La crypto post quantique.... (bah ça ne fait pas envie).

Quelques éléments de réponse..

4 mars 2025

Suite à l'article sur la stabilité des qubits et le focus sur la puce Ocelot d'AWS mettant en oeuvre des catbits (https://www.linkedin.com/posts/jeromev_catqubit-qubit-symetrie-activity-7302739959776464896-R8jX?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAChnekBGRF-CabUoQQTA69kbB9dT2pmuMI) Franck POULLAIN a posé des questions intéressantes. La limitation de caractères de la réponse dans linkedin m'oblige donc à tenter d'y répondre ici :)

Questions :

- Comment l'information est-elle stockée dans ces ordinateurs ?
- Quelle est la base de calcul : binaire ou autre ?
- Quelle architecture mémoire processeur est utilisée par ces ordinateurs quantiques ?
- Si je veux décrypter une clé de 2048 bits, faut-il 2048 qubits logiques ?

Bonsoir Franck, je vous remercie d'avoir posé cette question, elle illustre bien le recours à des schémas architecturaux qui n'ont pas cours en informatique quantique. Je vous recommande si vous ne l'avez pas lu, le premier article de la série qui pose les bases (https://www.linkedin.com/posts/jeromev_quantum-computing-etatsabrquantiques-activity-7291875882078470145-

f4T9?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAChnekBGRF-CabUoQQTA69kbB9dT2pmuMI).

STOCKAGE DE L'INFORMATION

Stockage très temporaire dans les qubits eux-mêmes

L'information est codée et stockée dans les qubits eux-mêmes, sous "forme" d'états quantique, et ce de façon très temporaire (cf les durées de cohérence des qubits de quelques µs à quelques minutes pour les plus stables).

Les qubits sont représentés mathématiquement par des vecteurs d'état dans un espace de Hilbert à deux dimensions :

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

où

- $|0\rangle$ et $|1\rangle$ sont les états de base (comme les 0 et 1 classiques) et aussi les vecteurs de **la base dite computationnelle** de l'espace de Hilbert associé.
- α et β sont des coefficients complexes qui respectent la contrainte de normalisation : $|\alpha|^2+|\beta|^2=1$

Propriétés fondamentales du stockage d'information quantique

L'ordinateur quantique stocke l'information en exploitant trois propriétés clés :

a) Superposition des états : Un qubit existe simultanément dans plusieurs combinaisons linéaires de la base computationnelle. Tant qu'il n'est pas mesuré, il reste dans cet état superposé.

b) Intrication (entanglement) : Lorsque plusieurs qubits sont intriqués, leur état devient corrélé quelle que soit la distance qui les sépare. Cela signifie que si l'on mesure le premier qubit et qu'on trouve 0, alors l'autre sera automatiquement 0 aussi, et pareil pour 1.

c) Interférence quantique : L'ordinateur quantique peut manipuler les probabilités des états superposés en appliquant des portes logiques quantiques pour modifier les amplitudes de α et β, permettant ainsi d'optimiser certains calculs.

Lecture et effacement de l'information quantique :

L'information est lue par mesure quantique, qui projette un qubit dans un état classique (non quantique) déterminé avec une certaine probabilité. **Ce processus d'effondrement est irréversible, ce qui signifie que la mesure détruit la superposition**.

Tant que le qubit n'est pas lu/mesuré, il est dans un état de superposition des états.

Dès qu'on le mesure, il y a effondrement de la superposition, et un seul état subsiste, et on en revient au binaire 0 ou 1.

A partir de là, on peut recourir à l'architecture "classique" des systèmes informatiques pour stocker l'information car on n'est plus dans le "champ" quantique.

Architecture mémoire des ordinateurs quantiques

Contrairement aux ordinateurs classiques qui utilisent une mémoire structurée avec des registres, cache, RAM et stockage, les ordinateurs quantiques n'ont pas de mémoire au sens traditionnel. L'architecture mémoire repose principalement sur la manipulation des qubits et l'encodage d'informations quantiques via des états quantiques.

L'information est volatile, car un qubit ne peut pas être cloné (théorème de non-clonage) et ne peut pas être dupliqué en mémoire comme dans un ordinateur classique (la duplication impose la lecture, donc la perte de la superposition => un flux d'information quantique ne peut être "espionné" sans que cela ne se sache, car la moindre tentative de lecture va effondrer la superposition des états)... au moins pas besoin de "dedup" :)

On peut utiliser des qubits ancillaires dont la technologie pourra être choisie en fonction de la durée de cohérence qu'ils offrent (de quelques µs à quelques minutes). Pour le moment (excepté la puce Ocelot justement), il est plutôt rare d'avoir des architectures de qubits hybrides (mettant en oeuvre différentes technologie de support de qubits).

Encore une fois, les calculs réalisés par un ordinateur quantique, après lecture des qubits "de sortie" peuvent être stockés de manière classique car ils ne sont plus dans le "champ" quantique du fait de l'effondrement de la superposition liée à la mesure/lecture du qubit.

Les architectures font et feront donc coexister des systèmes "classiques", éventuellement HPC pour soutenir des calculs de "stabilisation" de qubits, avec des composants "quantiques".

La base de calcul

C'est la base computationnelle des états $|0\rangle$ et $|1\rangle$.

La grande différence est la superposition des états :

- Avec 2 bits, je peux encoder 00, 01,10,11 de **façon séquentielle**, je dois donc les évaluer un par un et éventuellement les intégrer dans un calcul algorithmique, mais toujours un par un.
- Avec 2 qubits je peux encoder $|00\rangle$, $|01\rangle$, $|10\rangle$ et $|11\rangle$ **SIMULTANEMENT** et je peux réaliser des calculs quantiques simultanément sur ces différents états du fait de

la superposition des états... MAIS dès que je veux mesurer/lire les qubits, ils s'effondreront alors dans l'un ou l'autre de ces 4 états.

Décryptage d'une clef RSA de 2048 bits

Le nombre de qubits requis dépend de **l'algorithme utilisé**, de **l'architecture du processeur quantique**, et des techniques de **correction d'erreurs** nécessaires pour compenser la décohérence.

Les bases de la sécurité RSA

Le chiffrement RSA repose sur la difficulté de factoriser un grand entier naturel N, qui est le produit de deux très grands nombres premiers p et q. N est le module RSA. Sa taille en bits définit la **sécurité du chiffrement**. Par exemple :

- RSA-1024 : N a **1024 bits** (~300 chiffres décimaux).
- RSA-2048 : N a **2048 bits** (~617 chiffres décimaux).

La sécurité de RSA repose sur la **difficulté de factorisation** d'un très grand entier N pour retrouver p et q et permettre le déchiffrement. Pour cela on dispose déjà d'algorithmes non quantique :

1. **Algorithme de fraction continue** : efficace pour les petites clés.
2. **Algorithme de Fermat** : exploite la proximité des facteurs premiers.
3. **Crible quadratique (QS, Quadratic Sieve)** : factorise les nombres jusqu'à ~100 chiffres.
4. **Crible général des corps de nombres (GNFS, General Number Field Sieve)** :**Le plus rapide actuellement** pour factoriser des nombres à plus de **200 chiffres** avec une complexité croissante avec la taille de N (complexité sous-exponentielle) incompatible avec les très grandes clefs : on rappelle ici RSA-2048 : ~617 chiffres.

A ce stade, on peut évaluer que :

- **RSA-512 (155 chiffres) a été factorisé en 1999**.
- **RSA-1024 (309 chiffres) est encore sécurisé** mais considéré vulnérable à long terme.
- **RSA-2048 est encore sûr aujourd'hui**, mais menacé par les avancées en **quantique**.

L'algorithme quantique de Shor

L'algorithme de **Shor** permet de factoriser ce nombre en un temps polynomial sur un **ordinateur quantique** en utilisant la **transformée de Fourier quantique** (QFT) pour

exploiter des propriétés arithmétiques des nombres. Il permet donc de "casser" la sécurité RSA dès lors qu'on dispose d'assez de qubits.

L'algorithme de Shor factorise un nombre composite N en trouvant un diviseur non trivial sans brute force. Il repose sur la théorie des nombres et la transformée de Fourier quantique (QFT).

Il suit 4 grandes étapes :

1. Réduction du problème à la recherche d'un ordre.
2. Utilisation d'un algorithme quantique pour trouver la période r.
3. Déduction des facteurs de N en utilisant le PGCD (plus grand commun diviseur... bienvenue en primaire :).
4. Vérification des facteurs trouvés.

Étape 1 : Choisir un entier aléatoire a

On choisit un nombre **aléatoire** a tel que : $1<a<N$

et qui est **premier avec N,** c'est-à-dire que le plus grand commun diviseur de a et N est 1 : $\gcd(a,N)=1$

Si par chance **$\gcd(a,N)\neq 1$** alors **on a déjà trouvé un facteur** de N et on a terminé (100% des gagnants ont tenté leur chance :)

Sinon, on passe à l'étape suivante.

Étape 2 : Trouver la période r

On définit la fonction exponentielle modulaire suivante : $f(x)=a^x \bmod N$

Cette fonction est **périodique**, ce qui signifie qu'il existe un plus petit entier r tel que :

$a^r\equiv 1 \bmod N$

L'objectif est donc de trouver r, qui est la période de la fonction.

L'algorithme de Shor utilise un ordinateur quantique pour trouver cette période en utilisant la transformée de Fourier quantique (QFT).

Étape 3 : Trouver les facteurs de N

Une fois r trouvé, on utilise la relation :

$a^r\equiv 1 \bmod N$

Ce qui peut être réécrit comme :

$(a^{(r/2)}-1)(a^{(r/2)}+1)\equiv 0 \bmod N$

Si r est **pair**, alors :

gcd(a^(r/2)−1,N) ou gcd(a^(r/2)+1,N) donne un facteur non trivial de N.

Si r est impair, l'algorithme recommence avec une autre valeur de a.

Partie quantique : Recherche de la période avec une transformée de Fourier quantique (QFT)

L'étape clé de l'algorithme est la recherche de la période r, qui est effectuée grâce à l'ordinateur quantique **en utilisant la superposition des états !**

1. **Superposition quantique** : On encode toutes les valeurs possibles de x en superposition dans un registre quantique.
2. **Calcul parallèle** : L'ordinateur quantique évalue simultanément f(x)=a^x mod N pour toutes les valeurs de x en une "seule passe" (superposition quantique).
3. **Transformée de Fourier quantique (QFT)** : On applique une **transformée de Fourier quantique** pour détecter les **fréquences cachées**, ce qui révèle la période r.
4. **Mesure** : Une fois la période r trouvée (étape 2 quantique), on passe aux étapes classiques sur du HPC par exemple.

Exemple avec une clef RSA de module 21.

Factorisons donc N=21 (et admettons que comme n'importe quel collégien moyen on n'a jamais appris nos tables de multiplication... flem!).

1. **Choisir a** : Prenons a=2, car a doit être premier avec 21, et on vérifie bien gcp(2,21)=1 (et si on prend 3, on plie le game).
2. **Trouver r** : La séquence 2^x mod 21 et rechercher **le plus petit r tel que** 2^r≡1 mod 21
3. **Calculons les puissances de 2 (=a)**

2^1 =2 et 2^x mod 21 donne 2 (car 2/21=0*21+2)

2^2 =4 et 2^x mod 21 donne 4 (car 4/21=0*21+4)

2^3=8 et 2^x mod 21 donne 8 (car 8/21=0*21+8)

2^4=16 et 2^x mod 21 donne 16 (car 16/21=0*21+16)

2^5=32 et 2^x mod 21 donne 11 (car 32/21=1*21+11)

2^6=64 et 2^x mod 21 donne 1 (car 64/21=3*21+1)

Le plus petit r tel que 2^r≡1 mod 21 est donc r=6

L’algorithme de Shor nous dit que **si r est pair**, alors nous pouvons utiliser :

gcd(2^(r/2)−1,21) et gcd(2^(r/2)+1,21)

avec r=6, r/2=3 et donc :

gcd(7,21)=7 et gcd(9,21)=3

Nous avons trouvé **les facteurs de 21** :

21=3×7 (bon ok on aurait mieux fait d'apprendre nos tables de multiplication plutôt que de vider 3 piscines olympiques et consommé 1 GWh.... forcément avec une clef 2048 bits, on doit factoriser un entier composé d'environ 617 chiffres).

Combien de qubits sont nécessaires ?

L'algorithme de **Shor** nécessite :

1. **Qubits logiques** pour stocker les **entiers à factoriser** et effectuer les opérations.
2. **Registres auxiliaires** pour l'algèbre modulaire et les calculs quantiques.
3. **Correction d'erreurs quantiques**, qui peut nécessiter **beaucoup plus de qubits physiques**.

Nombre minimal de qubits logiques

Un article chinois publié sur **arxiv.org** (sans comité de lecture, donc on peut y publier à peu près n'importe quoi), estime à 372 le nombre de qubits physiques nécessaires pour casser une clef RSA de 2048 Bits (voir : **[2212.12372] Factoring integers with sublinear resources on a superconducting quantum processor**). Vu que le PC Chinois ne s'est pas opposé à cette publication, on peut penser qu'il s'agit de propagande :)

IBM en son temps (2023 de mémoire) avait indiqué qu'il faudrait 4100 qubits logiques cette fois ci, donc à peu près 4 millions de qubits physiques de type transmons. Rappelons que pour l'heure, la puce IBM Eagle 3 dispose de 127 qubits physiques.

Si on factorise dans cette estimation les données de stabilité des catqubits qui réduirait de 90% les besoins en correction d'erreur, on pourrait tabler à la louche sur 400'000 qubits physiques stables (Tomcat/catqubit voire Mode de Majorana). Autant dire que nous y serons peut être dans 15/20 ans. Il faudra alors peut être passer à des clefs 4096 bits, ou plus.

Il y a aussi la mise au point d'algorithmes résistants aux attaques quantiques comme le lattice-based cryptography. La cryptographie basée sur les réseaux euclidiens (Lattice-Based Cryptography, LBC) est une famille de systèmes cryptographiques qui reposent sur la difficulté des problèmes mathématiques liés aux réseaux euclidiens. Contrairement à RSA ou aux courbes elliptiques, ces systèmes sont résistants aux attaques des ordinateurs quantiques, y compris l’algorithme de Shor.

Cette approche est actuellement considérée comme l'une des meilleures candidates pour remplacer RSA et ECC (Elliptic Curve Cryptography) dans l'ère post-quantique.