Quantum Computation and Cryptography
Day 4

Alexandru Gheorghiu

The University of Edinburgh

Part II
Quantum algorithms

Second problem - Simon's problem

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$
(in this case, f is a *2-to-1* function)

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$
(in this case, f is a *2-to-1* function)

Note that the *1-to-1* case is the case $s = 0^N$

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$
(in this case, f is a *2-to-1* function)

Note that the *1-to-1* case is the case $s = 0^N$

E.g. *1-to-1* $\forall x, f(x) = x$

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$
(in this case, f is a *2-to-1* function)

Note that the *1-to-1* case is the case $s = 0^N$

E.g. *1-to-1* $\forall x, f(x) = x$
2-to-1 $\forall x, f(x) = f(x \oplus 1^N)$ (so $s = 1^N$)

Second problem - Simon's problem

Say we have $f : \{0,1\}^N \to \{0,1\}^N$ as a black box

We are **promised** that f is either *1-to-1* or has *Simon's property*

1-to-1 means bijective (essentially a permutation of the domain)

Simon's property:
$\exists s \in \{0,1\}^N, s \neq 0^N, \forall x, y \in \{0,1\}^N, x \neq y$
$f(x) = f(y)$ iff $x = y \oplus s$
(in this case, f is a *2-to-1* function)

Note that the *1-to-1* case is the case $s = 0^N$

E.g. *1-to-1* $\forall x, f(x) = x$
2-to-1 $\forall x, f(x) = f(x \oplus 1^N)$ (so $s = 1^N$)

How many calls to f do we need, to determine type with high probability?

Second problem - Simon's problem

Second problem - Simon's problem

What we're looking for is a *collision*

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?

$M/2$

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?
$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?
$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Well there are $k(k-1)/2$ possible pairs so...

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?

$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Well there are $k(k-1)/2$ possible pairs so...

$$\frac{k(k-1)/2}{M/2} = k(k-1)/M$$

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?
$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Well there are $k(k-1)/2$ possible pairs so...
$\frac{k(k-1)/2}{M/2} = k(k-1)/M$

So if $k = O(\sqrt{M})$, the probability will be high enough

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?
$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Well there are $k(k-1)/2$ possible pairs so...
$\frac{k(k-1)/2}{M/2} = k(k-1)/M$

So if $k = O(\sqrt{M})$, the probability will be high enough

In our case, that means $k = O(2^{N/2})$

Second problem - Simon's problem

What we're looking for is a *collision*

I.e. an x and a y, $x \neq y$ such that $f(x) = f(y)$

Denote the size of the domain of f as $M = 2^N$

Birthday paradox

If f is *2-to-1*, how many collision pairs are there?
$M/2$

Suppose I take k (different) elements at random
What is the probability that *any* two of them is a collision pair?

Well there are $k(k-1)/2$ possible pairs so...
$\frac{k(k-1)/2}{M/2} = k(k-1)/M$

So if $k = O(\sqrt{M})$, the probability will be high enough

In our case, that means $k = O(2^{N/2})$
This is actually optimal, so: #calls to f $= \Omega(2^{N/2})$

Quantum case

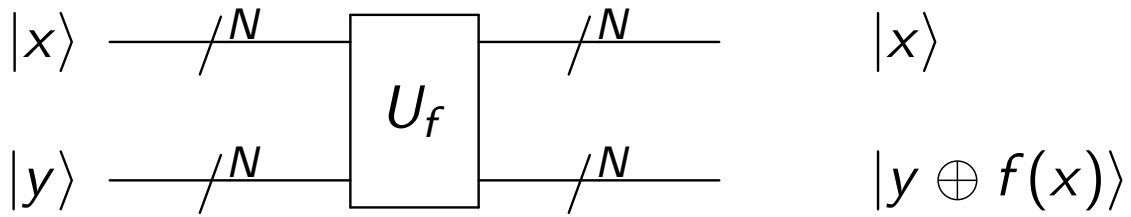
Quantum case

Again, need a quantum implementation of f, as unitary U_f

Quantum case

Again, need a quantum implementation of f, as unitary U_f

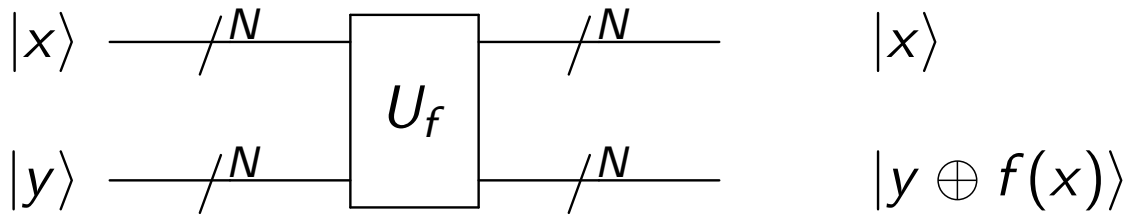
Would look like this:



Quantum case

Again, need a quantum implementation of f, as unitary U_f

Would look like this:

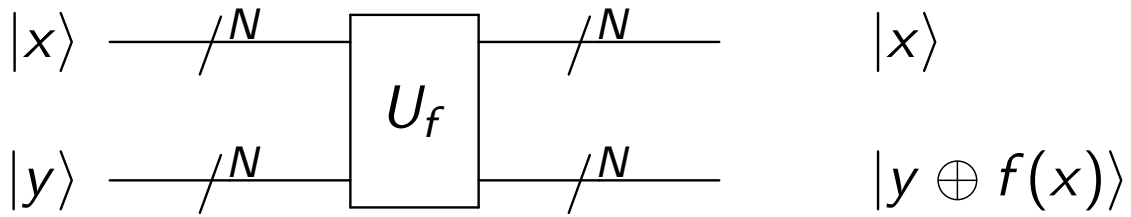


Note that in this case we have a $2N$-sized input

Quantum case

Again, need a quantum implementation of f, as unitary U_f

Would look like this:



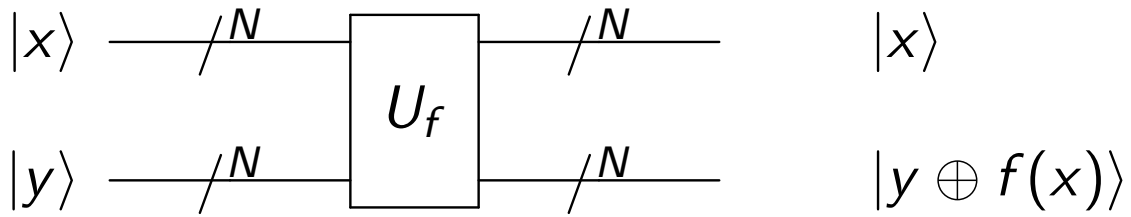
Note that in this case we have a $2N$-sized input

This time, set $|y\rangle = |0\rangle^{\otimes n}$

Quantum case

Again, need a quantum implementation of f, as unitary U_f

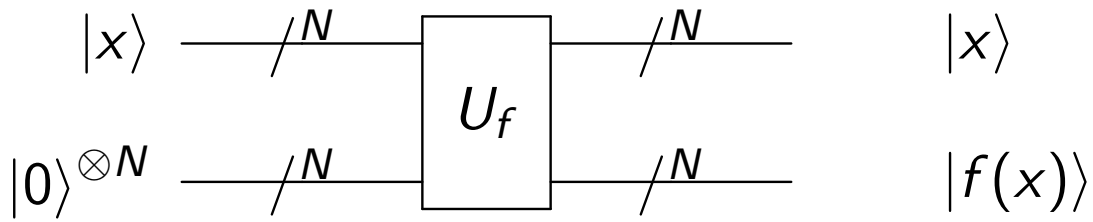
Would look like this:



Note that in this case we have a $2N$-sized input

This time, set $|y\rangle = |0\rangle^{\otimes n}$

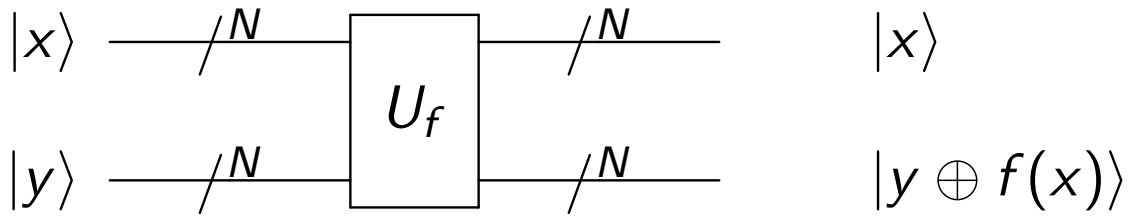
So we have:



Quantum case

Again, need a quantum implementation of f, as unitary U_f

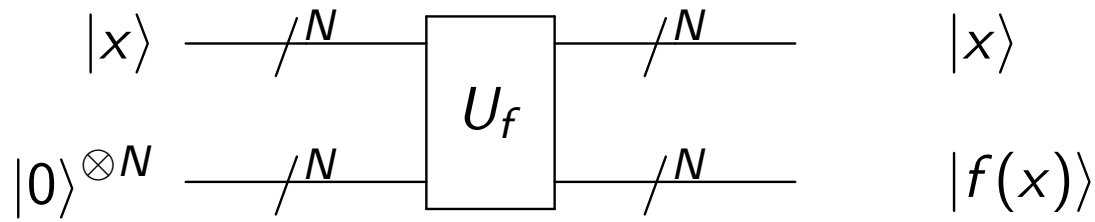
Would look like this:



Note that in this case we have a $2N$-sized input

This time, set $|y\rangle = |0\rangle^{\otimes n}$

So we have:



Call the upper part register 1 and the lower part register 2

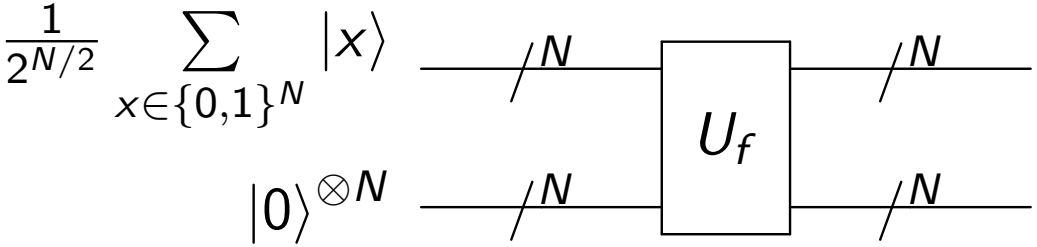
Quantum case

Quantum case

Suppose we put a superposition over all x's in the first register

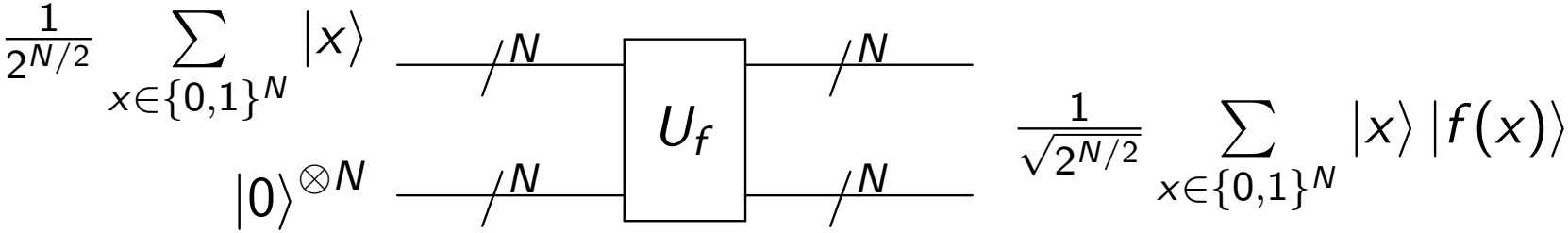
Quantum case

Suppose we put a superposition over all x's in the first register



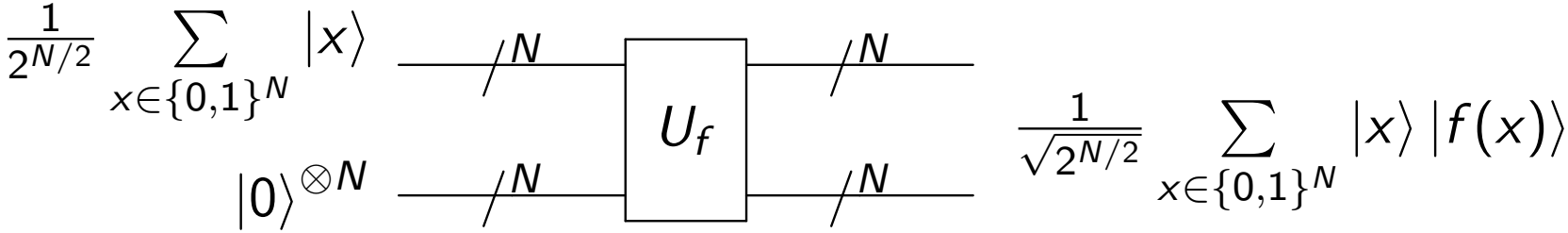
Quantum case

Suppose we put a superposition over all x's in the first register



Quantum case

Suppose we put a superposition over all x's in the first register

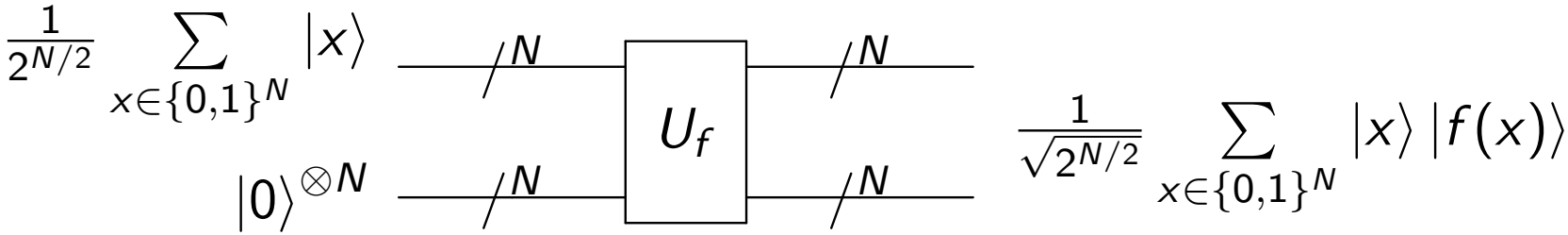


Suppose the function is *2-to-1*

What do we get if we measure the second register?

Quantum case

Suppose we put a superposition over all x's in the first register

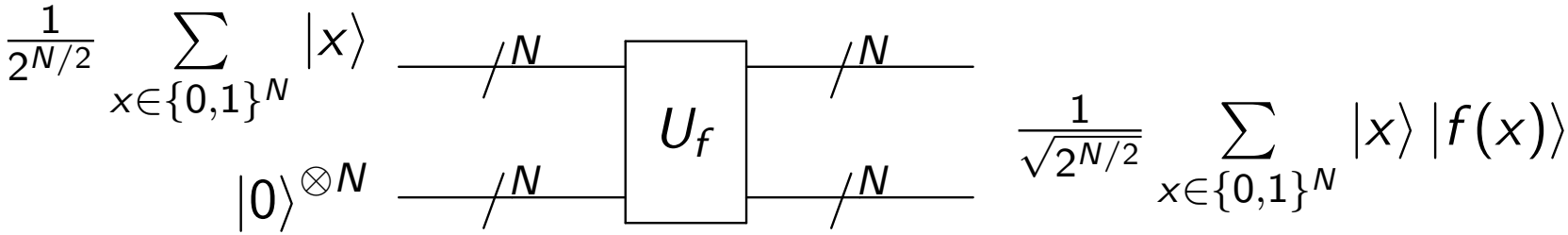


Suppose the function is *2-to-1*
What do we get if we measure the second register?

$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle)\,|f(z)\rangle$$

Quantum case

Suppose we put a superposition over all x's in the first register



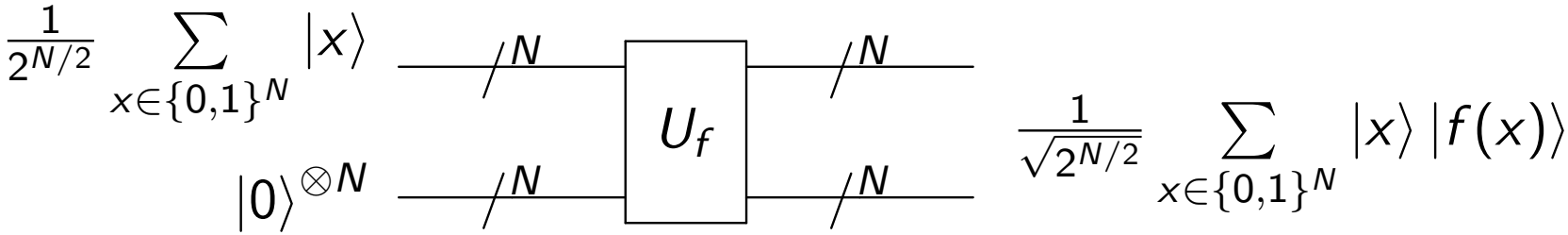
Suppose the function is *2-to-1*
What do we get if we measure the second register?

$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle)\,|f(z)\rangle$$

What if the function were *1-to-1*?

Quantum case

Suppose we put a superposition over all x's in the first register



Suppose the function is *2-to-1*
What do we get if we measure the second register?

$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle)\,|f(z)\rangle$$

What if the function were *1-to-1*?

$$|z\rangle\,|f(z)\rangle$$

Quantum case

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

The above state is what we get in the *1-to-1* case

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

The above state is what we get in the *1-to-1* case

In the *2-to-1* case we get:

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

The above state is what we get in the *1-to-1* case

In the *2-to-1* case we get:

$$\frac{1}{\sqrt{2}} \frac{1}{2^{N/2}} \left(\sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle + \sum_{y \in \{0,1\}^N} (-1)^{(z \oplus s) \cdot y} |y\rangle \right)$$

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

The above state is what we get in the *1-to-1* case

In the *2-to-1* case we get:

$$\frac{1}{\sqrt{2}} \frac{1}{2^{N/2}} \left(\sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle + \sum_{y \in \{0,1\}^N} (-1)^{(z \oplus s) \cdot y} |y\rangle \right)$$

$$\frac{1}{\sqrt{2}} \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} \left((-1)^{z \cdot y} + (-1)^{(z \oplus s) \cdot y} \right) |y\rangle$$

Quantum case

Recall the useful relation...

$$H^{\otimes N} |z\rangle = \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle$$

Suppose we apply $H^{\otimes N}$ to the first register

The above state is what we get in the *1-to-1* case

In the *2-to-1* case we get:

$$\frac{1}{\sqrt{2}} \frac{1}{2^{N/2}} \left(\sum_{y \in \{0,1\}^N} (-1)^{z \cdot y} |y\rangle + \sum_{y \in \{0,1\}^N} (-1)^{(z \oplus s) \cdot y} |y\rangle \right)$$

$$\frac{1}{\sqrt{2}} \frac{1}{2^{N/2}} \sum_{y \in \{0,1\}^N} \left((-1)^{z \cdot y} + (-1)^{(z \oplus s) \cdot y} \right) |y\rangle$$

What do we get if we measure this state?

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

$$s\cdot r=0$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

$$s\cdot r=0$$

Our measurement outcome is some **random** r such that:

$$s\cdot r=0$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}\left((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y}\right)|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

$$s\cdot r=0$$

Our measurement outcome is some **random** r such that:

$$s\cdot r=0$$

I.e.:

$$s_1r_1\oplus s_2r_2\oplus\ldots\oplus s_Nr_N=0$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y})\,|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

$$s\cdot r=0$$

Our measurement outcome is some **random** r such that:

$$s\cdot r=0$$

I.e.:

$$s_1r_1\oplus s_2r_2\oplus\ldots\oplus s_Nr_N=0$$

If we do the whole thing again, we get an r' (likely $r'\neq r$):

$$s_1r'_1\oplus s_2r'_2\oplus\ldots\oplus s_Nr'_N=0$$

Quantum case

$$\frac{1}{\sqrt{2}}\frac{1}{2^{N/2}}\sum_{y\in\{0,1\}^N}((-1)^{z\cdot y}+(-1)^{(z\oplus s)\cdot y})\,|y\rangle$$

Well, we'll get some $|r\rangle$ with the property that:

$$z\cdot r=(z\oplus s)\cdot r$$

$$z\cdot r=z\cdot r\oplus s\cdot r$$

$$s\cdot r=0$$

Our measurement outcome is some **random** r such that:

$$s\cdot r=0$$

I.e.:

$$s_1r_1\oplus s_2r_2\oplus\ldots\oplus s_Nr_N=0$$

If we do the whole thing again, we get an r' (likely $r'\neq r$):

$$s_1r'_1\oplus s_2r'_2\oplus\ldots\oplus s_Nr'_N=0$$

What if we do this $O(N)$ times?

Quantum case

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

$$
\begin{aligned}
s_1 r_1^1 \oplus s_2 r_2^1 \oplus \ldots \oplus s_N r_N^1 &= 0 \\
s_1 r_1^2 \oplus s_2 r_2^2 \oplus \ldots \oplus s_N r_N^2 &= 0 \\
s_1 r_1^3 \oplus s_2 r_2^3 \oplus \ldots \oplus s_N r_N^3 &= 0 \\
&\ldots
\end{aligned}
$$

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

$$
\begin{aligned}
s_1 r_1^1 \oplus s_2 r_2^1 \oplus \ldots \oplus s_N r_N^1 &= 0 \\
s_1 r_1^2 \oplus s_2 r_2^2 \oplus \ldots \oplus s_N r_N^2 &= 0 \\
s_1 r_1^3 \oplus s_2 r_2^3 \oplus \ldots \oplus s_N r_N^3 &= 0 \\
&\ldots
\end{aligned}
$$

We can solve this system (efficiently) and recover s

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

$$
\begin{aligned}
s_1 r_1^1 \oplus s_2 r_2^1 \oplus \ldots \oplus s_N r_N^1 &= 0 \\
s_1 r_1^2 \oplus s_2 r_2^2 \oplus \ldots \oplus s_N r_N^2 &= 0 \\
s_1 r_1^3 \oplus s_2 r_2^3 \oplus \ldots \oplus s_N r_N^3 &= 0 \\
&\ldots
\end{aligned}
$$

We can solve this system (efficiently) and recover s

Only required $O(N)$ queries to f

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

$$
\begin{aligned}
s_1 r_1^1 \oplus s_2 r_2^1 \oplus \ldots \oplus s_N r_N^1 &= 0 \\
s_1 r_1^2 \oplus s_2 r_2^2 \oplus \ldots \oplus s_N r_N^2 &= 0 \\
s_1 r_1^3 \oplus s_2 r_2^3 \oplus \ldots \oplus s_N r_N^3 &= 0 \\
&\ldots
\end{aligned}
$$

We can solve this system (efficiently) and recover s

Only required $O(N)$ queries to f

Classically, we needed $\Omega(2^{N/2})$

Quantum case

Repeating $O(N)$ times, with high probability, we get a system of equations:

$$
\begin{aligned}
s_1 r_1^1 \oplus s_2 r_2^1 \oplus \ldots \oplus s_N r_N^1 &= 0 \\
s_1 r_1^2 \oplus s_2 r_2^2 \oplus \ldots \oplus s_N r_N^2 &= 0 \\
s_1 r_1^3 \oplus s_2 r_2^3 \oplus \ldots \oplus s_N r_N^3 &= 0 \\
&\ldots
\end{aligned}
$$

We can solve this system (efficiently) and recover s

Only required $O(N)$ queries to f

Classically, we needed $\Omega(2^{N/2})$

Exponential speed-up!

Questions so far?

Questions so far?

Time to break some classical crypto...

Questions so far?

Time to break some classical crypto...

But first a short primer on RSA

Public-key crypto 101

Public-key crypto 101

RSA, ElGamal, ECC are *public-key* cryptosystems

Public-key crypto 101

RSA, ElGamal, ECC are *public-key* cryptosystems

What's that?

Public-key crypto 101

RSA, ElGamal, ECC are *public-key* cryptosystems

What's that?

Say you want to buy an app

Public-key crypto 101

RSA, ElGamal, ECC are *public-key* cryptosystems

What's that?

Say you want to buy an app

Public-key crypto 101

RSA, ElGamal, ECC are *public-key* cryptosystems

What's that?

Say you want to buy an app

You need to securely send your payment details (card number etc)

Public-key crypto 101

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key PK*, *secret key SK*)

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key* PK, *secret key* SK)
2. Broadcasts PK (assume that it's validated by some authority)

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key* PK, *secret key* SK)
2. Broadcasts PK (assume that it's validated by some authority)
3. Client uses PK to encrypt payment details

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key* PK, *secret key* SK)
2. Broadcasts PK (assume that it's validated by some authority)
3. Client uses PK to encrypt payment details
4. Sends encrypted content to Appstore

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key PK*, *secret key SK*)
2. Broadcasts *PK* (assume that it's validated by some authority)
3. Client uses *PK* to encrypt payment details
4. Sends encrypted content to Appstore
5. Appstore decrypts with *SK*

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key* PK, *secret key* SK)
2. Broadcasts PK (assume that it's validated by some authority)
3. Client uses PK to encrypt payment details
4. Sends encrypted content to Appstore
5. Appstore decrypts with SK

What an eavesdropper can know: PK, encrypted messages

Public-key crypto 101

Asymmetric encryption idea:

- There are 2 keys: K_1, K_2
- Messages encrypted with K_1 can only be decrypted with K_2

1. Appstore generates pair: (*public key* PK, *secret key* SK)
2. Broadcasts PK (assume that it's validated by some authority)
3. Client uses PK to encrypt payment details
4. Sends encrypted content to Appstore
5. Appstore decrypts with SK

What an eavesdropper can know: PK, encrypted messages

Eavesdropper should not be able to decrypt (even partially) encrypted messages

Public-key crypto 101

Public-key crypto 101

A bit more precise...

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

One-way function = easy to compute, hard to invert

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

One-way function = easy to compute, hard to invert
Trapdoor = easy to invert given some extra information (SK)

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

One-way function = easy to compute, hard to invert
Trapdoor = easy to invert given some extra information (SK)

Easy = polynomial time
Hard = not polynomial time (ideally exponential or worse)

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

One-way function = easy to compute, hard to invert
Trapdoor = easy to invert given some extra information (SK)

Easy = polynomial time
Hard = not polynomial time (ideally exponential or worse)

Do such functions exist?

Public-key crypto 101

A bit more precise...

Encryption procedure is some function $E(K, M) = C$

E.g. client computes $E(PK, \text{card number}) = C$

E is a **trapdoor one-way function**

One-way function = easy to compute, hard to invert
Trapdoor = easy to invert given some extra information (SK)

Easy = polynomial time
Hard = not polynomial time (ideally exponential or worse)

Do such functions exist?

We don't know! But it's strongly believed that they do

RSA 101

RSA 101

Suppose we pick two odd prime numbers P and Q

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

How many group elements?

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

How many group elements?

8

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

How many group elements?
8

What are the group elements?

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

How many group elements?
8

What are the group elements?
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

RSA 101

Suppose we pick two odd prime numbers P and Q

Let $N = P \times Q$

The integers co-prime with N form a group under multiplication (call it $\mathcal{G}$)

Group order is $(P-1)(Q-1)$ (Euler)

E.g. let $P = 3$, $Q = 5$

How many group elements?
8

What are the group elements?
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

For some group element g, we can efficiently find it's inverse, i.e. h such that $hg = 1 \ mod \ N$

RSA 101

RSA 101

Suppose M is the message and $M < N$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)
6. $D(SK, C) = C^d \ mod \ N$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)
6. $D(SK, C) = C^d \ mod \ N = M^{de} \ mod \ N = M \ mod \ N$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)
6. $D(SK, C) = C^d \ mod \ N = M^{de} \ mod \ N = M \ mod \ N$

The attacker knows N, e, C, but would need d to recover M

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)
6. $D(SK, C) = C^d \ mod \ N = M^{de} \ mod \ N = M \ mod \ N$

The attacker knows N, e, C, but would need d to recover M

The attacker would have to determine
$(P-1)(Q-1) = PQ - P - Q - 1$

RSA 101

Suppose M is the message and $M < N$
(we can break long messages in chunks of size $< N$)

1. Pick two large prime numbers P and Q ($N = P \times Q$)
2. Pick e in $\mathcal{G}$ such that $e \neq 1$
3. Compute the inverse of e, call it d (so $de = 1 \ mod \ N$)
4. $PK = (N, e)$, $SK = d$
5. $E(PK, M) = M^e \ mod \ N$ (let $C = M^e \ mod \ N$)
6. $D(SK, C) = C^d \ mod \ N = M^{de} \ mod \ N = M \ mod \ N$

The attacker knows N, e, C, but would need d to recover M

The attacker would have to determine
$(P-1)(Q-1) = PQ - P - Q - 1$

Well just factor N and recover P and Q

Factoring

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"

Carl Friedrich Gauss

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?
It hasn't been proven, but...

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?
It hasn't been proven, but...

Naive algorithm:
Trying factors up to N is exponential in the input length

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?
It hasn't been proven, but...

Naive algorithm:
Trying factors up to N is exponential in the input length

Check out *Pollard's rho algorithm* (uses the birthday paradox)

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?
It hasn't been proven, but...

Naive algorithm:
Trying factors up to N is exponential in the input length

Check out *Pollard's rho algorithm* (uses the birthday paradox)

No known (classical) polynomial-time algorithm for factoring!
(not even probabilistic)

Factoring

"*The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.*"
Carl Friedrich Gauss

Is factoring hard?
It hasn't been proven, but...

Naive algorithm:
Trying factors up to N is exponential in the input length

Check out *Pollard's rho algorithm* (uses the birthday paradox)

No known (classical) polynomial-time algorithm for factoring!
(not even probabilistic)

But there is a quantum algorithm! (Shor's algorithm)

Factoring

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

But there could be a smaller k such that g^k *mod* $N = 1$
(e.g. 4^2 *mod* $15 = 1$)

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

But there could be a smaller k such that g^k *mod* $N = 1$
(e.g. 4^2 *mod* $15 = 1$)

The smallest such k is called the *order* of g

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

But there could be a smaller k such that g^k *mod* $N = 1$
(e.g. 4^2 *mod* $15 = 1$)

The smallest such k is called the *order* of g

The order of an element always divides $(P-1)(Q-1)$

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

But there could be a smaller k such that g^k *mod* $N = 1$
(e.g. 4^2 *mod* $15 = 1$)

The smallest such k is called the *order* of g

The order of an element always divides $(P-1)(Q-1)$

Suppose we take a few elements and compute their orders.
What is (likely to be) their least common multiplier?

Factoring

Consider again the group $\mathcal{G}$
(of order $(P-1)(Q-1)$)

And as an example think of $P = 3$, $Q = 5$
$\{1, 2, 4, 7, 8, 11, 13, 14\}$

Let $g \in \mathcal{G}$. What is $g^{(P-1)(Q-1)}$ *mod* N?
1

But there could be a smaller k such that g^k *mod* $N = 1$
(e.g. 4^2 *mod* $15 = 1$)

The smallest such k is called the *order* of g

The order of an element always divides $(P-1)(Q-1)$

Suppose we take a few elements and compute their orders.
What is (likely to be) their least common multiplier?

$$(P-1)(Q-1)$$

Order finding

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding
Input: g, N
Output: smallest k for which $g^k \ mod \ N = 1$

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding
Input: g, N
Output: smallest k for which $g^k \ mod \ N = 1$

Let $f(x) = g^x \ mod \ N$

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding
Input: g, N
Output: smallest k for which $g^k \ mod \ N = 1$

Let $f(x) = g^x \ mod \ N$
Note that $f(x) = f(x+k)$

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding
Input: g, N
Output: smallest k for which $g^k \ mod \ N = 1$

Let $f(x) = g^x \ mod \ N$
Note that $f(x) = f(x+k)$

So we want the smallest k for which
$f(x) = f(x+k)$

Order finding

Knowing $(P-1)(Q-1)$ and PQ we can easily get P and Q

Sampling elements from $\mathcal{G}$ can be done efficiently

Computing least common multiplier is also efficient

If we can find the order of some g efficiently $\rightarrow$ done!

Order finding
Input: g, N
Output: smallest k for which $g^k \ mod \ N = 1$

Let $f(x) = g^x \ mod \ N$
Note that $f(x) = f(x+k)$

So we want the smallest k for which
$f(x) = f(x+k)$

Familiar?

Shor's algorithm

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Suppose we measure the second register.
What is the state of the first?

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Suppose we measure the second register.
What is the state of the first?

$$|z\rangle + |z + k\rangle + |z + 2k\rangle + ...$$

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Suppose we measure the second register.
What is the state of the first?

$$|z\rangle + |z + k\rangle + |z + 2k\rangle + ...$$

In Simon we used $H^{\otimes T} |z\rangle = \sum_y (-1)^{z \cdot y} |y\rangle$

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Suppose we measure the second register.
What is the state of the first?

$$|z\rangle + |z + k\rangle + |z + 2k\rangle + ...$$

In Simon we used $H^{\otimes T} |z\rangle = \sum_y (-1)^{z \cdot y} |y\rangle$

Here we use a generalization called **quantum Fourier transform**:
$QFT |z\rangle = \sum_y \omega^{zy} |y\rangle$

Shor's algorithm

Simon: find s for which $f(x) = f(x \oplus s)$

We're trying to do same thing, but over a different group!

Suppose we make the state:
$\sum_x |x\rangle |f(x)\rangle$

(we're using $T = log_2 N$ qubits for $|x\rangle$, same for $|f(x)\rangle$)

Suppose we measure the second register.
What is the state of the first?

$$|z\rangle + |z + k\rangle + |z + 2k\rangle + ...$$

In Simon we used $H^{\otimes T} |z\rangle = \sum_y (-1)^{z \cdot y} |y\rangle$

Here we use a generalization called **quantum Fourier transform**:
$QFT |z\rangle = \sum_y \omega^{zy} |y\rangle$

Where ω is the 2^T'th root of unity

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + \ldots$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + \ldots$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + \ldots\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + \ldots\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

If we measure this state, the amplitude for a particular $|y\rangle$ is proportional to:

$$\left|\sum_j \omega^{jky}\right|^2$$

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

If we measure this state, the amplitude for a particular $|y\rangle$ is proportional to:

$$\left|\sum_j \omega^{jky}\right|^2$$

This is maximal (constructive interference) when ω^{ky} is close to 1

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

If we measure this state, the amplitude for a particular $|y\rangle$ is proportional to:

$$\left|\sum_j \omega^{jky}\right|^2$$

This is maximal (constructive interference) when ω^{ky} is close to 1

Which means y close to $m2^T/k$, for some integer m

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

If we measure this state, the amplitude for a particular $|y\rangle$ is proportional to:

$$\left|\sum_j \omega^{jky}\right|^2$$

This is maximal (constructive interference) when ω^{ky} is close to 1

Which means y close to $m2^T/k$, for some integer m

Repeat few times: $m_1 2^T/k$, $m_2 2^T/k$, $m_3 2^T/k$ ...

Shor's algorithm

$$|z\rangle + |z+k\rangle + |z+2k\rangle + ...$$

$$QFT\,|z\rangle = \sum_y \omega^{zy}\,|y\rangle$$

$$\sum_y \left(\omega^{zy} + \omega^{(z+k)y} + \omega^{(z+2k)y} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(1 + \omega^{ky} + \omega^{2ky} + ...\right)|y\rangle$$

$$\sum_y \omega^{zy}\left(\sum_j \omega^{jky}\right)|y\rangle$$

If we measure this state, the amplitude for a particular $|y\rangle$ is proportional to:

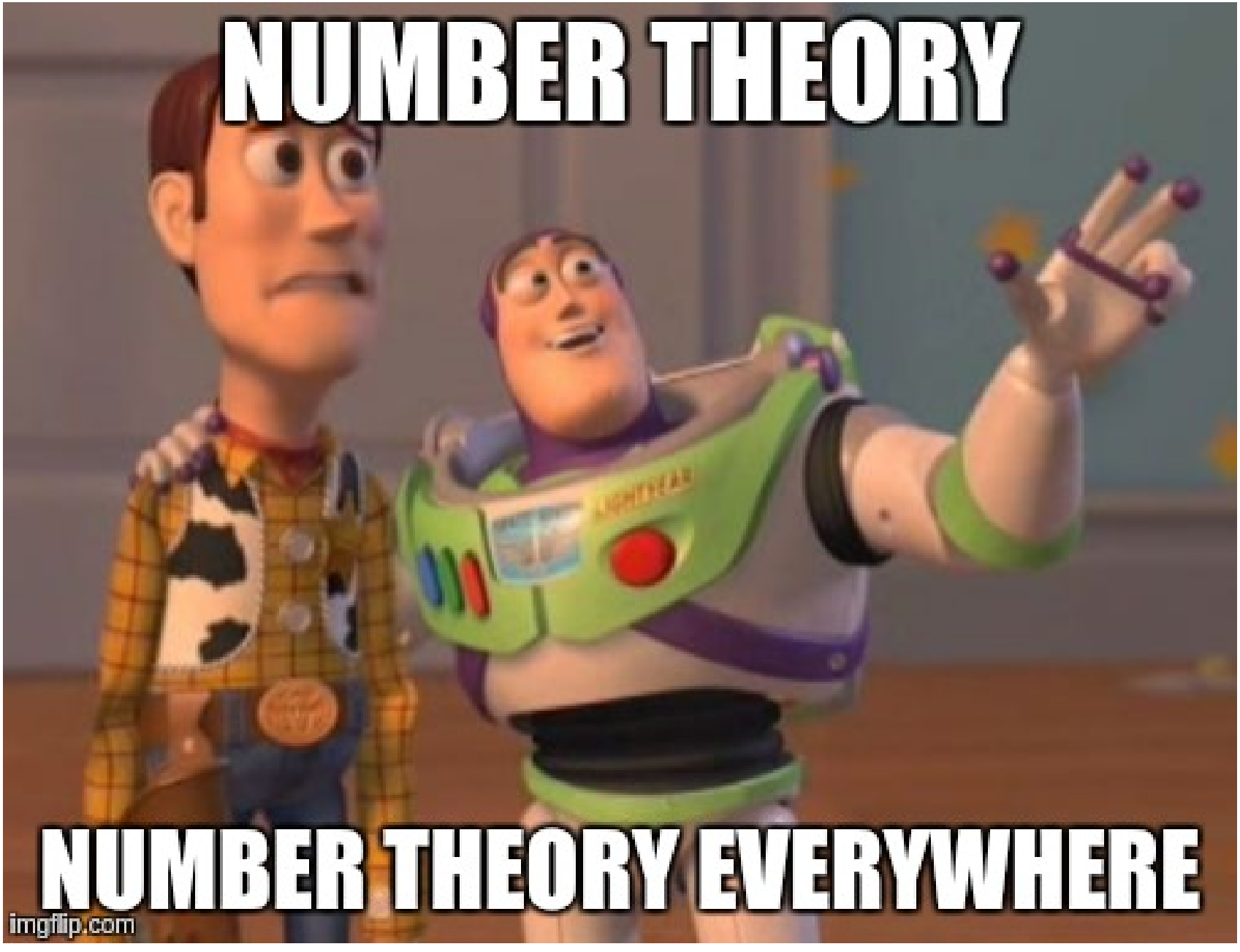
$$\left|\sum_j \omega^{jky}\right|^2$$

This is maximal (constructive interference) when ω^{ky} is close to 1

Which means y close to $m2^T/k$, for some integer m

Repeat few times: $m_1 2^T/k$, $m_2 2^T/k$, $m_3 2^T/k$ ...

We can find k!


NUMBER THEORY
NUMBER THEORY EVERYWHERE
imgflip.com

Abelian hidden subgroup problem

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Suppose there is a function f, such that:

$$f(g) = f(g + h) \textit{ iff } h \in H$$

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Suppose there is a function f, such that:

$$f(g) = f(g + h) \textit{ iff } h \in H$$

Abelian hidden subgroup problem:

Input: description of G, description of f

Output: description of H

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Suppose there is a function f, such that:

$$f(g) = f(g + h) \textit{ iff } h \in H$$

Abelian hidden subgroup problem:

Input: description of G, description of f

Output: description of H

Order finding and discrete logarithm are instances of this problem!

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Suppose there is a function f, such that:

$$f(g) = f(g + h) \textit{ iff } h \in H$$

Abelian hidden subgroup problem:

Input: description of G, description of f

Output: description of H

Order finding and discrete logarithm are instances of this problem!

RSA, Diffie-Hellman, ElGamal, ECC are based on these problems

Abelian hidden subgroup problem

In Simon, looking for s such that:

$$f(x) = f(x \oplus s)$$

In Shor, looking for those k such that:

$$f(x) = f(x + k)$$

Seems like these are particular instance of something general

Let $(G, +)$ be an Abelian group and H a subgroup of G

Suppose there is a function f, such that:

$$f(g) = f(g + h) \textit{ iff } h \in H$$

Abelian hidden subgroup problem:

Input: description of G, description of f

Output: description of H

Order finding and discrete logarithm are instances of this problem!

RSA, Diffie-Hellman, ElGamal, ECC are based on these problems

So they are insecure against a quantum computer!

Post-quantum cryptography

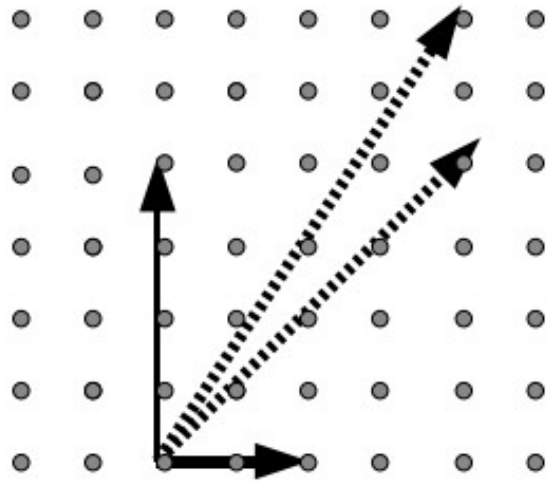
Post-quantum cryptography

Problems believed to be hard even for quantum computers

Post-quantum cryptography

Problems believed to be hard even for quantum computers

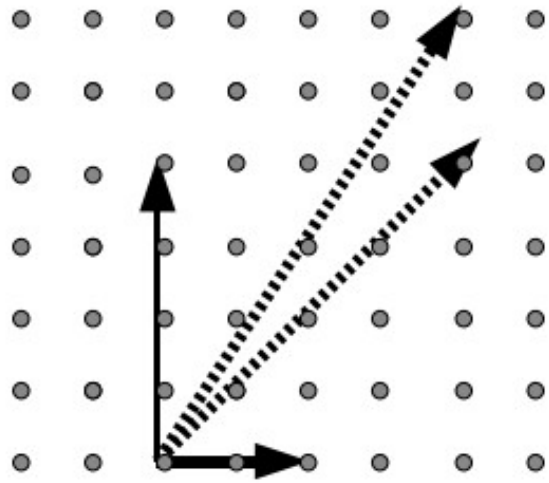
Good candidates for crypto: lattice problems



Post-quantum cryptography

Problems believed to be hard even for quantum computers

Good candidates for crypto: lattice problems

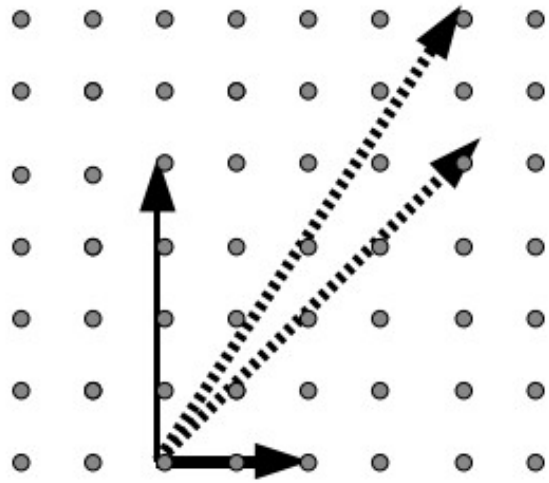


Lattices are like discrete vector spaces

Post-quantum cryptography

Problems believed to be hard even for quantum computers

Good candidates for crypto: lattice problems



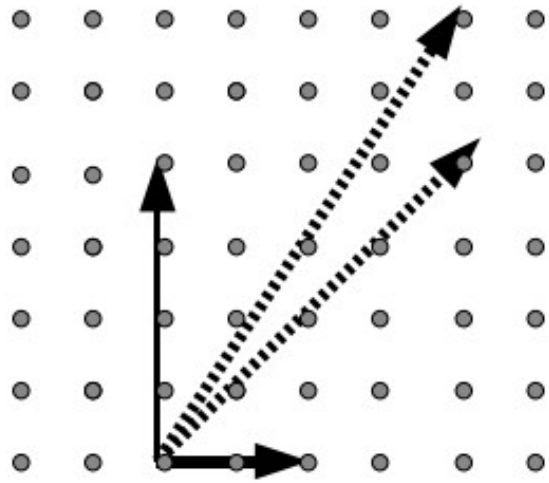
Lattices are like discrete vector spaces

They have bases, so: $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{e}_i$

Post-quantum cryptography

Problems believed to be hard even for quantum computers

Good candidates for crypto: lattice problems



Lattices are like discrete vector spaces

They have bases, so: $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{e}_i$

Shortest vector problem:

Input: a basis to a lattice

Output: shortest vector in the lattice

Food for thought

A quantum computer can do anything a classical computer can do
(at least equally as efficient)

To simulate a classical computer, it can work only in the
computational basis
(without superpositions)

But all operations would still have to be reversible
(because of unitarity)

So can't we just make an efficient circuit for multiplying two
numbers and run it in reverse?

Useful resources and references

- **Pollard's rho algorithm** - http://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm
- **A nice overview of quantum algorithms** - http://www.nature.com/articles/npjqi201523
- **Efficient circuit implementation of Shor's algorithm** - http://arxiv.org/pdf/quant-ph/0205095v3.pdf
- **Scott Aaronson explaining Simon, Shor, RSA and other stuff :)** - http://web.de.mit.edu/public/courses/6/6.045/2015spring/
- **Breaking symmetric key crypto using Simon's algorithm** - http://arxiv.org/abs/1602.05973
- Image on slide 11 (buying an app) - http://cdn2.iconfinder.com/data/icons/shopping-e-commerce-3/512/mobile-finger-512.png
- Image on slide 23 (lattice) - http://courses.cs.washington.edu/courses/cse522/04au/notes/lect18.pdf