TP : Algorithme de Shor
ZZ3 F5 - Réseaux et Sécurité Informatique

Pascal Lafourcade

En arithmétique modulaire et en informatique quantique, l'algorithme de Shor est un algorithme quantique pour factoriser un entier naturel N en temps $O((logN)^3)$ et en espace $O(logN)$, nommé en l'honneur de Peter Shor. Peter Williston Shor, né le 14 août 1959, est un mathématicien américain. Il est connu pour son travail sur le calcul quantique et il est professeur au MIT.

Peter Williston Shor Algorithme de Shor Beaucoup de cryptosystèmes à clé publique, tels que le RSA, deviendraient vulnérables si l'algorithme de Shor était un jour implémenté dans un calculateur quantique pratique. Un message chiffré avec RSA peut être déchiffré par factorisation de sa clé publique N, qui est le produit de deux nombres premiers. En l'état actuel des connaissances, il n'existe pas d'algorithme classique capable de faire cela en temps $O((logN)^k)$ pour n'importe quel k, donc, les algorithmes classiques connus deviennent rapidement impraticables quand N augmente, à la différence de l'algorithme de Shor qui peut casser le RSA en temps polynomial. Il a été aussi étendu pour attaquer beaucoup d'autres cryptosystèmes à clé publique.

Comme tous les algorithmes pour calculateur quantique, l'algorithme de Shor est probabiliste : il donne la réponse correcte avec une haute probabilité et la probabilité d'échec peut être diminuée en répétant l'algorithme. L'algorithme de Shor fut utilisé en 2001 par un groupe d'IBM, qui factorisa 15 en 3 et 5, en utilisant un calculateur quantique de 7 qubits.

Durant cette séance, nous allons implémenter l'algorithme de Shor afin de retrouver ce résultat.

1. Dans un premier temps, vous allez mettre en place l'algorithme de Shor (cf. annexe 1) sans la fonction quantique. Au vu des tests que nous allons réaliser ($N = 15$), une implémentation classique est largement suffisante. Cette fonction de recherche de la période r se fera donc avec une transformée de Fourier discrète classique.

2. Bien entendu, nous utiliserons pas un ordinateur quantique mais une librairie simulant son utilisation avec la manipulation de qubits, de registre de qubit et de portes quantiques. Dans un second temps, nous allons remplacer la fonction de recherche de période par une version quantique. Pour cela, vous utiliserez la description de la transformée de Fourier Quantique dans l'annexe 2 et la librairie Quantum++

`https://github.com/softwareQinc/qpp`

Annexe 1

Introduction : Le problème que nous essayons de résoudre est le suivant : étant donné un nombre composé impair N, trouver un entier d, strictement compris entre 1 et N, qui divise N. 1 et N, qui divise N. Nous nous intéressons aux valeurs impaires de N car toute valeur paire de N a trivialement le nombre 2 comme facteur premier. Nous pouvons utiliser un algorithme de test de primalité pour nous assurer que N est bien composite.

Algorithme : L'algorithme de Shor se compose de deux parties :

- Une réduction, qui peut être faite sur un ordinateur classique, du problème de la problème de factorisation au problème de recherche d'ordre ;
- Un algorithme quantique pour résoudre le problème de recherche d'ordre.

Algorithm 1 Algorithme de Shor pour la factorisation de N

Entrée : Un entier N à factoriser.
Sortie : Les facteurs premiers de N ou un facteur trivial.
Choisir un entier aléatoire a tel que $1 < a < N$.
if $\gcd(a, N) > 1$ **then**
    Retourner $\gcd(a, N)$ comme facteur non trivial de N.
end if
Trouver le plus petit entier r tel que $a^r \equiv 1 \pmod{N}$ (c'est-à-dire l'ordre de a modulo N).
if r est impair **ou** $a^{r/2} \equiv -1 \pmod{N}$ **then**
    Retourner à l'étape 1 avec un nouvel entier a.
end if
Calculer $\gcd(a^{r/2} - 1, N)$ et $\gcd(a^{r/2} + 1, N)$.
if $\gcd(a^{r/2} - 1, N) > 1$ **et** $\gcd(a^{r/2} + 1, N) > 1$ **then**
    Retourner les deux facteurs non triviaux de N.
else
    Retourner à l'étape 1 avec un nouveau choix de a.
end if

Par example : $N = 15$, $a = 7$, $r = 4$, $gcd(721, 15) = gcd(491, 15)$, où $gcd(48, 15) = 3$, et $gcd(50, 15) = 5$.

Les circuits quantiques utilisés pour cet algorithme sont conçus sur mesure pour chaque choix de N et chaque choix de a aléatoire utilisé dans $f(x) = a^x \bmod N$). Etant donné N, trouvez $Q = 2^q$ tel que $N^2 \leq Q < 2N^2$, ce qui implique $Q/r > N$. Les registres de qubits d'entrée et de sortie doivent contenir des superpositions de valeurs de 0 à $Q - 1$, et ont donc q qubits chacun.

L'utilisation de ce qui pourrait sembler être deux fois plus de qubits que nécessaire garantit qu'il y a au moins N x différents qui produisent le même $f(x)$, même lorsque la période r s'approche de $N/2$.

Annexe 2