Cryptographie, algorithmique quantique et crypto post-quantique

Abel Laval

March 4, 2024

La crytographie, qu'est-ce que c'est ?

Definition

La cryptographie c'est l'ensemble des méhodes mathématiques et algorithmiques qui permettent de sécuriser des données.

Cas d'usage le plus simple : le chiffrement

Les deux grandes catégories en cryptographie

Crpytographie symétrique : Une seule clef !

Alice
clef secrète

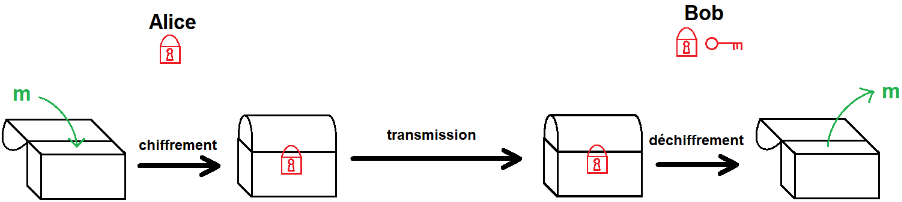
Bob
clef secrète

Cryptographie asymétrique : Deux clefs !

Alice
clef publique

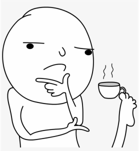
Bob
clef privée
clef publique

On s'intéresse ici à la cryptographie asymétrique.

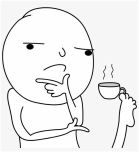


⇝ Objectif de la cryptographie symétrique : créer un "cadenas mathématique"

Qu'est-ce qu'un cadenas, au fond ?



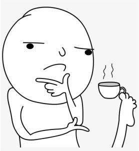
Qu'est-ce qu'un cadenas, au fond ?



C'est un objet avec trois propriétés :

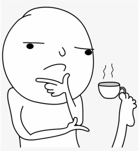
Cadenas et problèmes difficiles

Qu'est-ce qu'un cadenas, au fond ?



C'est un objet avec trois propriétés :

1. Facile à fermer

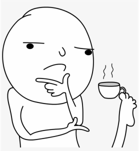


Qu'est-ce qu'un cadenas, au fond ?

C'est un objet avec trois propriétés :

1. Facile à fermer
2. Difficile à ouvrir

Qu'est-ce qu'un cadenas, au fond ?



C'est un objet avec trois propriétés :

1. Facile à fermer
2. Difficile à ouvrir
3. Facile à ouvrir si on dispose d'un objet particulier : la clef

Le problème de la factorisation

L'exemple le plus simple de cadenas mathématique :

Problème 1 : multiplication

Calculer 3×5.

L'exemple le plus simple de cadenas mathématique :

Problème 1 : multiplication

Calculer 3×5.

Facile !

Le problème de la factorisation

L'exemple le plus simple de cadenas mathématique :

Problème 1 : multiplication

Calculer 3×5.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 15$.

Le problème de la factorisation

L'exemple le plus simple de cadenas mathématique :

Problème 1 : multiplication

Calculer 3×5.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 15$.

Facile !

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Facile !

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 28560681366734964131$.

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 28560681366734964131$.

Difficile !

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 28560681366734964131$.

Difficile !

Difficile $\neq$ On ne connait pas d'algorithme pour résoudre

Le problème de la factorisation

Problème 1 : multiplication

Calculer $3458963929 \times 8257482139$.

Facile !

Problème 2 : factorisation

Trouver deux entiers a et b tels que $a \times b = 28560681366734964131$.

Difficile !

Difficile $\neq$ On ne connait pas d'algorithme pour résoudre

Definition (Problème Difficile)

Un problème est dit difficile si le temps nécessaire pour le résoudre croît plus que polynomialement en la taille de l'entrée.

À quoi sert la cryptographie asymétrique ?

- Chiffrement
- Signature électronique
- Protocole d'échange de clefs
- Fonction de hachage

À quoi sert la cryptographie asymétrique ?

- Chiffrement
- Signature électronique
- Protocole d'échange de clefs
- Fonction de hachage
- Zero-knowledge proofs
- Group signature
- Ring signature
- Oblivious transfer
- Verfiable RNG
- Homomorphic encryption
- Threshold signature
- Verifiable delay functions
- Etc...

À quoi sert la cryptographie asymétrique ?

- Chiffrement
- Signature électronique
- Protocole d'échange de clefs
- Fonction de hachage
- **Zero-knowledge proofs**
- Group signature
- **Ring signature**
- Oblivious transfer
- Verfiable RNG
- **Homomorphic encryption**
- Threshold signature
- Verifiable delay functions
- Etc...

À quoi sert la cryptographie asymétrique ?

- Chiffrement
- Signature électronique
- Protocole d'échange de clefs
- Fonction de hachage
- **Zero-knowledge proofs**
- Group signature
- **Ring signature**
- Oblivious transfer
- Verfiable RNG
- **Homomorphic encryption**
- Threshold signature
- Verifiable delay functions
- Etc...

À quoi ça sert, tout ça ?

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.

Cas d'usage de la cryptographie asymétrique

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

Cas d'usage de la cryptographie asymétrique

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

- **Scenario 2** : Vous voulez faire fuiter des informations sans qu'on ne puisse remonter jusqu'à vous.

Cas d'usage de la cryptographie asymétrique

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

- **Scenario 2** : Vous voulez faire fuiter des informations sans qu'on ne puisse remonter jusqu'à vous.
  - *Vous êtes Edward Snowden.*

Cas d'usage de la cryptographie asymétrique

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

- **Scenario 2** : Vous voulez faire fuiter des informations sans qu'on ne puisse remonter jusqu'à vous.
  - *Vous êtes Edward Snowden.*
  - *Vous êtes Julian Assange.*

Cas d'usage de la cryptographie asymétrique

Il existe un très large panel d'applications. Quelques exemples :

- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

- **Scenario 2** : Vous voulez faire fuiter des informations sans qu'on ne puisse remonter jusqu'à vous.
  - *Vous êtes Edward Snowden.*
  - *Vous êtes Julian Assange.*

- **Scenario 3** : Vous voulez chiffrer des données et effectuer des calculs sur les données chiffrées.

Il existe un très large panel d'applications. Quelques exemples :

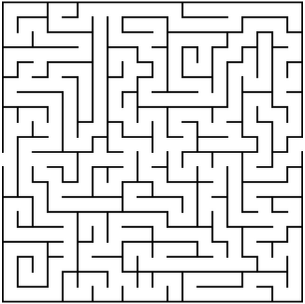
- **Scenario 1** : Vous voulez prouver que vous êtes en possession d'une certaine donnée secrète sans rien réveler de cette donnée.
  - *S'authentifier sur un site internet sans que ce dernier ne stocke les mots de passe (hachés) dans une base de données.*
  - *Prouver qu'on est bien le possesseur d'un certain ledger Bitcoin.*

- **Scenario 2** : Vous voulez faire fuiter des informations sans qu'on ne puisse remonter jusqu'à vous.
  - *Vous êtes Edward Snowden.*
  - *Vous êtes Julian Assange.*

- **Scenario 3** : Vous voulez chiffrer des données et effectuer des calculs sur les données chiffrées.
  - *Effectuer des calculs distribués dans le cloud sur des données confidentielles.*

Les preuves à divulgation nulle de connaissance
(a.k.a Zero-knowledge proofs)

Les preuves à divulgation nulle de connaissance

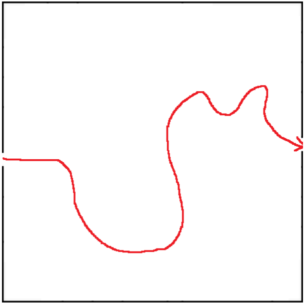


Imaginez que vous voulez prouver que vous connaissez la solution à ce labyinthe sans dévoiler cette dernière.

Comment faire ?

On commence par abstractiser le problème :

- On suppose qu'on dispose d'un immense labyrinthe.
- Il y a un très grand nombre de chemins différents qui relient l'entrée et la sortie.
- On ne représente même pas l'intérieur du labyrinthe.



Les preuves à divulgation nulle de connaissance

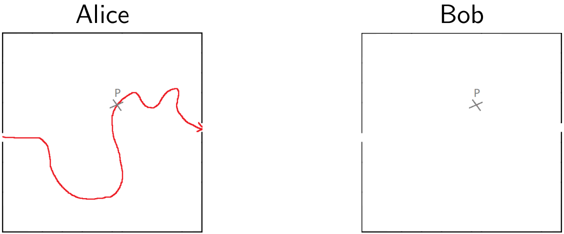
Supposons qu'Alice connaisse des chemins entre l'entrée et la sortie et veut le prouver à Bob

Cela se fait en trois étapes :

Supposons qu'Alice connaisse des chemins entre l'entrée et la sortie et veut le prouver à Bob

Cela se fait en trois étapes :

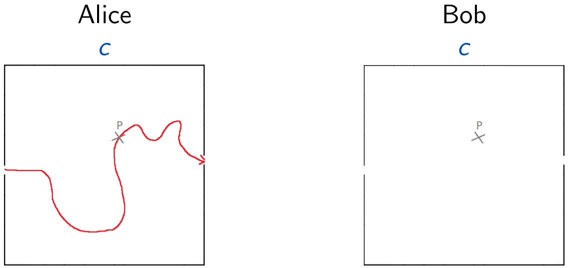
Étape 1 : Alice choisit un chemin aléatoire, un point P sur ce chemin et l'envoie à Bob.
Ce point P est le *commitment*.



Supposons qu'Alice connaisse des chemins entre l'entrée et la sortie et veut le prouver à Bob

Cela se fait en trois étapes :

Étape 2 : Bob choisit au hasard une direction $c \in \{\text{Gauche}, \text{Droite}\}$.
Cette direction c est le *challenge*.

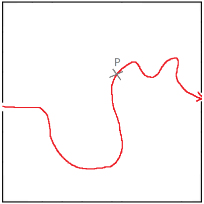


Supposons qu'Alice connaisse des chemins entre l'entrée et la sortie et veut le prouver à Bob

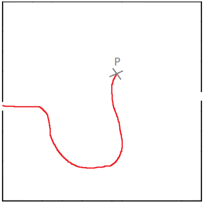
Cela se fait en trois étapes :

Étape 3 : Si le challenge est "Gauche", Alice révèle le chemin qui va de l'entrée à P; sinon elle révèle celui qui va de P à la sortie .
Ce chemin est la *réponse*.
Bob est satisfait si le chemin relie bien P à la bonne extrémité du labyrinthe.

Alice
$c = \text{Gauche}$



Bob
$c = \text{Gauche}$



On suppose qu'Alice arrive à répondre correctement au challenge. Qu'est-ce que cela prouve ?
Il faut se poser deux questions :

- Pr(Alice répond correctement **sachant que** elle connait un chemin) ?
- Pr(Alice répond correctement **sachant que** elle ne connait pas de chemin) ?

On suppose qu'Alice arrive à répondre correctement au challenge. Qu'est-ce que cela prouve ?

Il faut se poser deux questions :

- Pr(Alice répond correctement **sachant que** elle connait un chemin) ?
  100%
- Pr(Alice répond correctement **sachant que** elle ne connait pas de chemin) ?

On suppose qu'Alice arrive à répondre correctement au challenge. Qu'est-ce que cela prouve ?
Il faut se poser deux questions :

- Pr(Alice répond correctement **sachant que** elle connait un chemin) ?
  100%
- Pr(Alice répond correctement **sachant que** elle ne connait pas de chemin) ?
  50%

En trichant, Alice a 50% de chances de répondre correctement.

On suppose qu'Alice arrive à répondre correctement au challenge. Qu'est-ce que cela prouve ?
Il faut se poser deux questions :

- Pr(Alice répond correctement **sachant que** elle connait un chemin) ?
  100%
- Pr(Alice répond correctement **sachant que** elle ne connait pas de chemin) ?
  50%

En trichant, Alice a 50% de chances de répondre correctement.
Il suffit de répéter la procédure pour faire baisser cette probabilité !

On suppose qu'Alice arrive à répondre correctement au challenge. Qu'est-ce que cela prouve ?
Il faut se poser deux questions :

- $\Pr$(Alice répond correctement **sachant que** elle connait un chemin) ?
  100%
- $\Pr$(Alice répond correctement **sachant que** elle ne connait pas de chemin) ?
  50%

En trichant, Alice a 50% de chances de répondre correctement.
Il suffit de répéter la procédure pour faire baisser cette probabilité !

Avec k répétitions, la probabilité de réussir à tricher est de $1/2^k$.
C'est inférieur à 1 chance sur un milliard pour $k = 30$.

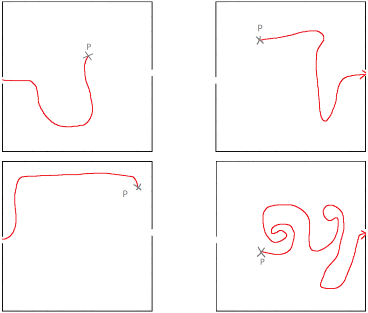
Le point crucial de ce système c'est de respecter l'ordre des étapes

$$
\begin{array}{ccc}
\text{Alice} & & \text{Bob} \\
& \xrightarrow{\text{commitment}} & \\
& \xleftarrow{\text{challenge}} & \\
& \xrightarrow{\text{response}} &
\end{array}
$$

Bob finit par être convaincu que parce qu'Alice a envoyé le *commitment* P avant de recevoir le *challenge*.

Question : Est-il possible pour Bob de retrouver un chemin reliant l'entrée et la sortie grâce aux informations transmises par Alice ?

On peut représenter l'ensemble des informations que Bob a obtenu :



Ces informations ne valent rien en dehors du protocole !

Vers l'algorithmique quantique : la cryptanalyse de RSA

RSA-KeyGen

1. Choisir deux **très grands** nombres premiers p et q. Calculer $N = pq$.
2. À partir de N, calculer deux entiers e et d avec certaines propriétés particulières.
3. Renvoyer la clef privée (N, d) et la clef publique (N, e).

RSA-Encrypt(m,N,e)

1. Renvoyer $c := m^e \bmod N$.

RSA-Decrypt(c,N,d)

1. Renvoyer $m := c^d \bmod N$.

Réussir à factoriser $N \Longrightarrow$ Retrouver la clef secrète (N, d) !

On connait un algorithme qui permet de factoriser des grands nombres et donc de casser RSA : *l'algorithme de Shor*.
Problème : c'est un algorithme quantique.

Question : À quoi ressemble concrètement un algorithme quantique ?

L'algorithme de Shor

L'algorithme de Shor

1. Choisir un entier $a \in [2, N-1]$ aléatoirement.
2. Calculer le plus petit entier r tel que $a^r = 1 \bmod N$.
3. Si r est impair, ou si $a^{r/2} + 1$ est un multiple de N, on revient à l'étape 1.
4. Sinon, retourner $\mathrm{pgcd}(a^{r/2} \pm 1, N)$ qui sont des facteurs premiers non-triviaux de N.

C'est un algorithme tout simple !

L'algorithme de Shor

1. Choisir un entier $a \in [2, N-1]$ aléatoirement.
2. Calculer le plus petit entier r tel que $a^r = 1 \bmod N$.
3. Si r est impair, ou si $a^{r/2} + 1$ est un multiple de N, on revient à l'étape 1.
4. Sinon, retourner $\mathrm{pgcd}(a^{r/2} \pm 1, N)$ qui sont des facteurs premiers non-triviaux de N.

C'est un algorithme tout simple !

Sauf que... l'étape 2 est elle aussi un **problème difficile** : on ne connait pas d'algorithme qui calcule r efficacement. C'est le *Period Finding Problem*.

L'algorithme de Shor

1. Choisir un entier $a \in [2, N-1]$ aléatoirement.
2. Calculer le plus petit entier r tel que $a^r = 1 \bmod N$.
3. Si r est impair, ou si $a^{r/2} + 1$ est un multiple de N, on revient à l'étape 1.
4. Sinon, retourner $\text{pgcd}(a^{r/2} \pm 1, N)$ qui sont des facteurs premiers non-triviaux de N.

C'est un algorithme tout simple !

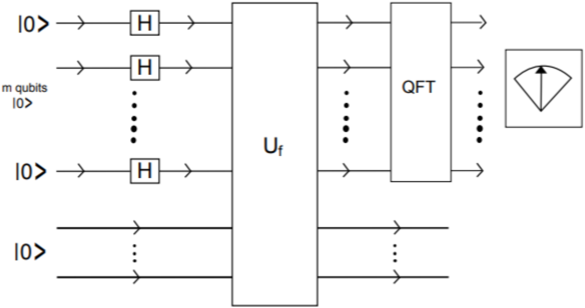
Sauf que... l'étape 2 est elle aussi un **problème difficile** : on ne connait pas d'algorithme qui calcule r efficacement. C'est le *Period Finding Problem*.

Plus exactement : on ne connait pas d'algorithme *classique* qui résolve ce problème. Mais il y a un algorithme quantique qui peut le faire.

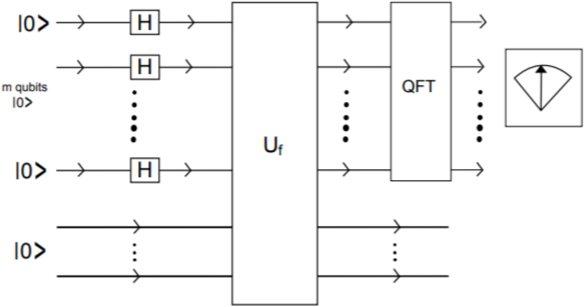
Le Period Finding Problem

Pour résoudre ce problème il suffit d'utiliser l'algorithme suivant :

Pour résoudre ce problème il suffit d'utiliser l'algorithme suivant :



Pour résoudre ce problème il suffit d'utiliser l'algorithme suivant :



C'est ce qu'on appelle un *circuit quantique*.

Un bit classique = 0 ou 1.
Un bit quantique (qubit) = un "mélange" de 0 et 1.

Un bit classique $= 0$ ou 1.
Un bit quantique (qubit) $=$ un "mélange" de 0 et 1.

Plus exactement, un qubit est de la forme :

$$a|0\rangle + b|1\rangle$$

Où a et b sont des nombres complexes tels que $|a|^2 + |b|^2 = 1$.

Un bit classique = 0 ou 1.
Un bit quantique (qubit) = un "mélange" de 0 et 1.

Plus exactement, un qubit est de la forme :

$$a|0\rangle + b|1\rangle$$

Où a et b sont des nombres complexes tels que $|a|^2 + |b|^2 = 1$.

Exemples de qubits :

- $|0\rangle$
- $|1\rangle$
- $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
- $\frac{i}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \longleftarrow$ Pas un qubit valide

Une très rapide introduction à l'algorithmique quantique

On peut "concatener" des qubits pour former des n-qubits.
Cette opération est notée $\otimes$:

$$(a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

On peut vérifier que si les deux 1-qubits de gauche sont valides, alors le 2-qubit de droite l'est aussi. Autrement dit, on a

$$|ac|^2 + |ad|^2 + |bc|^2 + |bd|^2 = 1$$

Pour manipuler des qubits, on utilise des *portes logiques quantiques*. On peut citer :

- Le 'OU' logique : $a|0\rangle + b|1\rangle \mapsto b|0\rangle + a|1\rangle$

Pour manipuler des qubits, on utilise des *portes logiques quantiques*. On peut citer :

- Le 'OU' logique : $a|0\rangle + b|1\rangle \mapsto b|0\rangle + a|1\rangle$
- La porte de Hadamard : $|0\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

Pour manipuler des qubits, on utilise des *portes logiques quantiques*. On peut citer :

- Le 'OU' logique : $a|0\rangle + b|1\rangle \mapsto b|0\rangle + a|1\rangle$
- La porte de Hadamard : $|0\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
- La transformée de Fourier quantique (QFT) : $|x\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}\omega^x|1\rangle$

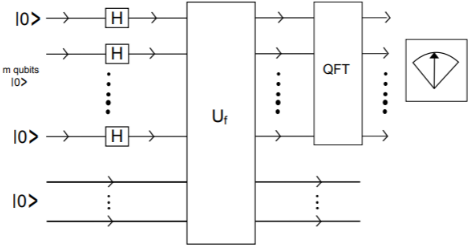
Pour manipuler des qubits, on utilise des *portes logiques quantiques*. On peut citer :

- Le 'OU' logique : $a|0\rangle + b|1\rangle \mapsto b|0\rangle + a|1\rangle$
- La porte de Hadamard : $|0\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
- La transformée de Fourier quantique (QFT) : $|x\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}\omega^x|1\rangle$

Il existe de plus une opération spéciale : la mesure de l'état d'un qubit.

- C'est comme ça qu'on récupère le résultat d'un circuit quantique.
- C'est une opération probabiliste.
- l'appliquer "détruit" l'état du qubit $\rightsquigarrow$ c'est une opération non-réversible.

$$\mathrm{Mesure}(a|0\rangle + b|1\rangle) = \begin{cases} 0 & \text{avec probabilité } |a|^2 \\ 1 & \text{avec probabilité } |b|^2 \end{cases}$$



Un circuit quantique se décompose en général en 4 étapes :

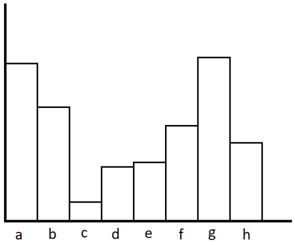
1. En entrée, on met une série de qubits purs $|0\rangle$.
2. On applique la porte de Hadamard pour "égaliser" les probabilités.
3. On applique une série de portes qui constituent le coeur du circuit.
4. On mesure l'état pour obtenir le résultat.

Visualiser l'éxecution d'un circuit quantique

Il est possible de visualiser ce que fait un circuit quantique grace à une représentation graphique.
Considérons par exemple un 3-qubit. C'est un objet de la forme

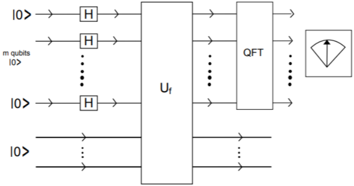
$$a|000\rangle + b|001\rangle + c|010\rangle + d|011\rangle + e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle$$

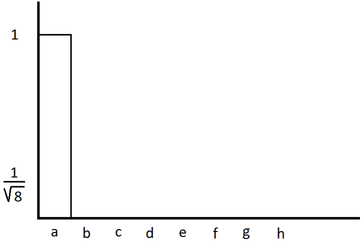
On a donc 8 coefficients qui vérifient
$|a|^2 + |b|^2 + |c|^2 + |d|^2 + |e|^2 + |f|^2 + |g|^2 + |h|^2 = 1$



La hauteur d'un bâton représente sa probabilité d'être "observé" lors de la mesure.

Étape 1 : On donne en entrée le qubit pur $|000\rangle$:

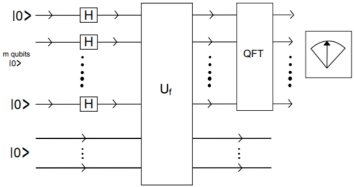


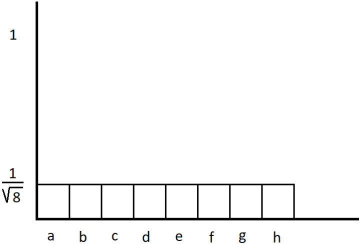


État courant :

$$1|000\rangle + 0|001\rangle + 0|010\rangle + \dots$$

Étape 2 : On "égalise" avec la porte de Hadamard :

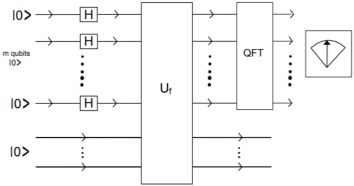


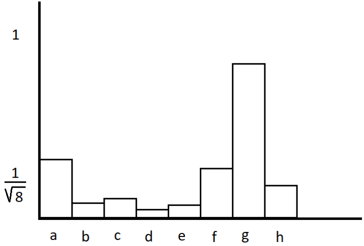


État courant :

$$\frac{1}{\sqrt{8}}|000\rangle + \frac{1}{\sqrt{8}}|001\rangle + \frac{1}{\sqrt{8}}|010\rangle + \ldots$$

Étape 3 : On applique certaines portes bien choisies :





État courant :

$$a|000\rangle + b|001\rangle + c|010\rangle + \ldots$$

Il ne reste plus qu'à mesurer !
Avec grande probabilité (égale à $|h|^2$), le résultat de la mesure sera $|110\rangle$.

Résumé sur l'algorithmique quantique

Reprenons depuis le début :

- Pour casser le chiffrement RSA, il suffit de savoir factoriser (car factoriser N en $p \times q$ permet de recalculer la clef secrète (N, d))

Résumé sur l'algorithmique quantique

Reprenons depuis le début :

- Pour casser le chiffrement RSA, il suffit de savoir factoriser (car factoriser N en $p \times q$ permet de recalculer la clef secrète (N, d))
- Pour factoriser, il suffit de savoir trouver la période d'une certaine fonction (car l'algorithme de Shor nécessite de trouver r tel que $a^r = 0 \bmod N$)

Résumé sur l'algorithmique quantique

Reprenons depuis le début :

- Pour casser le chiffrement RSA, il suffit de savoir factoriser (car factoriser N en $p \times q$ permet de recalculer la clef secrète (N, d))
- Pour factoriser, il suffit de savoir trouver la période d'une certaine fonction (car l'algorithme de Shor nécessite de trouver r tel que $a^r = 0 \bmod N$)
- Pour trouver la période de la fonction ci-dessus, il faut trouver le bon circuit quantique.

Résumé sur l'algorithmique quantique

Reprenons depuis le début :

- Pour casser le chiffrement RSA, il suffit de savoir factoriser (car factoriser N en $p \times q$ permet de recalculer la clef secrète (N, d))
- Pour factoriser, il suffit de savoir trouver la période d'une certaine fonction (car l'algorithme de Shor nécessite de trouver r tel que $a^r = 0 \bmod N$)
- Pour trouver la période de la fonction ci-dessus, il faut trouver le bon circuit quantique.
- Trouver le bon circuit quantique c'est essentiellement bien choisir quelles portes appliquer à l'étape 3 du circuit.

Avec tout ça, on peut casser RSA !

Conséquences de l'algorithme de Shor

L'algorithme de Shor permet de casser RSA.

Et alors ?

Conséquences de l'algorithme de Shor

L'algorithme de Shor permet de casser RSA.

Et alors ?

La cryptographie asymétrique actuelle ne repose que sur deux problèmes difficiles

- La factorisation : RSA

Conséquences de l'algorithme de Shor

L'algorithme de Shor permet de casser RSA.

Et alors ?

La cryptographie asymétrique actuelle ne repose que sur deux problèmes difficiles

- La factorisation : RSA
- Le problème du logarithme discret : Tout le reste

L'algorithme de Shor permet de casser RSA.

Et alors ?

La cryptographie asymétrique actuelle ne repose que sur deux problèmes difficiles

- La factorisation : RSA
- Le problème du logarithme discret : Tout le reste

Problème : l'algorithme de Shor permet aussi de résoudre le problème du logarithme discret...

Conséquences de l'algorithme de Shor

L'algorithme de Shor permet de casser RSA.

Et alors ?

La cryptographie asymétrique actuelle ne repose que sur deux problèmes difficiles

- La factorisation : RSA
- Le problème du logarithme discret : Tout le reste

Problème : l'algorithme de Shor permet aussi de résoudre le problème du logarithme discret...

Résultat : Aujourd'hui, disposer d'un ordinateur quantique assez puissant = tout casser.

La crpytographie post-quantique

La crpytographie post-quantique

Si les deux *problèmes difficiles* actuels sont rendus obsolètes par l'algorithme de Shor, il faut en trouver d'autres.

La crpytographie post-quantique

Si les deux *problèmes difficiles* actuels sont rendus obsolètes par l'algorithme de Shor, il faut en trouver d'autres.

Definition (Cryptographie post-quantique)

C'est l'ensemble des schémas de cryptographie asymétriques dont le problème difficile sous-jacent reste difficile, même pour un ordinateur quantique.
En particulier, les schémas post-quantiques ne sont pas basés sur les problèmes de la factorisation ou le logarithme discrets.

La crpytographie post-quantique

Si les deux *problèmes difficiles* actuels sont rendus obsolètes par l'algorithme de Shor, il faut en trouver d'autres.

Definition (Cryptographie post-quantique)

C'est l'ensemble des schémas de cryptographie asymétriques dont le problème difficile sous-jacent reste difficile, même pour un ordinateur quantique.
En particulier, les schémas post-quantiques ne sont pas basés sur les problèmes de la factorisation ou le logarithme discrets.

Il existe aujourd'hui cinq grandes familles en cryptographie post-quantique :

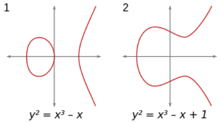
- Les réseaux euclidiens
- Les codes correcteurs d'erreurs
- **Les isogénies**
- le hachage
- les polyômes multivariés

On se repose sur deux objets principaux : les **courbes elliptiques** et les **isogénies** :

Definition (Courbe elliptique (très informel voire faux))

Une courbe elliptique c'est une courbe qui est solution d'une équation de la forme

$$y^2 = x^3 + ax + b$$



Definition (Isogénie)

Une isogénie est un morphisme (une fonction) entre deux courbes elliptiques.

Quel est le problème difficile en cryptographie à base d'isogénies ?

Quel est le problème difficile en cryptographie à base d'isogénies ?
Il s'agit de...

Quel est le problème difficile en cryptographie à base d'isogénies ?

Il s'agit de... trouver son chemin dans un labyrinthe...

Quel est le problème difficile en cryptographie à base d'isogénies ?

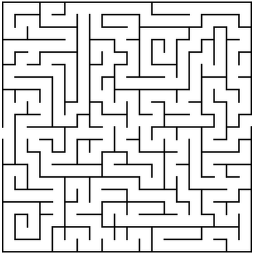
Il s'agit de... trouver son chemin dans un labyrinthe...

Labyrinthe $\leftrightarrow$ **Graph**

Quel est le problème difficile en cryptographie à base d'isogénies ?

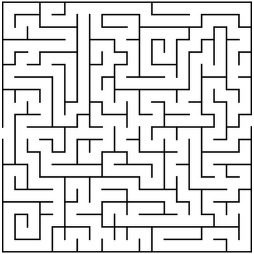
Il s'agit de... trouver son chemin dans un labyrinthe...

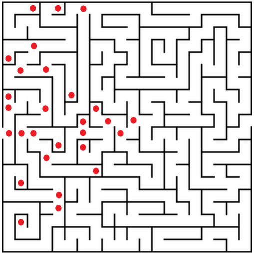
Labyrinthe $\leftrightarrow$ **Graph**



Quel est le problème difficile en cryptographie à base d'isogénies ?
Il s'agit de... trouver son chemin dans un labyrinthe...

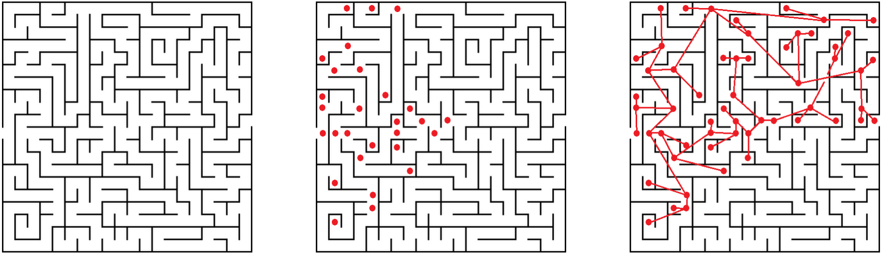
Labyrinthe $\leftrightarrow$ Graph



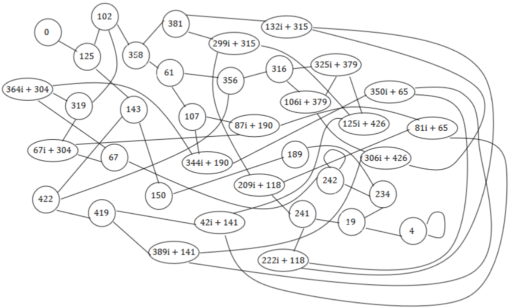


Quel est le problème difficile en cryptographie à base d'isogénies ?
Il s'agit de... trouver son chemin dans un labyrinthe...

Labyrinthe $\leftrightarrow$ **Graph**



On peut définir un graphe d'isogénies :



- Les sommets de ce graphe sont de courbes elliptiques.
- Les arrêtes sont des isogénies entre les courbes.

Il existe des preuves à divulgation nulle de connaissance sur ce genre de graphe : *Short Quaternion Isogeny Signature* (SQISign).