avec pondération 1 et en rouge, $f_2(t) = \sin(2t)$ avec pondération $\frac{1}{2}$ (image générée avec le calculateur Desmos : https://www.desmos.com/calculator/lab9nylxsi)

Exemple 2.18 (Transformée de Fourier)**.** La fonction $f(t) = \cos(t) + \frac{\sin(2t)}{2}$ (figure 2.2) équivaut à la somme pondérée des fonctions simples

- $f_1(t) = \cos(t)$, dont la fréquence est 1, et
- $f_2(t) = \sin(2t)$, dont la fréquence est 2 (figure 2.3),

accordant le poids 1 à la fréquence 1, $\frac{1}{2}i$ à la fréquence 2, ainsi que 0 à toute autre fréquence. La transformée de Fourier de f est

$$F(\omega) = \begin{cases} 1 & \text{si } \omega = 1 \\ \frac{1}{2}i & \text{si } \omega = 2 \\ 0 & \text{sinon} \end{cases}$$

Remarquons qu'une transformée de Fourier conserve toute l'information permettant de retrouver la fonction initiale, donc elle est inversible.

Dans le monde quantique, la transformée de Fourier permet de passer d'une superposition d'états sur n qubits à une somme pondérée des *fréquences* des éléments de la superposition.

Une fréquence m nous indique combien de fois on retrouve un état correspondant à $y + mr$ pour un $y < m$ fixe et $r \in \{0, 1, ..., \frac{2^n}{m} - 1\}$.

Comme la transformée de Fourier quantique est appliquée à l'état $\frac{1}{\sqrt{|H_y|}} \sum_{j \in H_y} |j\rangle$, la superposition n'ayant que les éléments du coset H_y, et puisque les éléments de H_y sont tous équidistants, toutes les fréquences auront une amplitude nulle sauf la fréquence $|H_y|$ et ses multiples.

Définition 2.19. La **transformée de Fourier quantique** fait correspondre un état donné par $\sum_{j=0}^{2^n-1} x_j |j\rangle$ avec l'état

$$\sum_{j=0}^{2^n-1} y_j |j\rangle \text{, où } y_j = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} x_k \omega^{jk} \text{ et } \omega = e^{\frac{2\pi i}{2^n}}$$

Dans cette définition, ω est choisi pour être une racine complexe de l'unité de 2^n, c'est-à-dire être tel que $\omega^{2^n} = 1$. La matrice QFT correspondant à cette opération est :

$$QFT = \frac{1}{\sqrt{2^n}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{(2^n-1)} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(2^n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(2^n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(2^n-1)} & \omega^{2(2^n-1)} & \omega^{3(2^n-1)} & \dots & \omega^{(2^n-1)(2^n-1)} \end{bmatrix}$$

Proposition 2.20. *QFT est unitaire.*

Démonstration. On doit montrer $QFT^\dagger \cdot QFT = QFT \cdot QFT^\dagger = \mathbb{1}$. Par symétrie de QFT, on a que $QFT^\dagger = QFT^*$, le conjugué complexe de QFT.

Chaque élément q_{ij} de $QFT \cdot QFT^*$ est donné par

$$\begin{aligned} q_{ij} &= \sum_{k=0}^{2^n-1} (\frac{1}{\sqrt{2^n}} \omega^{ik})(\frac{1}{\sqrt{2^n}} (\omega^{jk})^*) \\ &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^{ik} (\omega^{jk})^* \\ &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^{ik} \omega^{-jk} \\ &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^{(i-j)k} \end{aligned}$$

Si $i = j$, c'est-à-dire qu'il s'agit d'un élément sur la diagonale, alors

$$q_{ij} = \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^0 = \frac{1}{2^n} \sum_{k=0}^{2^n-1} 1 = \frac{1}{2^n} 2^n = 1$$

En revanche, si $i \neq j$, alors $\omega^{i-j} \neq 1$ mais demeure une racine de l'unité de 2^n, donc on peut utiliser le lemme suivant, où toutes les puissances de la racine de l'unité, lorsque additionnées, s'annulent mutuellement :

Lemme 2.21. *Soit u tel que $u^N = 1$ mais $u \neq 1$. Alors $\sum_{k=0}^{N-1} u^k = 0$* [18].

On a donc bien $q_{ij} = \frac{1}{2^n} \sum_{k=0}^{2^n-1} (\omega^{i-j})^k = 0$. Notons que $\forall i, j,\ q_{ij} = q_{ji}$, donc $QFT \cdot QFT^* = QFT^* \cdot QFT = \mathbb{1}$ [18]. □

Étant unitaire, on sait que la transformée de Fourier quantique est réalisable en théorie. Dans [53], un circuit de taille *polynomiale* est présenté pour une transformée quantique agissant sur 2^n dimensions, ce qui confirme que nous pouvons l'utiliser dans notre circuit.

Voyons ce qui arrive lorsqu'on donne notre état $\frac{1}{\sqrt{|H_y|}} \sum_{j \in H_y} |j\rangle$ à la porte quantique QFT. D'abord, on a que

$$\frac{1}{\sqrt{|H_y|}} \sum_{j \in H_y} |j\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle \text{, où } x_j = \begin{cases} \frac{1}{\sqrt{|H_y|}} & \text{si } j \in H_y \\ 0 & \text{sinon} \end{cases}$$

On peut donc utiliser la correspondance donnée à la définition 2.19 telle quelle pour obtenir :

$$QFT\left(\frac{1}{\sqrt{|H_y|}} \sum_{j \in H_y} |j\rangle\right) = \sum_{j=0}^{2^n-1} y_j |j\rangle \text{, où } y_j = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} x_k \omega^{jk} \text{ et } x_k = \begin{cases} \frac{1}{\sqrt{|H_y|}} & \text{si } k \in H_y \\ 0 & \text{sinon} \end{cases}$$

qu'on peut réécrire

$$\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} x_k \omega^{jk} |j\rangle$$

Comme chaque élément de H_y est séparé par un nombre r inconnu mais fixe d'éléments, on peut noter chaque k où x_k est non nul par $k = y + mr$, avec $m \in \{0, ..., |H_y| - 1\}$. L'état obtenu suite à la transformée de Fourier quantique peut donc s'écrire

$$\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \sum_{m=0}^{|H_y|-1} x_{(y+mr)} \omega^{j(y+mr)} |j\rangle$$

Avec $x_{(y+mr)} = \frac{1}{\sqrt{|H_y|}}$, cela équivaut à

$$\frac{1}{\sqrt{2^n |H_y|}} \sum_{j=0}^{2^n-1} \sum_{m=0}^{|H_y|-1} \omega^{j(y+mr)} |j\rangle$$

$$= \sum_{j=0}^{2^n-1} \frac{\omega^{jy}}{\sqrt{2^n |H_y|}} \sum_{m=0}^{|H_y|-1} \omega^{jmr} |j\rangle$$

Il s'agit d'une superposition sur tous les $|j\rangle$ possibles, où chacun a le coefficient

$$\frac{\omega^{jy}}{\sqrt{2^n|H_y|}} \sum_{m=0}^{|H_y|-1} \omega^{jmr}$$

Rappelons-nous que, dans une superposition, la somme des normes au carré des coefficients doit toujours être égale à 1 :

$$\sum_{j=0}^{2^n-1} \left| \frac{\omega^{jy}}{\sqrt{2^n|H_y|}} \sum_{m=0}^{|H_y|-1} \omega^{jmr} \right|^2 = 1$$

Or, prenons l'un des $|j\rangle$ tels que $j = l|H_y|$, avec $l \in \{0, ..., r-1\}$. On a que $jr = l|H_y|r = l2^n$, donc $\omega^{jmr} = (\omega^{2^n})^{lm} = 1^{lm} = 1$. Le coefficient est donc $\frac{\omega^{jy}}{\sqrt{2^n|H_y|}} \sum_{m=0}^{|H_y|-1} 1 = \frac{\omega^{jy}|H_y|}{\sqrt{2^n|H_y|}} = \frac{\omega^{jy}\sqrt{|H_y|}}{\sqrt{2^n}}$, ce qui équivaut à $\frac{\omega^{jy}}{\sqrt{r}}$. Finalement,

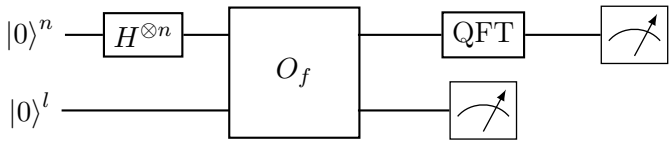
$$\left|\frac{\omega^{jy}}{\sqrt{r}}\right|^2 = \left|\omega^{jy}\right|^2 \left|\frac{1}{\sqrt{r}}\right|^2 = \frac{1}{r}$$

Comme il y a r états dont la norme au carré du coefficient est $\frac{1}{r}$, la somme pour ces r états seulement atteint déjà 1, donc tout autre état a *nécessairement* un coefficient nul. La transformée de Fourier quantique nous laisse donc dans la superposition finale

$$\sum_{l=0}^{r-1} \frac{\omega^{(l|H_y|)y}}{\sqrt{r}} |l|H_y|\rangle$$

Il ne nous reste plus qu'à mesurer l'état, ce qui donne un des $l|H_y|$, chacun avec probabilité $\frac{1}{r}$. On connaît ainsi un multiple de la taille du sous-groupe caché [18].

2.1.9 Le circuit quantique comme sous-routine



En résumé, le circuit ci-dessus commence avec deux registres, l'un pour le groupe G et l'autre pour X. Le premier contient en quelque sorte l'entrée de la fonction $f : G \to X$ et nécessite $n = \lceil \log_2 |G| \rceil$ qubits pour pouvoir représenter chaque entrée possible, tandis que le second est préparé pour contenir la sortie, soit un élément de X, et doit donc contenir $l = \lceil \log_2 |X| \rceil$ qubits.

Le premier registre est mis en **superposition** des 2^n possibles entrées grâce à n portes d'Hadamard en parallèle. Le registre est donné comme entrée à l'oracle calculant f, qui laisse ce registre inchangé mais écrit la sortie correspondant à la superposition des entrées de f, qui équivaut à la superposition des images de f par **linéarité**, sur le second. La **mesure** de ce dernier registre donne un $x \in X$ inintéressant mais, par **intrication**, fait s'effondrer la superposition des entrées sur le premier registre à la superposition des $j \in G$ tels que $f(j) = x$.

L'équivalent quantique d'une **transformée de Fourier** est alors appliquée à cette superposition pour faire en sorte qu'on mesure finalement non pas un $j \in G$ tel que $f(j) = x$, mais plutôt un multiple du nombre d'éléments superposés.

Nous avons donc un circuit qui, pour une fonction f constante sur les cosets d'un sous-groupe H mais différente de l'un à l'autre, retourne un $l|H| < |G|$. Un nombre polynomial t d'exécutions du circuit convient pour avoir, avec haute probabilité, suffisamment de l différents pour calculer de façon classique $|H| = pgcd(l_1|H|, l_2|H|, ..., l_t|H|)$. De $|H|$ on peut aisément calculer $r = \frac{|G|}{|H|}$, la distance entre les éléments du sous-groupe, qui nous informe directement sur le sous-groupe [18].

Exemple 2.22 (Calcul de r à partir de $|H|$)**.** Soit le groupe additif $\mathbb{Z}_{12}$ et la fonction périodique

$$\begin{aligned} f : \mathbb{Z}_{12} &\to \mathbb{Z}_4 \\ x &\mapsto x \mod 4 \end{aligned}$$

constante sur le sous-groupe $\{0, 4, 8\}$ et ses cosets $\{1, 5, 9\}, \{2, 6, 10\}, \{3, 7, 11\}$. Savoir que le sous-groupe est de taille 3 sachant que $\mathbb{Z}_{12}$ a 12 éléments indique que les éléments du sous-groupe sont séparés par $\frac{12}{3} = 4$ éléments. Sachant que le sous-groupe doit contenir l'élément neutre 0, on peut déduire qu'il contient aussi 4 et 8.

Remarque. La puissance d'un algorithme quantique n'est pas simple à cerner. L'algorithme de Shor nécessite que la fonction f analysée soit calculable polynomialement sur un ordinateur classique ; il ne s'agit donc pas d'exécuter des fonctions requérant un nombre d'étapes exponentiel. De plus, bien qu'une des étapes du circuit consiste à calculer simultanément les 2^n entrées possibles à une fonction de n bits d'entrée, ce n'est pas ce que le circuit, globalement, permet d'obtenir en sortie. L'algorithme trouve plutôt une *méta-information* sur la fonction analysée, en l'occurence sa période.

2.2 Avantage du calcul quantique

2.2.1 Avantage superpolynomial : BQP

Si l'on peut modéliser un problème algorithmique comme le problème de trouver la période d'une fonction polynomiale, alors l'algorithme de Shor est bel et bien une façon de le résoudre en temps polynomial, avec une probabilité d'échec bornée et suivant le paradigme quantique. Le problème se retrouve ainsi dans une nouvelle classe de complexité combinant ces caractéristiques : BQP (pour *Bounded-error Quantum Polynomial time*).

Définition 2.23. La classe de complexité BQP contient les problèmes pour lesquels il existe un algorithme *quantique probabiliste polynomial* dont la sortie est correcte avec une probabilité d'au moins $\frac{2}{3}$ [11].

On remarque qu'à l'exception du modèle de calcul, cette classe est définie exactement comme BPP. De façon analogue, on considère donc que BQP contient tous les problèmes traitables efficacement par un ordinateur quantique. Ainsi :

Énoncé 2.24. Dans un monde où l'espion peut détenir un ordinateur quantique, tout problème algorithmique servant d'hypothèse calculatoire à un cryptosystème doit être basé sur un problème dans $\mathsf{NP} \setminus \mathsf{BQP}$.

Grâce à l'algorithme de Shor, on sait que certains problèmes que l'on croit être dans $\mathsf{NP} \setminus \mathsf{BPP}$, qui nécessitent un algorithme de complexité exponentielle pour les résoudre, *ne sont pas* dans $\mathsf{NP} \setminus \mathsf{BQP}$, et donc l'ordinateur quantique possède un *avantage superpolynomial* par rapport à l'ordinateur classique pour ces problèmes :

Théorème 2.25. *Le problème du sous-groupe caché pour les groupes abéliens (*HSP*) est dans* BQP *et donc pas dans* $\mathsf{NP} \setminus \mathsf{BQP}$.

Corollaire 2.26. Factorisation *et* Logarithme discret *ne sont pas dans* $\mathsf{NP} \setminus \mathsf{BQP}$ *et ne peuvent donc pas servir d'hypothèse calculatoire en présence d'ordinateurs quantiques.*

Reste-t-il des problèmes candidats pour servir d'hypothèse calculatoire en présence de ce nouveau paradigme ? Comme le calcul quantique peut simuler efficacement un calcul classique, on a nécessairement que $\mathsf{BPP} \subseteq \mathsf{BQP}$; par contre la relation entre NP et BQP demeure floue. Par exemple, aucun algorithme quantique offrant un avantage superpolynomial sur un problème NP-complet n'a été découvert à ce jour.

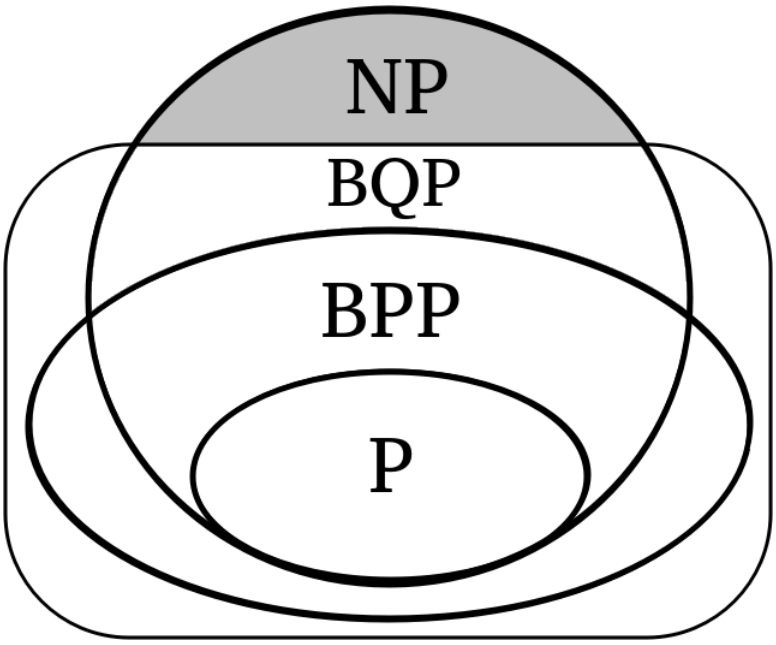


Figure 2.4 – Relations entre BQP et les classes P, BPP et NP avec en gris la classe NP \ BQP.

2.2.2 Avantage polynomial : Algorithme de Grover

Un autre algorithme quantique, l'algorithme de Grover, permet toutefois d'obtenir un *avantage polynomial* sur tous les problèmes dans NP \ BQP. Lorsque le problème consiste à trouver une solution parmi un ensemble non structuré de N solutions potentielles, comme c'est le cas des problèmes NP-complets, la difficulté du problème peut être réduite à $\mathcal{O}(\sqrt{N})$ [28].

Le circuit de la figure 2.5 consiste à appliquer plusieurs fois un même sous-circuit nommé itération de Grover (figure 2.6) à une superposition parfaite des 2^n bases pour la faire converger vers la base qui représente la solution. Chacune des itérations change le vecteur en entrée pour un vecteur un peu plus près de la base solution, si bien qu'après $\mathcal{O}(\sqrt{N})$ il est presque certain que la mesure de l'état donnera cette solution.

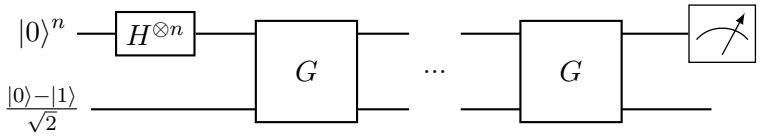


Figure 2.5 – Circuit quantique de Grover. L'itération de Grover, G, est répétée $\mathcal{O}(\sqrt{N})$ fois.

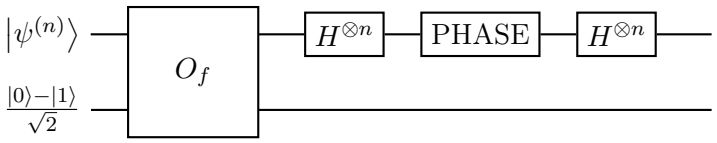


Figure 2.6 – L'itération de Grover

Le premier registre contient suffisamment de qubits pour encoder toutes les solutions possibles, alors que le deuxième contient un seul qubit restant toujours dans l'état $\frac{|0\rangle-|1\rangle}{2}$ et servant uniquement au calcul de l'oracle [2].

Ledit oracle est l'implémentation quantique d'un algorithme classique *reconnaissant* une solution au problème traité. Par définition de NP, un tel algorithme classique polynomial existe et nous pouvons donc assumer l'existence d'un oracle calculant la fonction $f : \mathbb{Z}_{2^n} \to \{0, 1\}$ suivante :

$$f(x) = \begin{cases} 0 & \text{si } x \text{ n'est pas une solution} \\ 1 & \text{si } x \text{ est une solution} \end{cases}$$

Pour assurer l'unitarité de l'oracle, il faut préserver l'entrée $|x\rangle$ dans le premier registre, et changer le qubit $|q\rangle$ du second registre par $|q \oplus f(x)\rangle$. Cependant, contrairement au cas de l'algorithme de Shor, $|q\rangle$ n'est pas dans un état de base. Alors si $f(x) = 0$, $|q\rangle$ reste inchangé, mais si $f(x) = 1$, l'oracle change $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ pour $\frac{-|0\rangle+|1\rangle}{\sqrt{2}}$. L'état global passe ainsi de $|x\rangle\,(\frac{|0\rangle-|1\rangle}{\sqrt{2}})$ à $|x\rangle\,(\frac{-|0\rangle+|1\rangle}{\sqrt{2}})$, ce qui équivaut à

$$(-|x\rangle)(\frac{|0\rangle - |1\rangle}{\sqrt{2}})$$

Le qubit du second registre, bien qu'utile dans la réalisation du circuit de l'oracle, reste donc toujours inchangé ; nous pouvons alors l'ignorer pour la suite et considérer que l'oracle fait correspondre $|x\rangle$ à $(-1)^{f(x)}\,|x\rangle$. Lorsque appliqué sur un état parfaitement superposé $|\psi^{(n)}\rangle$, l'oracle transforme l'état en

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}\,|x\rangle$$

c'est-à-dire qu'uniquement les composantes menant à des solutions sont inversées [53].

Supposons qu'il existe une et une seule solution, représentée par la base $|s\rangle$, $s \in \mathbb{Z}_{2^n}$. Rappelons que tout état $|\psi^{(n)}\rangle$ est une combinaison linéaire des vecteurs bases, incluant $|s\rangle$, et que ces bases sont toutes orthogonales. On peut décrire $|\psi^{(n)}\rangle$ comme une combinaison linéaire de deux vecteurs seulement : le vecteur $|s\rangle$ et le vecteur $|t\rangle = \frac{1}{\sqrt{2^n-1}} \sum_{x\in\mathbb{Z}_{2^n}\setminus\{s\}} |x\rangle$. Ces deux vecteurs de n qubits sont perpendiculaires et forment un plan dans lequel se trouve $|\psi^{(n)}\rangle = \alpha_s\,|s\rangle + \alpha_t\,|t\rangle$.

2. On peut obtenir un tel qubit en appliquant une porte H à $|1\rangle$.

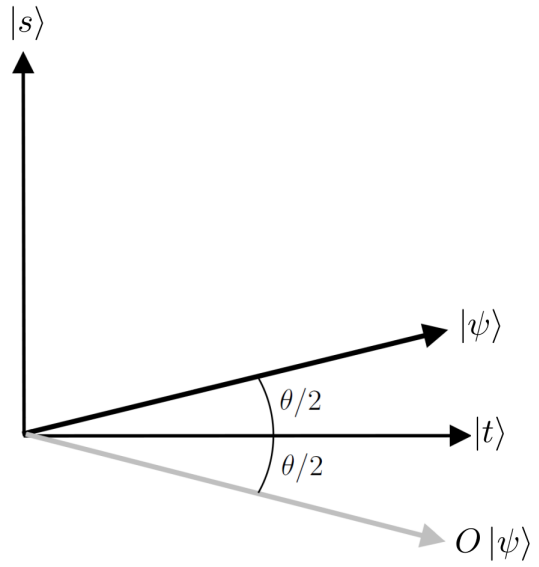


Figure 2.7 – L'application de l'oracle inverse la base $|s\rangle$, ce qui résulte en une réflexion par rapport à $|t\rangle$.

Appliquer l'oracle à $\left|\psi^{(n)}\right\rangle$ donne $-\alpha_s |s\rangle + \alpha_t |t\rangle$, ce qui correspond à une réflexion par rapport à l'axe $|t\rangle$ (figure 2.7). À la première exécution de l'oracle, l'état en entrée est $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$ et celui en sortie est ainsi $-\frac{1}{\sqrt{2^n}} |s\rangle + \frac{2^n-1}{\sqrt{2^n}} |t\rangle$.

La suite du circuit, une porte nommée $PHASE$ entourée de deux ensembles de portes d'Hadamard, permet d'augmenter l'amplitude des bases divergeant de la superposition parfaite –en l'occurence la base représentant la solution inversée par l'oracle– au détriment des autres. On peut définir la porte $PHASE$ par la matrice suivante :

$$PHASE = \begin{bmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & -1 & 0 & ... & 0 \\ 0 & 0 & -1 & ... & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & ... & -1 \end{bmatrix}$$

Cette porte est capable d'inverser tout vecteur de base à l'exception de $\left|0^{(n)}\right\rangle$. Globalement, avec les deux ensembles de portes d'Hadamard, cela devient :

$$H^{\otimes n} \cdot PHASE \cdot H^{\otimes n} = \begin{bmatrix} \frac{2}{\sqrt{2^n}} - 1 & \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} & ... & \frac{2}{\sqrt{2^n}} \\ \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} - 1 & \frac{2}{\sqrt{2^n}} & ... & \frac{2}{\sqrt{2^n}} \\ \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} - 1 & ... & \frac{2}{\sqrt{2^n}} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} & \frac{2}{\sqrt{2^n}} & ... & \frac{2}{\sqrt{2^n}} - 1 \end{bmatrix}$$

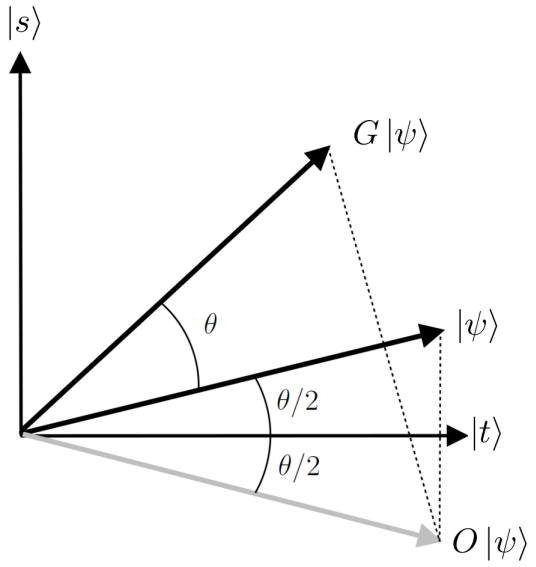


FIGURE 2.8 – Supposant un angle de $\frac{\theta}{2}$ entre $|\psi\rangle$ et $|t\rangle$, l'application à $|\psi\rangle$ d'une réflexion par rapport à $|t\rangle$ puis par rapport à $|\psi\rangle$ résulte en une rotation de θ degrés vers $|s\rangle$.

On peut voir cette porte comme une inversion par rapport à la moyenne des amplitudes : un état $\sum_{i=0}^{2^n-1} a_i |i\rangle$ passant par cette porte devient

$$\sum_{i=0}^{2^n-1} b_i |i\rangle$$

où

$$\begin{aligned} b_i &= (\frac{2}{\sqrt{2^n}} - 1)a_i + \sum_{j \in \mathbb{Z}_{2^n} \setminus \{i\}} \frac{2}{\sqrt{2^n}} a_j \\ &= -a_i + \frac{2}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} a_j \end{aligned}$$

Cela implique que $\frac{a_i+b_i}{2}$ est égal à la moyenne des amplitudes $\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} a_j$. Si a_i divergeait de x par rapport à la moyenne, alors la nouvelle amplitude de $|i\rangle$, b_i, doit diverger de $-x$ [12]. Dans le cas de notre vecteur $-\frac{1}{\sqrt{2^n}} |s\rangle + \frac{2^n-1}{\sqrt{2^n}} |t\rangle$, la moyenne est très près de $\frac{1}{\sqrt{2^n}}$ et donc la porte exécute approximativement une réflexion de ce vecteur par rapport à l'état initial $\left|\psi^{(n)}\right\rangle$.

La figure 2.8 montre que l'itération de Grover fait une première réflexion de l'état $|\psi\rangle$ par rapport à $|t\rangle$ pour obtenir $O|\psi\rangle$, puis réfléchit (approximativement) $O|\psi\rangle$ par rapport à $|\psi\rangle$, pour obtenir $G|\psi\rangle$. On remarque que le vecteur ainsi obtenu est bien plus près de la solution $|s\rangle$ qu'initialement. Dans son article, Grover montre que $\mathcal{O}(\sqrt{2^n})$ itérations suffisent pour mesurer la solution avec très grande probabilité [28].

Ce gain quadratique n'est pas dramatique en cryptographie –il suffit de doubler le nombre de bits du problème pour qu'il redevienne aussi difficile– mais il doit être pris en considération lors de la conception de cryptosystèmes résistant à des ordinateurs quantiques.

2.2.3 Propriétés nécessaires à la cryptographie post-quantique

Rappelons la définition de sécurité d'un cryptosystème asymétrique définie à la section 1.3.1. L'espionne Ève réussit son attaque si elle peut déduire la clé privée du récepteur Bob à partir de la clé publique ou la valeur originale du message à partir de sa version chiffrée. On suppose qu'elle ne peut interagir avec le système ; elle peut seulement écouter et effectuer des calculs de son côté. Tous ces critères demeurent valides dans le paradigme quantique, mais maintenant on considère qu'Ève est une machine BQP plutôt que BPP. En d'autres termes, les calculs qu'elle effectue de son côté sont exécutés sur un ordinateur quantique.

Définition 2.27. Un **cryptosystème asymétrique post-quantique** est un cryptosystème asymétrique conçu pour fonctionner avec des ordinateurs classiques tout en étant sécuritaire face à un espion ayant accès à un ordinateur quantique.

La profonde différence entre les cryptosystèmes classiques et les post-quantiques est la définition des fonctions à sens unique et à porte dérobée sous-jacentes, où l'on a simplement remplacé les occurences de BPP par BQP.

Définition 2.28. Une **fonction à sens unique post-quantique** f est une fonction telle que le calcul de $f(x) = y$ est dans P tandis que le calcul inverse, $f^{-1}(y) = x$ n'est pas dans BQP.

Définition 2.29. Une **fonction à porte dérobée post-quantique** est une famille de fonctions $\{f_k\}$ ayant les particularités suivantes :

- $f_k(x) = y$, où $f_k \in \mathsf{P}$.
- $f_k^{-1}(y) = x$, où $f_k^{-1} \in \mathsf{NP} \setminus \mathsf{BQP}$.
- $\exists g, s_k$ tel que $g(y, s_k) = x$, où $g \in \mathsf{P}$.

L'algorithme de Grover n'a pas d'impact sur ces définitions, puisque son gain est seulement polynomial. Toutefois, il nous oblige à utiliser bien plus de bits, ce qui ralentit les opérations de chiffrement et de déchiffrement par un facteur polynomial. Il ne faut donc pas être surpris que les protocoles post-quantiques soient en général considérablement moins efficaces que les protocoles actuels.

En résumé, le problème utilisé comme hypothèse calculatoire ne doit pas être dans BQP et doit demeurer suffisamment difficile malgré l'algorithme de Grover, en plus de continuer à

respecter tout critère nécessaire dans le paradigme classique. Heureusement, un tel problème peut provenir d'à peu près n'importe quelle branche des mathématiques, ce qui donne espoir que nous ne serons jamais à court de problèmes difficiles [3].

3. En revanche, cela nous oblige à définir une quantité importante de notions mathématiques complexes pour faire un bon recensement des méthodes proposées. Nous ferons de notre mieux pour nous limiter à l'essentiel !

2.3 Protocoles post-quantiques

Dans la prochaine section, nous présentons quelques propositions de problèmes calculatoires résistant jusqu'à maintenant au paradigme quantique et qui peuvent ainsi servir à bâtir une cryptographie post-quantique. Les cryptosystèmes basés sur le **décodage de syndrome**, sur les **réseaux euclidiens** et sur les **polynômes multivariés** sont parmi les alternatives les plus populaires[4], mais nous explorons aussi deux propositions moins étudiées basées respectivement sur les **isogénies sur courbes elliptiques supersingulières** et sur les **groupes non abéliens**.

Chacun de ces domaines est un champ de recherche en soi, et chaque cryptosystème proposé (et demeurant à ce jour hors de portée d'une attaque classique ou quantique) présente des avantages et désavantages dépendants de son contexte d'utilisation et des paramètres utilisés. Par exemple, certains nécessitent de plus grandes clés que d'autres, alors que d'autres permettent des algorithmes de chiffrement plus rapides. Les implémentations spécifiques de protocoles, leurs paramètres et leur utilisabilité en pratique sont des notions déjà bien documentées (voir par exemple [39]) ; nous concentrerons plutôt nos efforts sur l'analyse des fonctions à sens unique sous-jacentes, dans le but de pouvoir comparer les catégories de cryptosystèmes sur la base de leur sécurité théorique, c'est-à-dire le plus haut niveau de sécurité que l'on peut espérer obtenir.

2.3.1 Cryptographie basée sur les codes

Dans la théorie des codes correcteurs, on encode généralement un mot qui pourrait prendre seulement k bits sur un nombre de bits n plus grand. Cela ajoute une redondance des données qui permet de déceler les erreurs de transmission possibles. Un tel code linéaire peut être défini à l'aide d'une matrice génératrice :

Définition 2.30. Un **code**-$[n, k]$ C est un sous-espace vectoriel de $\{0,1\}^n$ ayant k dimensions engendré par toute combinaison linéaire des rangées d'une **matrice génératrice G** de taille $k \times n$. En d'autres termes, $C = \{\mathbf{uG} \mid \mathbf{u} \in \{0,1\}^k\}$.

L'intérêt de décrire 2^k mots différents sur plus de k bits est que, si certains bits sont inversés, on ne tombe pas sur un autre mot du code qui possède une autre sémantique, mais plutôt sur un mot ne faisant pas partie du code, ce qui indique directement qu'il y a eu une erreur de transmission. Non seulement on peut détecter les erreurs, mais on peut aussi parfois les corriger, à condition que l'on connaisse une structure spécifique du code (souvent des codes de

4. L'autre alternative généralement mentionnée est la cryptographie basée sur les fonctions de hachage, mais est uniquement utilisable pour générer des signatures électroniques et non comme cryptosystème asymétrique.

Goppa, voir [10]) et que le nombre d'inversions de bits soit inférieur à un entier t dépendant de la structure.

C'est la structure du code qui permet l'existence d'un algorithme polynomial pour le décoder. Si l'on ne connaît pas cette structure, on fait face au problème suivant :

Définition 2.31.

Problème de McEliece	
Entrée :	Une matrice génératrice $\mathbf{G} \in \{0,1\}^{k\times n}$, provenant d'une permutation d'un code décodable mais d'apparence aléatoire, et un vecteur $\mathbf{c} \in \{0,1\}^n$
Promesse :	$\mathbf{c}$ a t bits inversés par rapport à un mot du code généré par $\mathbf{G}$
Question :	Quel est le vecteur $\mathbf{m} \in \{0,1\}^k$ tel que $\mathbf{mG} \oplus \mathbf{c}$ a t éléments valant 1 ? [10]

Supposons pour l'instant que ce problème est difficile et voyons comment l'on peut baser un cryptosystème dessus.

Définition 2.32. Avec des paramètres publics $k, n, t \in \mathbb{N}$, le **cryptosystème de McEliece** est un cryptosystème asymétrique à porte dérobée défini par $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

- $\mathcal{K}_{priv} = \{0,1\}^{k\times k} \times \{0,1\}^{k\times n} \times \{0,1\}^{n\times n}$;
- $\mathcal{K}_{pub} = \{0,1\}^{k\times n}$;
- $gen((\mathbf{S}, \mathbf{G}, \mathbf{P})) = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}$;
- $\mathcal{M} = \{0,1\}^k$;
- $\mathcal{C} = \{0,1\}^n$;
- $\mathcal{E} = \{E_{\mathbf{SGP}} | E_{\mathbf{SGP}}(\mathbf{m}) = \mathbf{m}(\mathbf{SGP}) \oplus \mathbf{z}$, où $\mathbf{z} \in_{\mathcal{U}} \{\mathbf{v} \in \{0,1\}^n \mid \mathbf{v}$ a t éléments valant $1\}\}$
- $\mathcal{D} = \{D_{\mathbf{SGP}} | D_{\mathbf{SGP}}(\mathbf{m}(\mathbf{SGP}) \oplus \mathbf{z}, (\mathbf{S}, \mathbf{G}, \mathbf{P})) = \mathbf{m}\}$ [46].

Les matrices $\mathbf{S}$, $\mathbf{G}$ et $\mathbf{P}$ doivent respectivement être aléatoire inversible, générer un code décodable en présence d'au plus t erreurs, et représenter une permutation.

Algorithme 22 Génération de la clé publique McEliece

1: **Entrée :** La clé privée composée des matrices $\mathbf{S} \in \{0,1\}^{k\times k}$, $\mathbf{G} \in \{0,1\}^{k\times n}$, $\mathbf{P} \in \{0,1\}^{n\times n}$
2: **Retourner** $\mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}$

Algorithme 23 Chiffrement McEliece

1: **Entrée :** La clé publique $\mathbf{SGP} \in \{0,1\}^{k\times n}$, le message $\mathbf{m} \in \{0,1\}^k$ et les informations publiques $k, n, t \in \mathbb{N}$
2: Calculer $\mathbf{m}(\mathbf{SGP})$
3: Sélectionner un $\mathbf{z} \in \{0,1\}^n$ ayant t éléments valant 1 au hasard
4: **Retourner** $\mathbf{c} = \mathbf{m}(\mathbf{SGP}) \oplus \mathbf{z}$, où $\oplus$ est le ou exclusif bit à bit

Algorithme 24 Déchiffrement McEliece

1: **Entrée :** La clé privée composée de $\mathbf{S} \in \{0,1\}^{k\times k}$, $\mathbf{G} \in \{0,1\}^{k\times n}$, $\mathbf{P} \in \{0,1\}^{n\times n}$, le message chiffré $\mathbf{c} \in \{0,1\}^n$ et les informations publiques $k, n, t \in \mathbb{N}$
2: Calculer $\mathbf{mSG} \oplus \mathbf{zP}^{-1} = \mathbf{cP}^{-1}$; il s'agit d'un vecteur du code décodable auquel on aurait ajouté le vecteur d'erreurs $\mathbf{zP}^{-1}$
3: Décoder $\mathbf{mSG} \oplus \mathbf{zP}^{-1}$ en $\mathbf{mSG}$
4: **Retourner** $\mathbf{m} = (\mathbf{mSG})\mathbf{G}^{-1}\mathbf{S}^{-1}$

La matrice génératrice menant à un code facilement décodable est cachée par une matrice de permutation, à laquelle l'espion n'a pas accès. Sans connaître cette matrice, l'espion fait face au problème de Renverser une permutation :

Définition 2.33.

Renverser une permutation	
Entrée :	Un oracle calculant une permutation π et un vecteur permuté $\mathbf{v}$
Question :	Que vaut $\pi^{-1}(\mathbf{v})$?

Conjecture 2.34. Renverser une permutation $\notin$ BQP [52]

En fait, les auteurs de [52] montrent que le problème est aussi difficile qu'une recherche non ordonnée, pour lequel l'algorithme de Grover est le meilleur algorithme connu et est insuffisamment puissant.

Proposition 2.35. *En supposant la conjecture 2.34, la génération de la clé publique McEliece est une fonction à sens unique post-quantique.*

Démonstration.

— La génération de la clé consiste en deux produits de matrices, ce qui se fait en temps polynomial.

— Retrouver $\mathbf{G}$ à partir de $\mathbf{SGP}$ nécessite d'inverser sans la connaître la matrice $\mathbf{P}$, qui est une matrice de permutation, ce qui revient à Renverser une permutation [5].

□

Conjecture 2.36. Problème de McEliece $\in$ NP $\setminus$ BQP.

Proposition 2.37. *En supposant la conjecture 2.36, le chiffrement McEliece est une fonction à porte dérobée post-quantique.*

Démonstration.

5. $\mathbf{S}$ n'a pas de fonction cryptographique, elle sert plutôt à assurer que la matrice génératrice n'ait pas un format systématique, où chaque rangée contiendrait exactement un mot de code concaténé avec des bits de redondance, ce qui révélerait de l'information [15].

— Le chiffrement consiste à multiplier un vecteur avec une matrice, puis à additionner (modulo 2) n éléments, ce qui se fait en temps polynomial.

— S'il ne connaît pas la clé privée, un espion doit déduire $\mathbf{m}$ à partir de $\mathbf{m}(\mathbf{SGP}) \oplus \mathbf{z}$. Comme il n'a aucune idée de quels t éléments de $\mathbf{z}$ valent 1, il doit soit procéder à un décodage, mais sans connaître la structure de $\mathbf{SGP}$ à cause de la permutation, ce qui revient au Problème de McEliece, soit renverser ladite permutation.

— Pour déchiffrer, il faut calculer l'inverse de la permutation $\mathbf{P}$ et la multiplier, deux opérations polynomiales, puis trouver le mot de code en enlevant les t erreurs, ce qui est possible en assumant que $\mathbf{G}$ représente un code décodable. Enfin, il faut encore inverser et multiplier deux matrices, toujours en temps polynomial.

□

Le Problème de McEliece est un cas particulier du très semblable Problème du décodage général :

Définition 2.38.

Problème du décodage général	
Entrée :	Une matrice génératrice $\mathbf{G} \in \{0,1\}^{k \times n}$ et un vecteur $\mathbf{c} \in \{0,1\}^n$
Promesse :	$\mathbf{c}$ est à t bits inversés d'un mot du code généré par $\mathbf{G}$
Question :	Quel est le vecteur $\mathbf{m} \in \{0,1\}^k$ tel que $\mathbf{mG} \oplus \mathbf{c}$ a t éléments valant 1 ? [9]

La seule différence est que dans le cas du problème du McEliece, le code est nécessairement la permutation d'un code décodable. Cela est suffisant pour qu'il ait été possible de montrer la NP-complétude du décodage général mais pas celle du problème McEliece. Néanmoins, par cette similitude avec un problème NP-complet, le cryptosystème semble avoir des bases de sécurité assez solides.

2.3.2 Cryptographie basée sur les réseaux euclidiens

Le réseau euclidien est un outil mathématique qui s'est révélé être un bon candidat pour la cryptographie post-quantique étant donnée la difficulté de plusieurs problèmes le concernant.

Définition 2.39. Un **réseau euclidien** $\mathcal{L}$ est un sous-espace vectoriel de $\mathbb{R}^n$ contenant toutes les combinaisons linéaires *entières* d'une base $\{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n\}$ de $\mathbb{R}^n$ donnée :

$$\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n) = \Big\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \Big\}$$

La figure 2.9 montre un réseau euclidien en deux dimensions. Tout endroit sur le plan est un point de $\mathbb{R}^n$, mais seuls les points noirs sont éléments de $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2)$. Notons que $\mathbf{b}_1$ et $\mathbf{b}_2$

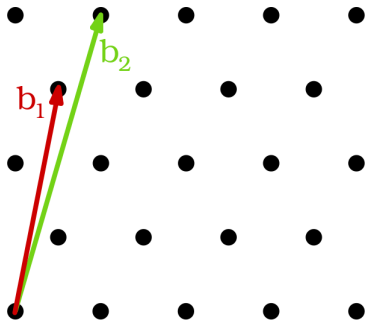


Figure 2.9 – Réseau euclidien en 2 dimensions avec base quelconque $\mathbf{b_1}, \mathbf{b_2}$

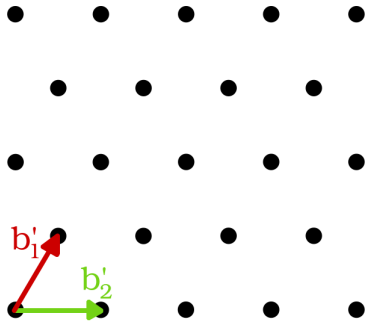


Figure 2.10 – Réseau euclidien en 2 dimensions avec base la plus courte $\mathbf{b'_1}, \mathbf{b'_2}$

forment bel et bien une base, mais qu'il serait possible d'en définir d'autres dont, en particulier, la base $\{\mathbf{b}'_1, \mathbf{b}'_2\}$ de la figure 2.10, ayant les vecteurs les plus courts possibles (ayant la plus petite norme euclidienne $\sqrt{\sum_{i=1}^{n} x_i^2}$). Trouver cette base plus concise est toutefois un problème qui est trop difficile :

Définition 2.40.

Plus courts vecteurs indépendants	
Entrée :	Un réseau euclidien $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n)$
Question :	Quelle est la base la plus concise $\mathbf{b}'_1, \mathbf{b}'_2, ..., \mathbf{b}'_n$ telle que $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n) = \mathcal{L}(\mathbf{b}'_1, \mathbf{b}'_2, ..., \mathbf{b}'_n)$?

Un problème encore plus simple, celui de trouver *le* vecteur le plus court (à l'exception du vecteur nul), est lui aussi difficile :

Définition 2.41.

Plus court vecteur	
Entrée :	Un réseau euclidien $\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n)$
Question :	Quel est le vecteur $\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i$, $v_i \in \mathbb{Z}$, ayant la plus petite norme non nulle ?

Les versions d'approximation Plus courts vecteurs indépendants (approximés) et Plus court vecteur (approximé), où l'on ne cherche pas exactement les plus courts vecteurs mais plutôt des vecteurs suffisamment courts –à un facteur constant près de la meilleure solution– semblent elles aussi assez difficiles.

En cryptographie, l'espace vectoriel a généralement les scalaires $\mathbb{Z}_q$, $q \in \mathbb{N}$, plutôt que $\mathbb{R}$. À partir d'une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, on peut définir le réseau euclidien suivant :

$$\mathcal{L}(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}_q^n \mid \mathbf{y} = \mathbf{As} \bmod q, \mathbf{s} \in \mathbb{Z}_q^m\}$$

Il s'agit de toutes les combinaisons linéaires entières des colonnes de $\mathbf{A}$, un réseau euclidien de n dimensions sur l'ensemble de scalaires $\mathbb{Z}_q$.

Un troisième problème, dont la difficulté sert d'hypothèse calculatoire au cryptosystème offrant la meilleure garantie de sécurité théorique parmi les cryptosystèmes basés sur les réseaux euclidiens, est l'Apprentissage avec erreurs :

Définition 2.42.

Apprentissage avec erreurs	
Entrée :	Une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ dont les éléments suivent une distribution **uniforme** sur $\mathbb{Z}_q$ et un vecteur $\mathbf{v} \in \mathbb{Z}_q^n$
Promesse :	$\mathbf{v} = \mathbf{As} + \mathbf{e}$, où $\mathbf{s} \in \mathbb{Z}_q^m$ suit une distribution **uniforme** et $\mathbf{e} \in \mathbb{Z}_q^n$ suit une distribution **normale**
Question :	Quel est le vecteur $\mathbf{s}$?

La question peut être reformulée ainsi : le vecteur $\mathbf{v}$ en entrée est un élément de $\mathcal{L}(\mathbf{A})$ ayant subi des perturbations suivant une loi normale ; quel est l'élément de $\mathcal{L}(\mathbf{A})$ original ? Une autre façon de le voir est qu'il faut trouver un vecteur secret $\mathbf{s}$ à partir de n résultats approximatifs de produits scalaires entre des vecteurs connus et le vecteur secret.

Définition 2.43. Étant donnés des paramètres $n, m, l, t, q \in \mathbb{N}$, le **cryptosystème basé sur l'apprentissage avec erreurs**, ou **LWE** pour *Learning With Errors*, est un cryptosystème asymétrique à porte dérobée défini par le tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

- $\mathcal{K}_{priv} = \mathbb{Z}_q^{n \times l}$;
- $\mathcal{K}_{pub} = \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times l}$;
- $gen(\mathbf{S}) = (\mathbf{A}, \mathbf{P})$, où $\mathbf{A} \in_{\mathcal{U}} \mathbf{Z}_q^{m \times n}$, et $\mathbf{P} = \mathbf{AS} + \mathbf{E}$, où $\mathbf{E} \in_{\mathcal{N}} \mathbb{Z}_q^{m \times l}$;
- $\mathcal{M} = \mathbb{Z}_t^l$;
- $\mathcal{C} = \mathbb{Z}_q^n \times \mathbb{Z}_q^l$;
- $\mathcal{E} = \{E_{(\mathbf{A},\mathbf{P})} | E_{(\mathbf{A},\mathbf{P})}(\mathbf{m}) = (\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil))$, où $\mathbf{a} \in_{\mathcal{U}} \{0,1\}^m$ et $\lfloor \mathbf{x} \rceil$ dénote le vecteur où chaque coordonnée de $\mathbf{x}$ est arrondie à l'entier le plus près ;
- $\mathcal{D} = \{D_{(\mathbf{A},\mathbf{P})} | D_{(\mathbf{A},\mathbf{P})}((\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil), \mathbf{S}) = \lfloor \frac{t}{q}(\mathbf{E}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil) \rceil\}$ [10] [59].

Algorithme 25 Génération de la clé publique LWE

1: **Entrée :** La clé privée $\mathbf{S} \in \mathbb{Z}_q^{n \times l}$
2: Sélectionner une matrice $\mathbf{A}$ de taille $m \times n$ en tirant chaque élément dans $\mathbb{Z}_q$ selon une distribution *uniforme*
3: Sélectionner une matrice $\mathbf{E}$ de taille $m \times l$ en tirant chaque élément dans $\mathbb{Z}_q$ selon une distribution *normale*
4: Calculer $\mathbf{P} = \mathbf{AS} + \mathbf{E}$
5: **Retourner** $(\mathbf{A}, \mathbf{P})$

Algorithme 26 Chiffrement LWE

1: **Entrée :** La clé publique $(\mathbf{A}, \mathbf{P})$, le message $\mathbf{m} \in \mathbb{Z}_t^l$ et les informations publiques $n, m, k, t, q \in \mathbb{N}$
2: Sélectionner un vecteur binaire $\mathbf{a}$ de taille m au hasard
3: Calculer $\mathbf{A}^T\mathbf{a}$, ce qui donne un vecteur de n éléments égal à la somme des rangées de $\mathbf{A}$ dont la coordonnée correspondante de $\mathbf{a}$ est 1.
4: Calculer $\mathbf{P}^T\mathbf{a}$
5: Calculer $\lfloor \frac{q}{t}\mathbf{m} \rceil$ en multipliant chaque coordonnée de $\mathbf{m}$ par $\frac{q}{t}$ et l'arrondissant à l'entier le plus près
6: **Retourner** $(\mathbf{A}^T\mathbf{a}, \mathbf{P}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil)$

Algorithme 27 Déchiffrement LWE

1: **Entrée :** La clé publique $(\mathbf{A}, \mathbf{P}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times l}$, la clé privée $\mathbf{S} \in \mathbb{Z}_q^{n \times l}$, le message chiffré $(\mathbf{c_1}, \mathbf{c_2}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$ et les informations publiques $n, m, k, t, q \in \mathbb{N}$
2: Calculer $(\mathbf{AS})^T\mathbf{a} = \mathbf{S}^T\mathbf{A}^T\mathbf{a} = \mathbf{S}^T\mathbf{c_1}$
3: Calculer $\mathbf{E}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil = (\mathbf{AS} + \mathbf{E})^T\mathbf{a} - (\mathbf{AS})^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil = \mathbf{P}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil - (\mathbf{AS})^T\mathbf{a} = \mathbf{c_2} - (\mathbf{AS})^T\mathbf{a}$
4: **Retourner** $\lfloor \frac{t}{q}(\mathbf{E}^T\mathbf{a} + \lfloor \frac{q}{t}\mathbf{m} \rceil) \rceil$

Notons que le déchiffrement ne retourne pas nécessairement le message $\mathbf{m}$ tel quel : chaque coordonnée a été transformée en un autre entier avec un arrondi, on lui a ajouté des erreurs suivant une distribution normale, puis on a arrondi de nouveau en faisant la transformation inverse. Avec le bon choix de paramètres, cependant, la probabilité d'erreur peut être rendue négligeable [59].

L'absence d'algorithmes connus –classiques comme quantiques– pour trouver de suffisamment courts vecteurs dans un réseau euclidien, malgré un grand effort de recherche dans ce domaine, permet d'avoir une bonne confiance en la conjecture suivante :

Conjecture 2.44. Plus court vecteur (approximé) et Plus courts vecteurs indépendants (approximés) ne sont pas dans BQP [38].

Or, le problème qui sert d'hypothèse calculatoire au cryptosystème serait au moins aussi difficile que ces problèmes pour un ordinateur quantique.

Théorème 2.45. Plus court vecteur (approx.) $\leq_q$ Apprentissage avec erreurs *et* Plus courts vecteurs indépendants (approx.) $\leq_q$ Apprentissage avec erreurs, *où* $\leq_q$ *représente une réduction polynomiale quantique* [59].

La réduction polynomiale quantique signifie que, en présence d'un oracle qui permettrait de résoudre Apprentissage avec erreurs en temps polynomial, on pourrait définir un algorithme quantique polynomial pour Plus court vecteur (approximé) et Plus courts vecteurs indépendants (approximés). Nous ne présenterons pas cet algorithme, mais donnons l'intuition derrière son fonctionnement :

— Supposons un oracle qui, étant donné un vecteur correspondant à un élément du réseau euclidien $\mathcal{L}$ ayant subi des perturbations, retourne l'élément de $\mathcal{L}$ le plus près ;

— En donnant une superposition de vecteurs quelconques à l'oracle, on obtient en sortie une superposition de plusieurs points du réseau euclidien ;

— Une transformée de Fourier quantique permet de passer de ces points à une description succincte du réseau euclidien.

La conjecture suivante découle de la conjecture 2.44 et du théorème 2.45 :

Conjecture 2.46. Apprentissage avec erreurs $\in \mathsf{NP} \setminus \mathsf{BQP}$

Proposition 2.47. *En supposant la conjecture 2.46, la génération de clé LWE est une fonction à sens unique post-quantique.*

Démonstration. — La génération de la clé consiste en un produit et une addition de matrices, ce qui se fait en temps polynomial.

— Si l'on peut retrouver $\mathbf{S}$ sachant $\mathbf{A}$ et $\mathbf{AS} + \mathbf{E}$, alors on peut résoudre Apprentissage avec erreurs, car il s'agit du cas particulier où $\mathbf{S}$ et $\mathbf{E}$ sont des vecteurs.

□

Proposition 2.48. *En supposant la conjecture 2.46, le chiffrement LWE est une fonction à porte dérobée post-quantique*[6].

Démonstration. — Le chiffrement consiste en des additions et multiplications d'entiers et de matrices, en plus d'une opération d'arrondi. Le tout se fait clairement en temps polynomial.

— Retrouver le message de façon illégitime signifie que l'on peut annuler toutes les perturbations qui ont été ajoutées à un vecteur du réseau euclidien ; cela revient à résoudre Apprentissage avec erreurs.

6. Comme le déchiffrement peut échouer, la fonction ne correspond pas exactement à notre définition de fonction à porte dérobée post-quantique, mais les implications sont les mêmes car la probabilité d'échec est négligeable.

— Le déchiffrement consiste, tout comme le chiffrement, en des additions et multiplications d'entiers et de matrices, en plus d'une opération d'arrondi.

□

2.3.3 Cryptographie basée sur les polynômes multivariés

Supposons n variables $x_1, x_2, ..., x_n$ prenant leur valeur dans un corps fini comme $\mathbb{Z}_p$. Supposons maintenant un polynôme de degré 2 défini sur ces variables, par exemple $ax_3^2 + bx_1x_2 + cx_n + d$ (avec $a, b, c, d \in \mathbb{Z}_q$) ou, de manière générale :

$$y = \sum_{i=1}^{n} a_i x_i^2 + \sum_{i=1}^{n} \sum_{j=1}^{i-1} b_{ij} x_i x_j + \sum_{i=1}^{n} c_i x_i, \text{ où } a_i, b_i j, c_i \in \mathbb{Z}_q \ \forall i, j$$

Connaissant les valeurs de $x_1, x_2, ..., x_n$, il est très facile d'évaluer la valeur de y. Imaginons maintenant que nous ayons m différentes équations de la sorte, toutes sur les mêmes variables ; il est seulement m fois plus long de calculer $y_1, y_2, ..., y_m$ que de calculer un seul y.

Renversons maintenant le problème : si l'on connaît des valeurs $y_1, y_2, ..., y_m \in \mathbb{Z}_p$, toutes égales au résultat d'une équation quadratique sur $f_k(x_1, x_2, ..., x_n)$, peut-on trouver des valeurs aux variables telles que chaque équation f_k donne y_k ? En d'autres termes, peut-on résoudre le *système d'équations quadratiques* donné ?

Système d'équations quadratiques	
Entrée :	m équations $y_k = \sum_{i=1}^{n} a_{ik} x_i^2 + \sum_{i=1}^{n} \sum_{j=1}^{i-1} b_{ijk} x_i x_j + \sum_{i=1}^{n} c_{ik} x_i$ où $y_k, a_{ik}, b_{ijk}, c_{ik} \in \mathbb{Z}_q \ \forall i, j, k$
Question :	Quelles valeurs de $x_1, x_2, ..., x_n$ rendent chaque équation vraie ?

Ce problème est, dans sa forme générale, NP-complet [25]. Cela donne un très bel exemple de fonction à sens unique : en connaissant les coefficients publics, on peut chiffrer un message $x_1, x_2, ...x_n$ en un texte chiffré $y_1, y_2, ...y_m$ (il faut prendre $m \geq n$ pour assurer que l'on retrouve un message unique au déchiffrement [56]). La difficulté dans la conception d'un tel cryptosystème consiste à choisir les coefficients de manière à obtenir une instance véritablement difficile, tout en permettant l'existence d'une structure secrète qui permet de retrouver les valeurs initiales en temps polynomial. Comme dans le cas de la cryptographie basée sur les codes (section 2.3.1), cette porte dérobée empêche de garantir que l'espion fait face à un problème NP-complet, mais il semble possible de cacher suffisamment la structure pour que le problème reste difficile [10].

Dans les cryptosystèmes proposés, la clé privée prend généralement la forme d'un système d'équations quadratiques $\mathcal{Q}$ choisi pour être facile à inverser, auquel on applique deux transformations linéaires S et T pour obtenir la clé publique $S \cdot \mathcal{Q} \cdot T$, un système d'apparence difficile à résoudre.

Plusieurs protocoles de chiffrement asymétrique basés sur les polynômes multivariés, jadis prometteurs, ont finalement été cryptanalysés [20] [30], mais certains sont toujours en vie (par exemple [57]).

2.3.4 Cryptographie basée sur les isogénies sur courbes elliptiques supersingulières

Nous avons vu que l'algorithme de Shor rendait obsolète le protocole El Gamal sur les courbes elliptiques, car il est capable de calculer le logarithme discret sur le groupe abélien composé des points de la courbe avec une opération d'addition bien spécifique (voir section 1.3.4 pour un rappel sur les courbes elliptiques). Cela ne rend toutefois pas l'utilisation de courbes elliptiques en cryptographie complètement irréalisable, mais il ne faut pas réutiliser la même structure algébrique.

Cette fois, les éléments ne sont pas les points d'une courbe, mais bien des *transformations* entre courbes (supersingulières [7]), nommées isogénies.

Définition 2.49. Une **isogénie** de $\mathbf{E_1}$ à $\mathbf{E_2}$ (des courbes elliptiques) est une fonction $\phi : \mathbf{E_1} \to \mathbf{E_2}$ surjective où $\phi(P_{\infty_1}) = P_{\infty_2}$ et $\phi(P +_{\mathbf{E_1}} Q) = \phi(P) +_{\mathbf{E_2}} \phi(Q)\ \forall P, Q \in E_1$.

L'opération entre isogénies est la composition. À la différence de la multiplication utilisée dans El Gamal, il ne s'agit pas d'une opération commutative. C'est d'ailleurs l'absence d'une telle structure qui empêche l'algorithme de Shor de briser ce type de cryptographie [34]. Cependant, c'est aussi grâce à la commutativité qu'il est généralement facile de créer un cryptosystème asymétrique (alors qu'Alice calcule l'opération f puis l'opération g, Bob calcule g puis f, ce qui mène au même résultat).

Les courbes elliptiques sont toutefois dotées d'un paramètre qui s'avère crucial pour régler ce problème : le j-invariant.

Définition 2.50. Le **j-invariant** d'une courbe $y^2 = x^3 + ax + b$ est donné par

$$1728\frac{4a^3}{4a^3 + 27b^2}$$

Le j-invariant est préservé par les isogénies, de sorte qu'on peut appliquer de telles fonctions dans un ordre différent et tout de même retrouver la même valeur par la suite [23].

Le protocole commence donc avec une courbe E publique, que Bob peut transformer en $\phi_B(E)$ où l'isogénie ϕ_B demeure privée. Alice reçoit la clé publique $\phi_B(E)$ et calcule $\phi_A\phi_B(E)$ suivi du j-invariant correspondant, noté $j(\phi_A\phi_B(E))$, un nombre d'apparence aléatoire avec lequel

7. Ne pas se laisser intimider par ce terme, il s'agit d'une technicalité dont nous ne tiendrons pas compte, mais qui est nécessaire pour la sécurité puisqu'un algorithme efficace existe pour des courbes ordinaires [17].

elle peut masquer son message et obtenir le texte chiffré $c = m \oplus j(\phi_A\phi_B(E))$ [8]. Parallèlement, elle calcule $\phi_A(E)$ puis envoie $(\phi_A(E), c)$ à Bob. Ce dernier retrouve le message ainsi :

$$m = c \oplus j(\phi_B\phi_A(E))$$

L'important est que, bien que $\phi_A\phi_B(E) \neq \phi_B\phi_A(E)$ (la composition d'isogénie n'étant pas commutative), il s'agit de courbes elliptiques isomorphes et ont donc le même j-invariant. C'est cette propriété spécifique aux isogénies sur courbes elliptiques qui permet d'envisager ce domaine pour la cryptographie post-quantique.

Nous devons admettre que nous avons omis plusieurs détails importants pour implanter un véritable protocole (voir [34] pour le protocole à clé publique détaillé). En effet, les notions théoriques supportant la cryptographie basée sur les isogénies sont plutôt avancées, ce qui éveille des soupçons quant à sa sécurité : obtenir une compréhension profonde de ce domaine n'est pas nécessairement à la portée des concepteurs d'algorithmes quantiques, qui sont spécialisés dans un domaine tout à fait différent. En conséquence, les attaques quantiques (et même classiques !) envers ce protocole n'ont pas eu toute l'attention qu'elles méritaient [23].

On peut malgré tout cerner un problème général qu'un attaquant devrait résoudre pour briser le protocole :

Problème de griffe	
Entrée :	Des fonctions $f : X \to Z$ et $g : Y \to Z$
Question :	Quelle paire $(x, y) \in X \times Y$ est telle que $f(x) = g(y)$?

Notons que l'attaquant a accès aux descriptions de $\phi_A(E)$ et $\phi_B(E)$. S'il pouvait résoudre le Problème de griffe, alors en prenant X et Y comme l'ensemble des isogénies obtenables respectivement à partir de $\phi_A(E)$ et $\phi_B(E)$, f et g comme des fonctions calculant le j-invariant de la courbe suite à l'application de l'isogénie, et Z comme étant le corps $\mathbb{K}$ sous-jacent aux courbes elliptiques, il pourrait directement trouver le j-invariant secret [3]. Le meilleur algorithme quantique pour ce problème inventé à ce jour s'exécute toutefois en $\mathcal{O}(\sqrt[6]{p})$, où p est la taille du corps. Cela demeure exponentiel en le nombre de bits décrivant p [66].

2.3.5 Cryptographie basée sur les groupes non abéliens

Les isogénies sur courbes elliptiques supersingulières amènent déjà l'idée d'utiliser une opération non commutative pour former un groupe non abélien. La technique utilisée est toutefois seulement possible avec ce groupe précis, puisque la notion d'avoir deux courbes différentes ayant le même j-invariant est cruciale. Nous présentons ici un système plus général pour la

8. Rappelons que l'utilisation du *ou exclusif* $\oplus$ avec une clé utilisée une seule fois masque parfaitement le message (voir section 1.1.2).

cryptographie basée sur les groupes non abéliens, qui nécessite pour sa part des sous-groupes ayant une propriété particulière.

Soit un groupe G non abélien (i.e. $\exists g_1, g_2 \in G$ tel que $g_1 g_2 \neq g_2 g_1$), et supposons qu'il existe deux sous-groupes A et B de G dont l'intersection contient uniquement l'élément neutre de G, strictement plus petits que G et tels que $\forall a \in A, b \in B$, $ab = ba$. Muni d'un tel groupe et d'un élément $x \in G$ public, il est possible de définir le cryptosystème El Gamal non commutatif [36].

Algorithme 28 Génération de la clé publique El Gamal non commutatif

1: **Entrée :** La clé privée $b \in B$
2: Calculer $z = b^{-1}xb$
3: **Retourner** z

Algorithme 29 Chiffrement El Gamal non commutatif

1: **Entrée :** La clé publique $z \in G$, le message $m \in \{0,1\}^*$ et les informations publiques G, $x \in G$
2: Sélectionner $a \in A$ au hasard
3: Calculer $a^{-1}xa$
4: Calculer $b^{-1}a^{-1}xab = a^{-1}b^{-1}xba = a^{-1}za$
5: **Retourner** $(a^{-1}xa, b^{-1}a^{-1}xab \oplus m)$

Algorithme 30 Déchiffrement El Gamal non commutatif

1: **Entrée :** La clé publique $z \in G$, la clé privée $b \in B$, le message chiffré $(c_1, c_2) \in G \times G$ et les informations publiques G, $x \in G$
2: Calculer $b^{-1}a^{-1}xab = b^{-1}c_1 b$
3: **Retourner** $(b^{-1}a^{-1}xab) \oplus c_2$

Un exemple de groupe utilisé dans l'implémentation de ce protocole est le groupe de tresses (voir par exemple [24]) car son opération est non commutative et il est facile de séparer le groupe en deux sous-groupes dont les éléments commutent deux à deux. L'article [45] montre bien comment séparer le groupe.

Briser ce protocole se réduit clairement au problème calculatoire suivant :

Recherche de conjugué	
Entrée :	$x, y \in G$
Question :	Quel $c \in G$ est tel que $y = c^{-1}xc$?

En effet, si un espion pouvait résoudre ce problème, il pourrait directement renverser la fonction à sens unique utilisée pour générer la clé privée.

Il est toutefois montré dans [65] que, même si ce problème s'avère trop difficile, il pourrait être possible de briser le cryptosystème en profitant de la porte dérobée, si l'on peut résoudre ce problème :

Problème de décomposition	
Entrée :	$x, y \in G$ et un sous-groupe A
Question :	Quels $a_1, a_2 \in A$ sont tels que $y = a_1 x a_2$?

Notons que Problème de décomposition est nécessairement au moins aussi facile que Recherche de conjugué, car la réponse à ce dernier est une réponse au premier. L'espion, qui connaît x, $a^{-1}xa$ et $b^{-1}xb$, n'a qu'à trouver $a_1, a_2 \in A$ tels que $a_1 x a_2 = a^{-1}xa$ et calculer $a_1 b^{-1} x b a_2$. Par commutativité, $a_1 b^{-1} = b^{-1} a_1$ et $a_2 b = b a_2$, donc il obtient $b^{-1} a_1 x a_2 b$. Or, $a_1 x a_2 = a^{-1}xa$, donc il a vraiment trouvé $b^{-1}a^{-1}xab$, ce qui lui donne le message directement.

Horizon III :
L'ère quantique

Laissons quelques années aux ingénieurs pour miniaturiser et rendre peu coûteux les appareils de calcul quantique ; supposons qu'il faut attendre jusqu'en **2043** pour que ce qui était jadis une nouveauté soit maintenant très répandu et utilisable par essentiellement tout le monde.

À l'horizon précédent, nous supposions un ordinateur quantique dans les mains d'un espion, donc utilisé en mode *attaque*. Est-ce que les mêmes particularités de la physique quantique peuvent être utilisées pour se *défendre* de ces attaques ?

L'horizon III explore deux façons d'utiliser le paradigme quantique dans la conception de protocoles cryptographiques. La première suppose simplement qu'il est possible de générer des qubits et de les partager sur un canal préservant leur état ; cela ne nécessite pas d'avoir un ordinateur quantique à proprement parler, le calcul peut être fait de façon classique [1]. La seconde suppose qu'Alice et Bob ont eux-mêmes en leur possession des ordinateurs quantiques complets, leur permettant de résoudre des problèmes auparavant difficiles lors des opérations de génération des clés, de chiffrement et de déchiffrement.

En dernier lieu, nous explorons la question suivante : d'un point de vue théorique, avec toutes ces options disponibles, de RSA à la cryptographie à l'aide d'ordinateurs quantiques, en passant par les protocoles post-quantiques et la cryptographie sur canaux quantiques, quels protocoles s'annoncent les plus sécuritaires ?

1. Nous n'avons donc pas vraiment à attendre l'an 2043 pour ce type de cryptographie ; en fait certaines implémentations sont déjà accessibles sur le marché mais sont très difficiles à mettre en œuvre.

3.1 Cryptographie sur canaux quantiques

3.1.1 Hypothèses physiques

Pour ce type de cryptographie, on suppose qu'Alice et Bob sont capables de générer, mesurer et transmettre des qubits.

On suppose également que l'espionne Ève est un ordinateur quantique et peut mesurer les qubits communiqués sur le fil. Cela vient toutefois avec une contrainte car, rappelons-nous, la mesure d'un qubit superposé entraîne son effondrement sur un état de la base utilisée pour la mesure (voir énoncé 2.4).

Peut-elle simplement copier ce qui passe sur le fil et le mesurer plus tard ? La réponse est définitivement *non*, à cause du théorème d'impossibilité du clonage quantique :

Théorème 3.1. *Aucune porte quantique U ne peut être telle que*

$$\exists \left|s\right\rangle \in \mathcal{H},\ \forall \left|\psi\right\rangle \in \mathcal{H},\ U(\left|\psi\right\rangle \left|s\right\rangle) = \left|\psi\right\rangle \left|\psi\right\rangle$$

Démonstration. Supposons au contraire qu'une telle porte U existe, avec un $\left|s\right\rangle$ adéquatement choisi. Soit $\left|\alpha\right\rangle, \left|\beta\right\rangle \in \mathcal{H}$. Alors on a

- $U(\left|\alpha\right\rangle \left|s\right\rangle) = \left|\alpha\right\rangle \left|\alpha\right\rangle$ et
- $U(\left|\beta\right\rangle \left|s\right\rangle) = \left|\beta\right\rangle \left|\beta\right\rangle$, ce qui équivaut à $(\left|s\right\rangle^\dagger \left|\beta\right\rangle^\dagger)U^\dagger = \left|\beta\right\rangle^\dagger \left|\beta\right\rangle^\dagger$ où † signifie la conjuguée transposée.

Effectuons le produit scalaire suivant :

$$(\left|s\right\rangle^\dagger \left|\beta\right\rangle^\dagger)U^\dagger \cdot U(\left|\alpha\right\rangle \left|s\right\rangle) = \left|\beta\right\rangle^\dagger \left|\beta\right\rangle^\dagger \cdot \left|\alpha\right\rangle \left|\alpha\right\rangle$$

Tout d'abord, par unitarité de U, on a que $U^\dagger U = \mathbb{1}$, donc cela revient à

$$\left|s\right\rangle^\dagger \left|\beta\right\rangle^\dagger \cdot \left|\alpha\right\rangle \left|s\right\rangle = \left|\beta\right\rangle^\dagger \left|\beta\right\rangle^\dagger \cdot \left|\alpha\right\rangle \left|\alpha\right\rangle$$

Posons $\left|\beta\right\rangle^\dagger \left|\alpha\right\rangle = c \in \mathbb{C}$. Alors on obtient

$$\left|s\right\rangle^\dagger c \left|s\right\rangle = \left|\beta\right\rangle^\dagger c \left|\alpha\right\rangle$$

Étant un scalaire, on peut mettre c en évidence, ce qui donne

$$c \left|s\right\rangle^\dagger \left|s\right\rangle = c \left|\beta\right\rangle^\dagger \left|\alpha\right\rangle$$

Comme $\left|s\right\rangle^\dagger \left|s\right\rangle = 1$ par définition du produit scalaire, on obtient finalement

$$c = c^2$$

Or, seuls $c = 0$ et $c = 1$ permettent l'équation $c = c^2$.

— Si $c = |\beta\rangle |\alpha\rangle = 0$ alors $|\beta\rangle$ et $|\alpha\rangle$ sont *orthogonaux*.

— Si $c = |\beta\rangle |\alpha\rangle = 1$ alors $|\beta\rangle = |\alpha\rangle$.

Il est impossible de cloner des états $|\alpha\rangle$ et $|\beta\rangle$ différents mais non orthogonaux, alors la porte U qui clone n'importe quel état ne peut exister [53]. □

Ne pouvant cloner ce qui passe sur le fil, Ève ne peut obtenir de l'information sur les qubits communiqués sans laisser de trace que si elle mesure chaque qubit selon une base dans laquelle le qubit est un vecteur de base. Si Alice et Bob utilisent tantôt la base $\{|0\rangle, |1\rangle\}$, tantôt la base $\{\frac{|0\rangle+|1\rangle}{\sqrt{2}}, \frac{|0\rangle-|1\rangle}{\sqrt{2}}\}$, il devient très peu probable qu'Ève mesure toujours selon la bonne base et ses actions pourront ainsi être détectées.

3.1.2 BB84

L'idée de profiter du fait qu'un attaquant ne peut observer une série de qubits sans la perturber pour partager un secret commun date du protocole BB84, proposé en 1984 par Brassard et Bennett [8]. Bien d'autres protocoles de ce genre existent, il s'agit principalement de variations de BB84 dont les améliorations en sécurité ne sont pas significatives d'un point de vue théorique [13].

Le protocole nécessite quelques interactions de plus que les protocoles présentés précédemment, où Bob envoyait simplement sa clé publique et Alice le message chiffré. Les paramètres utilisés sont résumés dans le tableau 3.1.

Au départ, Bob génère aléatoirement deux chaînes de bits $a, b \in \{0,1\}^n$ et génère en conséquence la chaîne de qubits suivante :

$$\left|\psi^{(n)}\right\rangle = \bigotimes_{i=1}^{n} |\psi_{a_i b_i}\rangle \text{, où } |\psi_{a_i b_i}\rangle = \begin{cases} |0\rangle & \text{si } a_i = b_i = 0 \\ |1\rangle & \text{si } a_i = 1, b_i = 0 \\ \frac{|0\rangle+|1\rangle}{\sqrt{2}} & \text{si } a_i = 0, b_i = 1 \\ \frac{|0\rangle-|1\rangle}{\sqrt{2}} & \text{si } a_i = b_i = 1 \end{cases}$$

On remarque que la chaîne b détermine quelle base orthogonale est utilisée, alors que a servira de chaîne binaire aléatoire à partager secrètement. Néanmoins, Bob envoie $\left|\psi^{(n)}\right\rangle$ à Alice via le canal quantique, qu'elle mesure du mieux qu'elle peut, c'est-à-dire arbitrairement. En effet, elle génère au hasard une chaîne $b' \in \{0,1\}^n$ qui détermine selon quelle base elle mesure.

Si $b'_i = 0$, elle mesure selon la base $\{|0\rangle, |1\rangle\}$. Si par chance, $b_i = 0$ également, alors elle obtient le bon résultat et peut déduire a_i (0 si elle mesure $|0\rangle$, 1 si $|1\rangle$ est mesuré). Il en va de façon analogue si $b'_i = b_i = 1$. Mais une fois sur deux, $b'_i \neq b_i$ et elle mesure par exemple $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ en $|0\rangle$ ou en $|1\rangle$, ce qui ne permet pas de déduire a_i avec plus de 50% de probabilité.

Paramètre	Signification
a	Chaîne binaire de taille n générée par Bob dont une partie doit devenir le secret s
a'	Interprétation de a par Alice suite à des mesures partiellement erronées
b	Chaîne binaire de taille n indiquant la base de chaque qubit de $\|\psi\rangle$
b'	Chaîne binaire de taille n indiquant dans quelle base Alice mesure $\|\psi\rangle$
c	Message chiffré transmis classiquement
i	Index quelconque
J	Ensemble d'index de taille l pour vérifier l'absence d'espion
j	Élément de J
k	Partie de a mesurée correctement (noté k_A pour Alice et k_B pour Bob, si leurs valeurs sont différentes)
l	Paramètre définissant la probabilité de détection d'espionnage
m	Message à transmettre (en binaire)
n	Nombre de qubits communiqués (taille de $\|\psi\rangle$)
s	Chaîne partagée secrètement entre Alice et Bob, utilisée comme clé de chiffrement
t	Taille du message m
$\|\psi\rangle$	Chaîne de qubits

Figure 3.1 – Paramètres du protocole BB84

Alice envoie alors b' pour indiquer à Bob avec quelles bases elle a mesuré chaque qubit. Bob calcule $b \oplus b'$, une chaîne telle que $(b \oplus b')_i = 1$ indique que a_i est potentiellement erroné, et l'envoie à Alice. Chacun de leur côté, Alice et Bob calculent respectivement les chaînes k_A et k_B (ayant probablement environ $\frac{n}{2}$ bits) qui consistent en tous les a_i déduits correctement concaténés entre eux.

Si tout s'est déroulé correctement, on devrait avoir $k_A = k_B$. Mais si un espion avait mesuré la chaîne de qubits, il est possible que certains a_i reçus par Alice aient été perturbés. Pour cette raison, Bob envoie aussi une série de $l < |k_B|$ couples différents (j, k_{Bj}). Alice peut ainsi vérifier que pour chaque index reçu, $k_{Aj} = k_{Bj}$. S'il existe un index pour lequel ce n'est pas le cas, cela indique la présence d'une perturbation lors du transfert de la chaîne de qubits. Dans le cas contraire, il est probable qu'aucune perturbation n'ait eu lieu, et on peut donc assumer que $k_{Aj} = k_{Bj}$ pour les $|k_B| - l$ index restants. Chacun de leur côté, Alice et Bob peuvent concaténer ces bits en un secret commun s.

Par la suite, Alice transmet son message $m \in \{0,1\}^t$ [2] en envoyant $c = m \oplus s$, puis Bob calcule simplement $m = c \oplus s$. Le schéma est résumé à la figure 3.2.

Nous présentons maintenant un exemple d'exécution non perturbée et un autre où Ève a tenté

2. On assume $t = |s|$; si $t < |s|$ on peut ignorer les derniers bits de s, tandis que si $t > |s|$ il suffit de répéter le protocole en boucle jusqu'à ce que le message soit entièrement transmis.

Alice	(Ève)	Bob
Au départ : $m \in \{0,1\}^t$, $t \leq \frac{n}{2} - l$		Au départ : $a, b \in \{0,1\}^n$
		Génération de $\left\|\psi^{(n)}\right\rangle = \bigotimes_{i=1}^{n} \left\|\psi_{a_i b_i}\right\rangle$
	$\leftarrow \left\|\psi^{(n)}\right\rangle$	
Génération de $b' \in \{0,1\}^n$		
a' : **Mesure** de $\left\|\psi^{(n)}\right\rangle$ selon b'		
	$b' \rightarrow$	Calcul de $b \oplus b'$
	$\leftarrow b \oplus b'$	
k_A : **Concaténation** des a'_i t.q. $(b \oplus b')_i = 0$		k_B : **Concaténation** des a_i t.q. $(b \oplus b')_i = 0$
		Calcul d'un sous-ensemble $J \subseteq \{1, ..., \|k_B\|\}$, où $\|J\| = l$
	$\leftarrow (j, k_{Bj})$, $\forall j \in J$	
Vérification que $k_{Aj} = k_{Bj}, \forall j \in J$		
Si ce n'est pas le cas, **arrêter**		
s : **Concaténation** des k_{Ai} t.q. $i \notin J$		s : **Concaténation** des k_{Bi} t.q. $i \notin J$
	$c = m \oplus s \rightarrow$	
		Déchiffrement de $c \oplus s$ en m

Figure 3.2 – Schéma d'exécution du protocole BB84

de mesurer les qubits. Nous utilisons les notations $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ et $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$ par souci de concision. Dans les deux cas, on prend $n = 8$, $l = 2$ et le message à transmettre m est la chaîne binaire 11.

Exemple 3.2 (Exécution non perturbée)**.** Bob génère $a = 01101110$ et $b = 00111100$, ce qui implique $\left\|\psi^{(n)}\right\rangle = |0\rangle\,|1\rangle\,|-\rangle\,|+\rangle\,|-\rangle\,|-\rangle\,|1\rangle\,|0\rangle$. À la réception de cette chaîne de qubits, Alice la mesure suivant $b' = 10110001$, ce qui lui donne $a' = ?110??1? = 01100111$ (les bits où nous avons d'abord placé des points d'interrogation sont obtenus au hasard car mesuré selon la mauvaise base, alors que les autres sont nécessairement mesurés tels que présentés).

Alice envoie b' à Bob, qui calcule et partage $b \oplus b' = 00111100 \oplus 10110001 = 10001101$, signifiant qu'ils doivent conserver les index 2, 3, 4 et 7 de a et a'. Cela donne $k = 1101$ dans les deux cas.

Bob choisit alors arbitrairement les index $\{2, 4\}$ et envoie $\{(2, 1), (4, 1)\}$, ce qui laisse les index $\{1, 3\}$ de k qu'il concatène pour former le secret $s = 10$. Alice vérifie que $k_2 = 1$ et $k_4 = 1$ de son côté également, puis forme à son tour $s = 10$, qui lui permet d'envoyer le message chiffré $c = m \oplus s = 11 \oplus 10 = 01$. Bob déchiffre cela en $m = c \oplus s = 01 \oplus 10 = 11$.

Exemple 3.3 (Exécution perturbée)**.** Prenons les mêmes chaînes $a = 01101110$, $b = 00111100$

et $b' = 10110001$, mais cette fois en supposant qu'Ève mesure la chaîne de qubits. Sans perte de généralité, supposons qu'elle effectue toutes ses mesures selon la base $\{|0\rangle, |1\rangle\}$. Elle fait ainsi s'effondrer la chaîne $|0\rangle |1\rangle |-\rangle |+\rangle |-\rangle |-\rangle |1\rangle |0\rangle$ sur $|0\rangle |1\rangle |?\rangle |?\rangle |?\rangle |?\rangle |1\rangle |0\rangle = |0\rangle |1\rangle |0\rangle |1\rangle |1\rangle |1\rangle |1\rangle |0\rangle$. Alice mesure toujours suivant b', obtenant $a' = ?1??111? = 01001111$.

Avec le même $b \oplus b' = 10001101$, Bob obtient encore $k_B = 1101$. Alice obtient toutefois $k_A = a'_2 a'_3 a'_4 a'_7 = 1001$. Supposant que Bob envoie encore $\{(2,1),(4,1)\}$, Alice vérifie que $k_{A2} = 1$ et $k_{A4} = 1$ ce qui, surprise, n'est pas le cas ! Alice et Bob conviennent donc d'interrompre leur protocole avant que le message ne soit compromis.

3.1.3 Sécurité inconditionnelle du protocole BB84

Un cas non souhaité peut survenir. Supposons qu'à l'exemple 3.3, les index vérifiés aient été 3 et 4 plutôt que 2 et 4. Alors Alice n'aurait pas remarqué d'incohérence et aurait calculé un secret différent de celui de Bob et le déchiffrement aurait donné un message incorrect (en l'occurence $m' = (m \oplus s_A) \oplus s_B = (11 \oplus 10) \oplus 11 = 10 \neq m$).

Pour un qubit donné, la probabilité qu'Ève choisisse la bonne base est de $\frac{1}{2}$. Même dans le cas où elle choisit la mauvaise, elle a une autre probabilité de $\frac{1}{2}$ d'obtenir le bon bit, ce qui implique qu'environ les trois quarts des bits mesurés l'ont été correctement. Ainsi, chacun des l bits sélectionnés a une probabilité de $\frac{3}{4}$ d'échouer à aider à remarquer la perturbation. La vérification fonctionne donc avec probabilité $1 - (\frac{3}{4})^l$, ce qui est certes bien peu dans notre cas où $l = 2$, mais qui, puisque $\frac{3}{4}$ est une constante, peut devenir arbitrairement près de 1 pour des paramètres assez grands.

L'exactitude du protocole n'est donc pas tout à fait certaine, mais la probabilité d'échec est négligeable. Qu'en est-il de la sécurité ? Supposons qu'Ève est *très* chanceuse et mesure chaque qubit selon la bonne base, ne causant aucune perturbation (cela est beaucoup demandé, puisque ça survient avec probabilité $\frac{1}{2^n}$). Alors elle pourra imiter toutes les opérations d'Alice et ainsi aboutir au même secret s, qui lui permettra de retrouver le bon message $m = c \oplus s$.

Ceci dit, Ève n'a aucun moyen de savoir qu'elle a par miracle tout bien mesuré ! Contrairement au cas où la clé est la solution à un problème calculatoire, elle n'a aucune façon de confirmer qu'il s'agit de la bonne clé, même en disposant d'un *temps infini* ! En effet, remarquons que la probabilité de tout bien mesurer est égale à la probabilité de deviner la clé par hasard même sans prendre de mesure. Or, cette probabilité de $\frac{1}{2^n}$ est synonyme d'une distribution uniforme de l'espace de clé, condition qui nous permet d'affirmer que nous sommes en présence d'un **secret parfait** (section 1.1.2).

3.2 Cryptographie à l'aide d'ordinateurs quantiques

3.2.1 Le retour du Logarithme discret et de la Somme de sous-ensembles

Un *cryptosystème à clé publique quantique*, que l'on peut décrire comme un protocole asymétrique standard où Alice et Bob peuvent, comme Ève, effectuer des opérations dans BQP, réutilise le schéma d'interaction simple que nous avons employé pour les horizons I et II. Dans le protocole d'Okamoto, Tanaka et Uchiyama (noté OTU) [54], certaines opérations doivent être effectuées avec un ordinateur quantique, mais toute la communication peut se faire sur un canal classique.

Selon nos connaissances actuelles, l'ordinateur quantique n'amène un gain superpolynomial que pour quelques rares problèmes comme la factorisation et le logarithme discret. C'est donc sans surprise que l'incorporation d'un algorithme pour le logarithme discret dans la génération de clé est la nouveauté majeure proposée par les auteurs. Contrairement au protocole El Gamal (section 1.3.3), on n'utilise pas le logarithme discret pour concevoir une fonction à sens unique, mais plutôt comme fonction non linéaire servant à cacher la structure particulière des éléments dans une instance d'un autre problème. La fonction

$$\begin{aligned} f : \mathbb{Z}_p &\to \mathbb{Z}_{p-1} \\ x &\mapsto y \text{ tel que } x = g^y \mod p \end{aligned}$$

réagit en effet de façon imprévisible pour qui ne connaît pas g et p. Ces paramètres pourront donc nous servir de porte dérobée.

Le problème auquel l'espion fait face lorsqu'il tente d'attaquer le cryptosystème doit, à l'instar des protocoles post-quantiques, être hors de BQP : c'est le problème NP-complet Somme de sous-ensembles qui sert d'hypothèse calculatoire dans ce cas-ci. Ce n'est pas la première fois que ce problème est utilisé pour concevoir un cryptosystème, mais les précédentes tentatives ont finalement mené à une cryptanalyse rapide [62] [67].

Nous avons précédemment présenté une fonction à porte dérobée basée sur Somme de sous-ensembles (exemple 1.21), où la structure secrète des éléments était un ordre super-croissant, que l'on cachait avec une simple multiplication modulo un nombre premier pour donner une apparence aléatoire. Chaque élément x était transformé ainsi :

$$x \mapsto xt \mod p$$

où t était secret. Pour éviter l'existence d'une cryptanalyse facile, on garde secrets les éléments

g, p, d et on transforme x ainsi [3] :

$$x \mapsto (y + d) \mod p - 1 \text{ tel que } x = g^y \mod p$$

3.2.2 Le cryptosystème OTU

Nous présentons une version plus accessible du protocole présenté dans [54], dont la version générale demande des notions avancées d'algèbre, mais qui demeure tout de même sécuritaire.

Définition 3.4. Étant donné des paramètres publics $n, k \in \mathbb{N}$, le **cryptosystème OTU** est un cryptosystème asymétrique à porte dérobée défini par le tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}_{pub}, \mathcal{K}_{priv}, gen, \mathcal{E}, D)$, où :

- $\mathcal{K}_{priv} = \mathbb{P} \times \mathbb{Z}_p \times \mathbb{Z}_{p-1}$;
- $\mathcal{K}_{pub} = \mathbb{Z}_{p-1}^n$;
- $gen((p, g, d)) = b_1, ..., b_n$, où $b_i = a_i + d$, a_i est tel que $p_i = g^{a_i} \mod p$, $p_i \in_{\mathcal{U}} \mathbb{Z}_p$;
- $\mathcal{M} = \{0, 1\}^n$ avec exactement k occurrences de 1 ;
- $\mathcal{C} = \mathbb{Z}$;
- $\mathcal{E} = \{E_{b_1,...b_n} | E_{b_1,...,b_n}(m) = \sum_{i=1}^n m_i b_i\}$
- $\mathcal{D} = \{D_{b_1,...b_n} | D_{b_1,...b_n}(\sum_{i=1}^n m_i b_i, (p, g, d)) = m\}$ [54].

Algorithme 31 Génération de la clé publique OTU

1: **Entrée :** La clé privée composée des entiers $p \in \mathbb{P}$, $g \in \mathbb{Z}_p$ et $d \in \mathbb{Z}_{p-1}$ et les informations publiques $n, k \in \mathbb{N}^+$
2: Sélectionner au hasard $p_1, p_2, ..., p_n \in \mathbb{Z}_p$, où $pgcd(p_i, p_j) = 1 \ \forall i, j$ tels que $1 \leq i < j \leq n$ et où le produit de k éléments de $\{p_1, p_2, ..., p_n\}$ est inférieur à p
3: Utiliser l'**algorithme de Shor** n fois pour obtenir $a_1, a_2, ..., a_n \in \mathbb{Z}_{p-1}$, où $p_i = g^{a_i} \mod p$
4: **Retourner :** $b_1, b_2, ..., b_n$ tels que $b_i = a_i + d \mod p - 1$

Algorithme 32 Chiffrement OTU

1: **Entrée :** La clé publique $b_1, b_2, ..., b_n \in \mathbb{Z}_{p-1}$, le message $m \in \{0, 1\}^n$ ayant précisément k occurrences de 1 et les informations publiques $n, k \in \mathbb{N}^+$
2: **Retourner** $\sum_{i=1}^n m_i b_i$

3. Rappelons que l'ordre de $\mathbb{Z}_p$ est $p - 1$ si p est premier.

Algorithme 33 Déchiffrement OTU

1: **Entrée :** La clé publique $b_1, b_2, ..., b_n \in \mathbb{Z}_{p-1}$, la clé privée composée des entiers $p \in \mathbb{P}$, $g \in \mathbb{Z}_p$ et $d \in \mathbb{Z}_{p-1}$, le message chiffré $c \in \mathbb{Z}$ et les informations publiques $n, k \in \mathbb{Z}$
2: Calculer $r = c - kd \mod p-1$
3: Calculer $u = g^r \mod p$
4: **pour** chaque $i \in \{1, ..., n\}$ **faire**
5: Calculer $p_i = g^{b_i - d \mod p-1} \mod p$
6: $m_i := \begin{cases} 1 & \text{si } p_i \text{ divise } u \\ 0 & \text{sinon} \end{cases}$
7: **fin pour**
8: **Retourner** $m = m_1 m_2 ... m_n$

Le déchiffrement fonctionne car

$$\begin{aligned} u &= g^r \\ &= g^{c-kd} \\ &= g^{\sum_{i=1}^n m_i b_i - kd} \\ &= g^{\sum_{i=1}^n m_i a_i + \sum_{i=1}^n m_i d - kd} \\ &= g^{\sum_{i=1}^n m_i a_i + d(\sum_{i=1}^n m_i - k)} \\ &= g^{\sum_{i=1}^n m_i a_i + d(k-k)} \\ &= g^{\sum_{i=1}^n m_i a_i} \\ &= \prod_{i=1}^n g^{m_i a_i} \\ &= \prod_{i=1}^n (g^{a_i})^{m_i} \\ &= \prod_{i=1}^n p_i^{m_i} \end{aligned}$$

La valeur de u est donc égale au produit des p_i tels que $m_i = 1$. Comme tous les p_i sont premiers entre eux, il est certain que si p_i divise u, m_i est égal à 1.

Sans connaître p, g et d, retrouver m semble toutefois très difficile. Les $b_1, ..., b_n$, étant d'apparence aléatoire, offrent très peu d'information utile. L'attaquant doit donc utiliser le message chiffré $\sum_{i=1}^n m_i b_i$, avec les éléments $b_1, ..., b_n$, ce qui est une instance de Somme de sous-ensembles.

Les précédents cryptosystèmes basés sur ce problème admettent une cryptanalyse à cause de leur faible densité, la densité étant une métrique égalant

$$\frac{n}{\log\left(\max_i\left(b_i\right)\right)}$$

Intuitivement, cela signifie que ces cryptosystèmes incorporaient généralement quelques grands éléments plutôt que beaucoup de petits. Il a été montré que presque toute instance de densité inférieure à $0,9408$ peut être résolue à l'aide d'un algorithme trouvant le plus court vecteur dans un réseau euclidien (voir section 2.3.2) [55], ce qui a contribué à mettre en péril les anciens cryptosystèmes. En effet, bien que l'algorithme LLL [43] trouve une solution *approximative* à ce problème, il fonctionne bien en pratique pour les densités inférieures à 1 [37].

Puisqu'il force la présence d'exactement k b_i, le protocole OTU permet, avec un choix judicieux des paramètres, de garantir que la densité est d'au moins 1, ce qui le rend pour l'instant invulnérable à l'attaque basée sur les réseaux euclidiens [54]. De plus, le fait que nous soyons dans un cadre quantique ne change rien de ce côté, car nous ne connaissons pas d'algorithme quantique substantiellement meilleur pour résoudre ce genre de problème. C'est d'ailleurs pour cette même raison que la cryptographie basée sur les réseaux euclidiens est un bon candidat post-quantique.

Contrairement à l'utilisation de *canaux* quantiques, le *calcul* quantique ne semble pas apporter grand chose d'intéressant au design de protocoles cryptographiques. Ici, l'utilisation de l'algorithme de Shor, bien qu'utile pour cacher la porte dérobée, n'amène rien de fondamentalement nouveau par rapport à la cryptographie post-quantique de l'horizon II.

3.3 Comparaison des protocoles contemporains, post-quantiques et quantiques

Nous avons au total présenté dix cryptosystèmes, divisés en trois catégories, selon qu'ils :

- **sont utilisés actuellement, mais vulnérables à des attaques quantiques** (RSA, El Gamal sur $\mathbb{Z}_p^*$, El Gamal sur courbes elliptiques) ;
- **sont utilisables avec des ordinateurs classiques et en apparence résistants aux attaques quantiques** (Codes, réseaux euclidiens, polynômes multivariés, isogénies sur courbes elliptiques supersingulières, groupes non abéliens) ; ou
- **nécessitent des appareils quantiques** (BB84, OTU).

Imaginons-nous en l'an 2043. Les appareils quantiques sont désormais très répandus et l'on suppose qu'aucun de ces protocoles n'est trop difficile à utiliser –non seulement nous considérons que l'on peut exécuter des algorithmes quantiques, mais l'on suppose aussi que la différence d'utilisabilité entre chaque protocole est négligeable, i.e. les différences entre tailles de clés et temps d'exécution des algorithmes de génération de clé, de chiffrement et de déchiffrement ne sont pas des arguments valables dans notre analyse. En effet, il est vraisemblable que 25 ans plus tard la puissance des ordinateurs se soit améliorée au point où tout algorithme polynomial est exécuté très efficacement ; cependant il est improbable que l'écart théorique séparant les algorithmes polynomiaux des algorithmes exponentiels ait disparu.

Nous nous concentrons donc sur la *sécurité* de ces protocoles et les analysons selon trois critères, à savoir :

- **Complexité de l'hypothèse sous-jacente** : Dans le cas d'une hypothèse calculatoire, dans quelle classe de complexité se trouve le problème algorithmique sous-jacent ? Est-il dans BQP, NP-complet, ou quelque part entre les deux ? Dans le cas d'un autre type d'hypothèse, à quel point peut-on se fier à cette supposition ?
- **Qualité de la réduction** : Dans quel sens est réalisée la réduction liant le cryptosystème et le problème sous-jacent ? La difficulté du problème nous donne-t-elle une borne supérieure seulement, ou une borne inférieure ? S'agit-il d'une preuve formelle, ou d'une simple ressemblance entre les problèmes ? Quel est l'impact de la porte dérobée sur la réelle difficulté du cryptosystème ?
- **Résistance à la cryptanalyse** : Le domaine des mathématiques dans lequel le cryptosystème prend place a-t-il été suffisamment étudié par les cryptanalystes ? Le protocole est-il conceptuellement simple et accessible aux concepteurs d'algorithmes quantiques ? Son invulnérabilité apparente vient-elle vraiment de la complexité inhérente du problème, ou de *notre* incapacité à définir un algorithme pour ce problème ? Les meilleures attaques jusqu'à maintenant semblent-elles optimales, ou laissent-elles présager de pires attaques ?

Chacun des protocoles sera discuté individuellement, puis un classement final viendra résumer nos recommandations en vue de cerner les protocoles que nous devrions prioriser dans le futur pour assurer la sécurité à long terme de nos communications.

3.3.1 Analyse individuelle

Cryptosystème RSA

Complexité : 8$^{\text{e}}$ sur 10 (ex aequo)
Le protocole RSA est associé au problème Factorisation, dont nous avons démontré l'appartenance à la classe de complexité BQP. Par définition, cela signifie qu'on peut résoudre le problème à l'aide d'un ordinateur quantique ; l'hypothèse selon laquelle Factorisation est difficile est donc très faible.

Réduction : 7$^{\text{e}}$ sur 10 (ex aequo)
Si RSA se réduit à Factorisation, l'inverse ne semble pas être vrai. La difficulté de Factorisation donne donc seulement une borne supérieure à la complexité de briser le protocole. De plus, la porte dérobée fait en sorte qu'il suffit en fait de résoudre le Problème RSA, possiblement plus facile encore que factorisation.

Cryptanalyse : 8$^{\text{e}}$ sur 10
Bien qu'il s'agisse d'un problème très célèbre et malgré tout sans solution classique, nous ne pouvons nier, dans un contexte post-quantique, que RSA est complètement vulnérable à la cryptanalyse, à cause de l'algorithme de Shor.

Cryptosystème El Gamal sur $\mathbb{Z}_p^*$

Complexité : 8$^{\text{e}}$ sur 10 (ex aequo)
El Gamal sur $\mathbb{Z}_p^*$ se réduit au Logarithme discret, un autre problème de BQP.

Réduction : 7$^{\text{e}}$ sur 10 (ex aequo)
À l'instar de RSA, la réduction donne une borne supérieure seulement à la complexité de briser El Gamal et la porte dérobée rend cette borne possiblement moins élevée puisqu'il suffit de résoudre le Problème de Diffie-Hellman, qui n'est certainement pas plus difficile que le Logarithme discret.

Cryptanalyse : 9$^{\text{e}}$ sur 10
Le Logarithme discret est un problème un peu moins célèbre que la factorisation, donc il semble légèrement plus probable qu'un algorithme efficace classique existe mais échappe à la communauté des cryptanalystes que dans le cas de la Factorisation. Du point de vue quantique, le protocole est toutefois assurément vulnérable.

Cryptosystème El Gamal sur courbes elliptiques

Complexité : 8$^{\text{e}}$ sur 10 (ex aequo)
El Gamal sur courbes elliptiques se réduit lui aussi au Logarithme discret de BQP.

Réduction : 7$^{\text{e}}$ sur 10 (ex aequo)
La réduction présente les mêmes problèmes que El Gamal sur $\mathbb{Z}_p^*$.

Cryptanalyse : 10$^{\text{e}}$ sur 10
En pratique, la situation de El Gamal sur courbes elliptiques est pire encore que pour les deux cryptosystèmes précédents. En effet, l'avantage de ce cryptosystème dans un monde classique est que des clés moins longues sont nécessaires pour atteindre un même niveau de sécurité. Mais en présence d'un ordinateur quantique, la difficulté devient directement proportionnelle au nombre de bits des clés utilisées. Ainsi, en supposant que les ordinateurs quantiques stables auront de plus en plus de qubits, ce protocole sera le premier à tomber [58].

Cryptographie basée sur les codes

Complexité : 2$^{\text{e}}$ sur 10 (ex aequo)
Le Décodage général est NP-complet ce qui, suivant notre hypothèse que chacun des problèmes calculatoires utilisés dans un cryptosystème doit être dans NP, rend ce problème aussi difficile qu'on pourrait l'espérer.

Réduction : 4$^{\text{e}}$ sur 10 (ex aequo)
Si un cryptosystème comme celui de McEliece se réduit au décodage général, il ne semble cependant pas *aussi* difficile : puisqu'il est obtenu à partir d'une permutation d'une instance facile, nous ne pouvons pas dire que briser le cryptosystème est un problème NP-complet.

Cryptanalyse : 2$^{\text{e}}$ sur 10 (ex aequo)
La théorie des codes est un domaine bien étudié, le protocole de McEliece a fait son apparition il y a déjà 40 ans, et la candidature de ce protocole en cryptographie post-quantique l'a ramené d'actualité ; nous pouvons donc affirmer que le problème a été plutôt bien étudié. Pourtant, aucune faille majeure n'a été détectée encore, ce qui laisse présager qu'il s'agit bien d'un protocole sécuritaire.

Cryptographie basée sur les réseaux euclidiens

Complexité : 6$^{\text{e}}$ sur 10 (ex aequo)
Le problème Plus court vecteur (approximé) est NP-complet [48]. Comme nous savons qu'il se réduit (quantiquement) à Apprentissage avec erreurs, ce dernier semble lui aussi être parmi les problèmes les plus difficiles de NP, sans être assurément NP-complet.

Réduction : 2e sur 10
Pour la génération de la clé, c'est le problème calculatoire qui se réduit à briser le cryptosystème et non l'inverse. Nous avons donc une borne inférieure à la complexité inhérente au cryptosystème.

Cryptanalyse : 2e sur 10 (ex aequo)
Tout comme pour la cryptographie basée sur les codes, les réseaux euclidiens sont très étudiés et ne présentent malgré tout pas de vulnérabilités pour l'instant.

Cryptographie basée sur les polynômes multivariés

Complexité : 2e sur 10 (ex aequo)
La résolution d'un système d'équations quadratiques est un problème NP-complet.

Réduction : 4e sur 10 (ex aequo)
Comme pour la cryptographie basée sur les codes, le problème NP-complet ne donne qu'une borne supérieure, et la porte dérobée empêche de garantir la NP-complétude du protocole lui-même.

Cryptanalyse : 4e sur 10
Le problème est très étudié en général, d'autant plus depuis qu'il s'agit d'un candidat post-quantique sérieux. Ceci dit, beaucoup de protocoles basés sur les polynômes multivariés qui semblaient sécuritaires par le passé ont finalement été attaqués avec succès, laissant croire que ceux qui survivent encore sont tout de même à risque.

Cryptographie basée sur les isogénies sur courbes elliptiques supersingulières

Complexité : 7e sur 10
Le Problème de griffe se résout en $\mathcal{O}(\sqrt[6]{N})$ sur un ordinateur quantique, ce qui est plus efficace que l'algorithme de Grover qui s'exécute en $\mathcal{O}(\sqrt{N})$. Ce dernier algorithme est pourtant actuellement le meilleur pour les problèmes NP-complet, alors il est invraisemblable que le Problème de griffe s'avère aussi difficile. Ce problème semblerait donc être quelque part entre BQP et NP-complet.

Réduction : 4e sur 10 (ex aequo)
La réduction ne donne qu'une borne supérieure, mais la borne semble assez serrée.

Cryptanalyse : 7e sur 10
Le domaine des isogénies sur courbes elliptiques supersingulières est complexe et difficile d'approche. Il est par conséquent peu étudié par les concepteurs d'algorithmes quantiques ; l'absence d'algorithme connu vient donc probablement davantage de notre manque de tentatives

de cryptanalyse que de la réelle difficulté du problème.

Cryptographie basée sur les groupes non abéliens

Complexité : 2$^{\text{e}}$ sur 10 (ex aequo)
La Recherche de conjugué dans un groupe non abélien est, dans le cas général, NP-complet [16].

Réduction : 7$^{\text{e}}$ sur 10 (ex aequo)
La réduction ne donne encore qu'une borne supérieure et ne serait pas si serrée ; le protocole présenté se réduit par exemple au Problème de décomposition, potentiellement plus facile. Bien que le problème général soit NP-complet, le choix d'un groupe en particulier peut rendre le problème plus simple.

Cryptanalyse : 5$^{\text{e}}$ sur 10
Les groupes non abéliens sont variés et pas toujours bien compris. L'existence d'attaques efficaces visant tous les groupes à la fois n'est pas impossible, et il est encore plus vraisemblable que des attaques existent pour chaque groupe spécifique utilisé.

Cryptosystème BB84

Complexité : 1$^{\text{er}}$ sur 10
L'hypothèse sous-jacente à ce protocole n'est pas calculatoire mais physique, ce qui le distingue profondément des autres. Comme on assume que l'espionnage peut être détecté, on n'a même pas à supposer l'existence d'un problème dans NP puisqu'on sait que l'espion manquera d'information et ne pourra rien résoudre même avec un temps infini. Les hypothèses sont basées sur la mécanique quantique, une théorie beaucoup mieux comprise que la théorie de la complexité, et sont ainsi beaucoup plus solides.

Réduction : 1$^{\text{er}}$ sur 10
Il ne s'agit pas ici d'une réduction, mais d'une preuve formelle que le protocole respecte le secret parfait. Supposant que nos hypothèses physiques sont bonnes, il est garanti que le cryptosystème ne peut être brisé.

Cryptanalyse : 1$^{\text{er}}$ sur 10
Le secret parfait empêche tout attaquant passif de concrétiser une attaque.

Position	Protocole	Commentaire
1	Cryptosystème BB84	**Sécurité inconditionnelle**
2	Cryptographie basée sur les réseaux euclidiens	**Sécurité calculatoire prometteuse**
3	Cryptographie basée sur les codes	
4	Cryptographie basée sur les polynômes multivariés	
5	Cryptosystème OTU	**Sécurité calculatoire douteuse**
6	Cryptographie basée sur les groupes non abéliens	
7	Cryptographie basée sur les isogénies sur courbes elliptiques supersingulières	
8	Cryptosystème RSA	**Non sécuritaire**
9	Cryptosystème El Gamal sur $\mathbb{Z}_p^*$	
10	Cryptosystème El Gamal sur courbes elliptiques	

Figure 3.3 – Classement final des 10 protocoles présentés

Cryptosystème OTU

Complexité : 2e sur 10 (ex aequo)
Le protocole OTU est basé sur Somme de sous-ensembles, un problème NP-complet typique.

Réduction : 3e sur 10
Encore une fois, la réduction ne donne qu'une borne supérieure, toutefois la porte dérobée semble mieux dissimulée qu'à l'habitude puisqu'elle tire son apparence aléatoire d'une fonction non linéaire.

Cryptanalyse : 6e sur 10
Somme de sous-ensembles, étant un problème NP-complet canonique, est très étudié ; cependant, l'histoire a montré que les protocoles basés sur ce problème sont généralement cryptanalysés avec succès. Étant une proposition plutôt récente, il ne serait pas surprenant que sa vulnérabilité n'ait simplement pas encore été découverte.

3.3.2 Classement final et recommandations

À la lumière des analyses individuelles précédentes, nous pouvons établir le classement final de la figure 3.3.

Nous nous permettons maintenant quelques recommandations :

— À l'intention de l'étudiant souhaitant débuter ses recherches en cryptographie post-quantique : notre classement rend évident que les protocoles post-quantiques les plus populaires, à savoir ceux basés sur les réseaux euclidiens, les codes et les polynômes

multivariés, sont populaires avec raison. Parmi ceux-ci, nous croyons que les réseaux euclidiens sont les plus prometteurs étant donné la borne inférieure à sa complexité calculatoire.

— À l'intention de l'analyste en sécurité du futur : si, en 2043, presque tous les canaux de communication permettent la transmission de qubits, alors nous conseillons fortement d'utiliser des protocoles dans le style de BB84, puisqu'il est *infiniment* plus sécuritaire que les autres. Sinon, il ne vaut pas la peine d'utiliser l'ordinateur quantique pour exécuter un protocole, puisque des protocoles post-quantiques classiques semblent plus fiables.

— À l'intention de l'analyste en sécurité contemporain : la menace quantique est à nos portes, un ordinateur de ce type peut venir au monde du jour au lendemain. Alors de grâce, cessez de vous servir de protocoles que l'on *sait* être vulnérables.

Conclusion

La cryptographie sera énormément impactée par l'arrivée de l'ordinateur quantique. Cependant, nous connaissons très bien la teneur de l'impact, et nous pouvons faire quelque chose pour prévenir un avenir chaotique. En dressant un portrait en trois horizons du futur de cette science, nous espérons avoir fait la lumière sur les véritables conséquences de l'informatique quantique, un des sujets scientifiques les plus mal interprétés.

En effet, contrairement à ce que l'on entend parfois, l'informatique quantique ne révolutionne que très peu l'algorithmique jusqu'à maintenant. Elle apporte un gain de calcul significatif seulement pour une poignée de problèmes, qui s'avèrent être ceux-là même dont la complexité est cruciale pour nos protocoles cryptographiques. Cette menace nous oblige à recourir à des problèmes plus difficiles, mais nous ne pouvons avoir qu'une confiance limitée en la difficulté de ces problèmes : la théorie de la complexité étant encore pleine de questions ouvertes, nous ne pouvons jamais véritablement garantir qu'ils sont impossibles à résoudre.

Néanmoins, il nous faut faire face à la menace avec le peu d'outils que l'on détient. Bien que cela ne soit pas une solution à court terme, ni probablement à long terme, si jamais les canaux quantiques deviennent très répandus alors nous pourrons nous sentir en sécurité grâce à un protocole dans le style de BB84. D'ici là, il faudra nous contenter d'un protocole post-quantique, et nous suggérons à la communauté scientifique de se concentrer tout particulièrement sur la cryptographie basée sur les réseaux euclidiens.

Si l'ordinateur quantique est si dangereux, devrait-on arrêter de poursuivre des recherches dans ce domaine ? Nous pensons qu'il est impossible d'arrêter l'humain pour ce genre de défi, surtout sachant que le potentiel de cette technologie dépasse sa capacité de cryptanalyse, et donc que la seule solution réaliste est de faire un grand effort commun dans la découverte de théorèmes en complexité de calcul et dans la conception de protocoles post-quantiques pour assurer d'avoir toujours un pas d'avance sur la cryptanalyse.

Québec, Octobre 2018

Bibliographie

[1] A Preview of Bristlecone, Google's New Quantum Processor. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html. Accédé le 18 juillet 2018.

[2] Complexity Zoo. https://complexityzoo.uwaterloo.ca/Complexity_Zoo. Accédé le 14 juin 2018.

[3] Daniel et Chi-Domínguez Jesús-Javier et Menezes Alfred et Rodríguez-Henríquez Francisco Adj, Gora et Cervantes-Vázquez. On the Cost of Computing Isogenies Between Supersingular Elliptic Curves. Technical report, Cryptology ePrint Archive, Report 2018/313, 2018. https://eprint.iacr.org/2018/313, 2018.

[4] Neeraj et Saxena Nitin Agrawal, Manindra et Kayal. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.

[5] Oded et Goldwasser Shafi et Moshkovitz-Dana Akavia, Adi et Goldreich. On Basing One-Way Functions on NP-Hardness. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 701–710. ACM, 2006.

[6] Boaz Arora, Sanjeev et Barak. *Computational Complexity : A Modern Approach*. Cambridge University Press, 2009.

[7] Phillip Bellare, Mihir et Rogaway. Random Oracles are Practical : A Paradigm for Designing Efficient Protocols. *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[8] Gilles Bennett, Charles H. et Brassard. Quantum Cryptography : Public Key Distribution and Coin Tossing. *Theoretical Computer Science*, 560(P1) :7–11, 2014.

[9] Robert et Van Tilborg Henk Berlekamp, Elwyn et McEliece. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory*, 24(3) :384–386, 1978.

[10] Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[11] Umesh Bernstein, Ethan et Vazirani. Quantum Complexity Theory. *SIAM Journal on computing*, 26(5) :1411–1473, 1997.

[12] Eva Borbely. Grover Search Algorithm. *arXiv preprint arXiv :0705.4171*, 2007.

[13] Gábor et Meyer Tim et Riege-Tobias et Rothe Jörg Bruss, Dagmar et Erdélyi. Quantum Cryptography : A Survey. *ACM Computing Surveys (CSUR)*, 39(2) :6, 2007.

[14] Johannes Buchmann. *Introduction to Cryptography*. Springer Science & Business Media, 2013.

[15] F. Canteaut, A. et Chabaud. Improvements of the Attacks on Cryptosystems Based on Error-correcting Codes. *LIENS*, 95 :21, 1995.

[16] Delaram Cavallo, Bren et Kahrobaei. A Family of Polycyclic Groups over which the Uniform Conjugacy Problem is NP-complete. *International Journal of Algebra and Computation*, 24(04) :515–530, 2014.

[17] David et Soukharev Vladimir Childs, Andrew et Jao. Constructing Elliptic Curve Isogenies in Quantum Subexponential Time. *Journal of Mathematical Cryptology*, 8(1) :1–29, 2014.

[18] Ivan Damgård. QIP Note : On the Quantum Fourier Transform and Applications. *Publié sur* `http://www.brics.dk/~ivan/fourier.pdf`, 2004.

[19] Martin Diffie, Whitfield et Hellman. New Directions in Cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976.

[20] Albrecht Ding, Jintai et Petzoldt. Current State of Multivariate Cryptography. *IEEE Security & Privacy*, 15(4) :28–36, 2017.

[21] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985.

[22] Stefan Friedl. An Elementary Proof of the Group Law for Elliptic Curves. *Groups Complexity Cryptology*, 9(2) :117–123, 2017.

[23] Frederik Galbraith, Steven D et Vercauteren. Computational Problems in Supersingular Elliptic Curve Isogenies. *Quantum Information Processing*, 17(10) :265, 2018.

[24] David Garber. Braid Group Cryptography. In *Braids : Introductory lectures on braids, configurations and their applications*, pages 329–403. World Scientific, 2010.

[25] David S Garey, Michael R et Johnson. *Computers and Intractability*. W.H. Freeman, New York, 2002.

[26] John Gill. Computational Complexity of Probabilistic Turing machines. *SIAM Journal on Computing*, 6(4) :675–695, 1977.

[27] Oded Goldreich. Computational Complexity : A Conceptual Perspective. *ACM Sigact News*, 39(3) :35–39, 2008.

[28] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[29] Papro Gruber, Hermann et Soft Gbr. Information-theoretic Cryptography. *Publié sur* http://wwwmayr.in.tum.de/konferenzen/Jass05/courses/1/papers/gruber_paper.pdf, 2005.

[30] Chunsheng Gu. Cryptanalysis of Simple Matrix Scheme for Encryption. *IACR Cryptology ePrint Archive*, 2016 :1075, 2016.

[31] Alfred J et Vanstone Scott Hankerson, Darrel et Menezes. *Guide to Elliptic Curve Cryptography*. Springer Science & Business Media, 2006.

[32] Richard E Hartmanis, Juris et Stearns. On the Computational Complexity of Algorithms. *Transactions of the American Mathematical Society*, 117 :285–306, 1965.

[33] Levin Impagliazzo, Russell et LA. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Foundations of Computer Science, 1990. Proceedings, 31st Annual Symposium on*, pages 812–821. IEEE, 1990.

[34] Luca Jao, David et De Feo. Towards Quantum-resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[35] Richard Jozsa. Quantum algorithms and the Fourier transform. *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, 454(1969) :323–337, 1998.

[36] Bilal Kahrobaei, Delaram et Khan. A Non-commutative Generalization of ElGamal Key Exchange Using Polycyclic Groups. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, pages 1–5. IEEE, 2006.

[37] Ian Kate, Aniket et Goldberg. Generalizing Cryptosystems Based on the Subset Sum Problem. *International Journal of Information Security*, 10(3) :189–199, 2011.

[38] Subhash Khot. Hardness of Approximating the Shortest Vector Problem in Lattices. *Journal of the ACM (JACM)*, 52(5) :789–808, 2005.

[39] Marcus Kindberg. A Usability Study of Post-Quantum Algorithms. *Lund University Publications*, 2017.

[40] Claus et Lenstra Arjen K et Priplata Christine et Stahlke Colin Kleinjung, Thorsten et Diem. Computation of a 768-bit Prime Field Discrete Logarithm. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 185–201. 2017.

[41] Kazumaro et Franke Jens et Lenstra Arjen K et Thomé Emmanuel et Bos Joppe W et Gaudry Pierrick et Kruppa Alexander et Montgomery Peter L et Osvik Dag Arne et others Kleinjung, Thorsten et Aoki. Factorization of a 768-bit RSA Modulus. *Annual Cryptology Conference*, pages 333–350, 2010.

[42] Richard E Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM (JACM)*, 22(1) :155–171, 1975.

[43] Hendrik Willem et Lovász László Lenstra, Arjen Klaas et Lenstra. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4) :515–534, 1982.

[44] Chris Lomont. The Hidden Subgroup Problem - Review and Open Problems. *arXiv preprint quant-ph/0411037*, 2004.

[45] Karl Mahlburg. An Overview of Braid Group Cryptography. *Publié sur* `http://www.math.wisc.edu/~boston/mahlburg.pdf`, 2004.

[46] Robert J. McEliece. A Public-key Cryptosystem Based on Algebraic Coding Theory. *Technical Report DSN progress report 42-44, Jet Propulsion Laboratory, Pasadena*, 4244 :114–116, 1978.

[47] Martin Merkle, Ralph et Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. *IEEE transactions on Information Theory*, 24(5) :525–530, 1978.

[48] Daniele Micciancio. The Shortest Vector Problem is NP-hard to Approximate to Within some Constant. *SIAM Journal on Computing*, 30(6) :2008–2035, March 2001. *Version préliminaire dans Foundations of Computer Science 1998.*

[49] Gary L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of computer et system sciences*, 13(3) :300–317, 1976.

[50] Artur Mosca, Michele et Ekert. The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer. *Quantum Computing and Quantum Communications*, pages 174–188, 1999.

[51] Michele Mosca. *Quantum Computer Algorithms.* PhD thesis, University of Oxford., 1999.

[52] Ashwin Nayak. Inverting a permutation is as hard as unordered search. *Theory of Computing*, 7(2) :19–25, 2011.

[53] Isaac Nielsen, Michael A et Chuang. *Quantum Computation and Quantum Information*. AAPT, 2002.

[54] Keisuke et Uchiyama Shigenori Okamoto, Tatsuaki et Tanaka. Quantum Public-key Cryptosystems. In *Annual International Cryptology Conference*, pages 147–165. Springer, 2000.

[55] Glenn Orton. A Multiple-iterated Trapdoor for Dense Compact Knapsacks. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 112–130. Springer, 1994.

[56] Albrecht Petzoldt. *Selecting and Reducing Key Sizes for Multivariate Cryptography*. PhD thesis, Université de technologie de Darmstadt, 2013.

[57] John et Ding Jintai Porras, Jaiberth et Baena. ZHFE, a New Multivariate Public Key Encryption Scheme. In *International workshop on post-quantum cryptography*, pages 229–245. Springer, 2014.

[58] Christof Proos, John et Zalka. Shor's Discrete Logarithm Quantum Algorithm for Elliptic Curves. *Quantum Information and Computation*, pages 317–344, 2003.

[59] Oded Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, 56(6) :34, 2009.

[60] Adi et Adleman Leonard Rivest, Ronald L et Shamir. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[61] René Schoof. Elliptic Curves over Finite Fields and the Computation of Square Roots mod p. *Mathematics of computation*, 44(170) :483–494, 1985.

[62] Adi Shamir. A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. In *Foundations of Computer Science, 1982*, pages 145–152. IEEE, 1982.

[63] Claude E Shannon. Communication Theory of Secrecy Systems. *Bell system technical journal*, 28(4) :656–715, 1949.

[64] Peter W. Shor. Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM review*, 41(2) :303–332, 1999.

[65] Alexander Shpilrain, Vladimir et Ushakov. The Conjugacy Search Problem in Public Key Cryptography : Unnecessary and Insufficient. *Applicable Algebra in Engineering, Communication and Computing*, 17(3-4) :285–289, 2006.

[66] Seiichiro Tani. Claw Finding Algorithms Using Quantum Walk. *Theoretical Computer Science*, 410(50) :5285–5297, 2009.

[67] Serge Vaudenay. Cryptanalysis of the Chor-Rivest Cryptosystem. In *Annual International Cryptology Conference*, pages 243–256. Springer, 1998.

[68] Andrew C. Yao. Theory and Application of Trapdoor Functions. In *Foundations of Computer Science (1982)*, pages 80–91.

[69] Hong Zhu. *Survey of Computational Assumptions Used in Cryptography Broken or Not by Shor's Algorithm.* PhD thesis, McGill University Libraries, 2001.