

Quantum Computation & Cryptography

Day 3

Quantum algorithms

Andru Gheorghiu

Recap

States

Unit vectors in a
complex vector space

$$|\psi\rangle \in \mathcal{H}$$
$$||\psi\rangle|^2 = 1$$

Transformations

Unitary operations

$$UU^\dagger = U^\dagger U = I$$

Composition

Tensor product

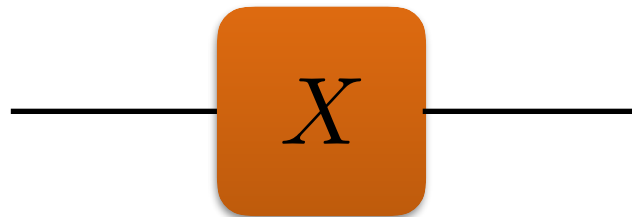
$$\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$$

Observation (measurement)

Orthonormal bases

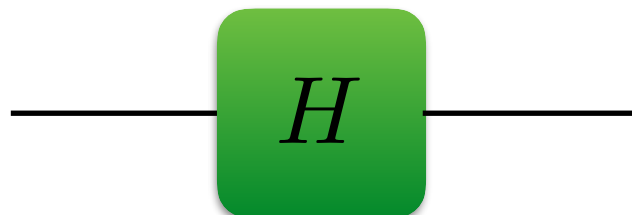
$$\{|x\rangle\}, x \in \{0, 1\}^N$$

Recap



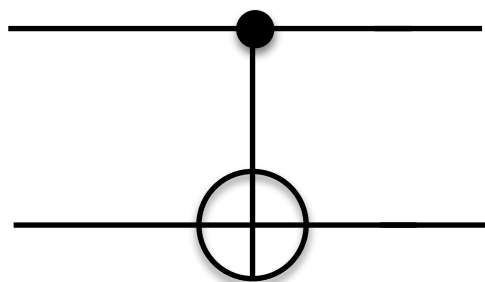
$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle$$

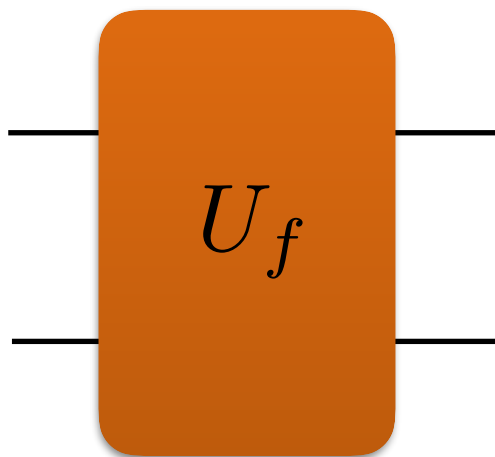


$$H|0\rangle = |+\rangle$$

$$H|1\rangle = |-\rangle$$



$$CNOT|x\rangle|y\rangle = |x\rangle|x \oplus y\rangle$$



$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

for some

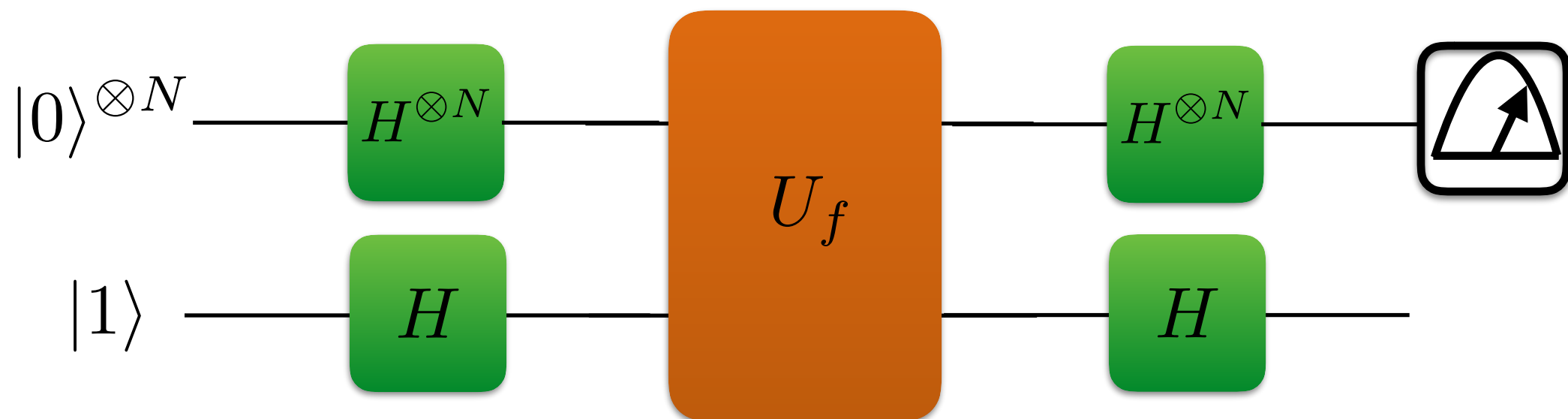
$$f : \{0, 1\}^N \rightarrow \{0, 1\}$$

Recap

Deutsch-Josza problem and algorithm

$$f : \{0, 1\}^N \rightarrow \{0, 1\}$$

Is it **constant** or **balanced**?

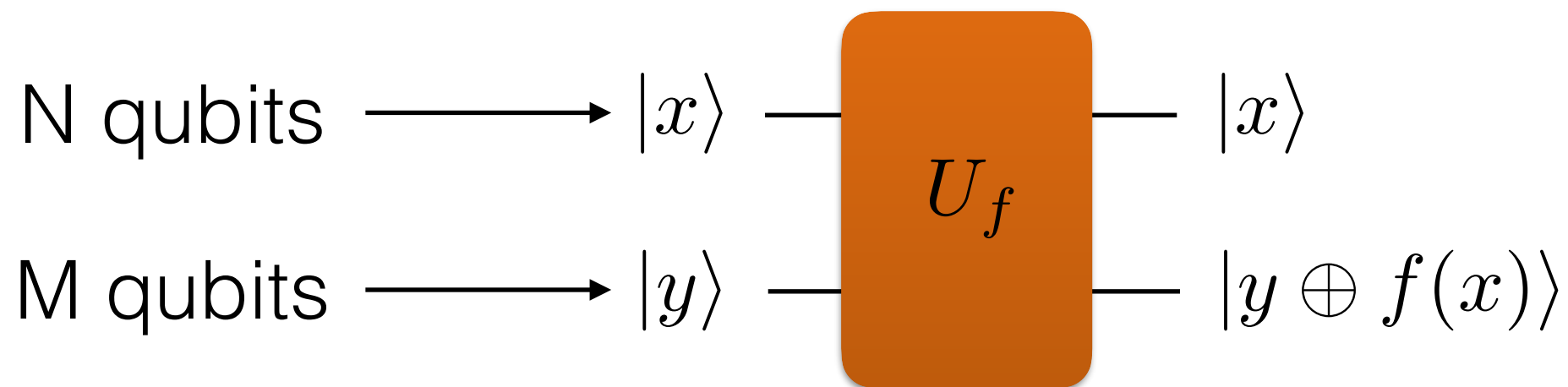


Measure first N qubits

If all zeroes, then function is constant,
otherwise balanced

Generalising the function gate

What about $f : \{0, 1\}^N \rightarrow \{0, 1\}^M$?



$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

We'll only consider functions of the form

$$f : \{0, 1\}^N \rightarrow \{0, 1\}^N$$

Simon's problem

Consider a function of the form

$$f : \{0, 1\}^N \rightarrow \{0, 1\}^N$$

that is promised to be either 1-to-1
or 2-to-1 and periodic

1-to-1

$$\forall x_1, x_2 \in \{0, 1\}^N, f(x_1) = f(x_2) \iff x_1 = x_2$$

2-to-1 with period s

$$\exists s \in \{0, 1\}^N, s \neq 0^N, \quad \forall x_1, x_2 \in \{0, 1\}^N, x_1 \neq x_2,$$

$$f(x_1) = f(x_2) \iff x_1 = x_2 \oplus s$$

Simon's problem

Consider a function of the form

$$f : \{0, 1\}^N \rightarrow \{0, 1\}^N$$

that is promised to be either 1-to-1
or 2-to-1 and periodic

1-to-1

$$\forall x_1, x_2 \in \{0, 1\}^N, f(x_1) = f(x_2) \iff x_1 = x_2$$

Simon function

$$\exists s \in \{0, 1\}^N, s \neq 0^N, \quad \forall x_1, x_2 \in \{0, 1\}^N, x_1 \neq x_2,$$

$$f(x_1) = f(x_2) \iff x_1 = x_2 \oplus s$$

Simon's problem

Consider a function of the form

$$f : \{0, 1\}^N \rightarrow \{0, 1\}^N$$

that is promised to be either 1-to-1
or 2-to-1 and periodic

1-to-1

$$\forall x_1, x_2 \in \{0, 1\}^N, f(x_1) = f(x_2) \iff x_1 = x_2$$

Simon function

$$\exists s \in \{0, 1\}^N, s \neq 0^N, \quad \forall x_1, x_2 \in \{0, 1\}^N, x_1 \neq x_2,$$

$$f(x_1) = f(x_2) \iff x_1 = x_2 \oplus s$$

Determine the function's type!

Simon's problem

Some examples first...

$$\forall x \in \{0, 1\}^N, f(x) = x$$

This is a 1-to-1 function

$$\forall x \in \{0, 1\}^N, g(x) = g(x \oplus 1^N) = \min(x, x \oplus 1^N)$$

This is a Simon function

Take $N = 2$

$$f(00) = 00 \quad f(01) = 01$$

$$f(10) = 10 \quad f(11) = 11$$

$$g(00) = 00 \quad g(01) = 01$$

$$g(10) = 01 \quad g(11) = 00$$

Simon's problem

Some examples first...

$$\forall x \in \{0, 1\}^N, f(x) = x$$

This is a 1-to-1 function

$$\forall x \in \{0, 1\}^N, g(x) = g(x \oplus 1^N) = \min(x, x \oplus 1^N)$$

This is a Simon function

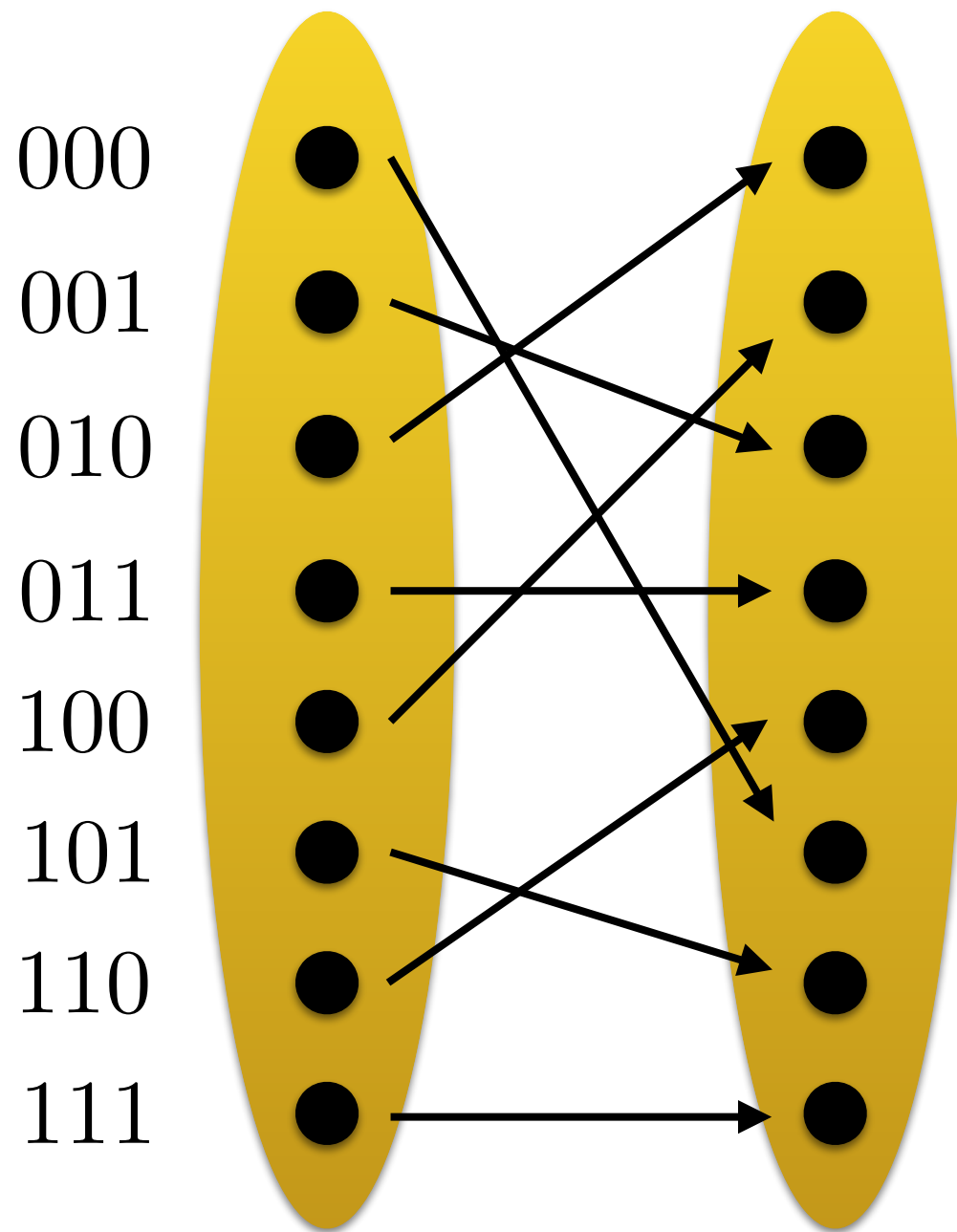
Take $N = 2$

$$\begin{array}{ll} f(00) = 00 & f(01) = 01 \\ f(10) = 10 & f(11) = 11 \end{array}$$

$$\begin{array}{ll} g(\textcolor{red}{00}) = \textcolor{red}{00} & g(\textcolor{blue}{01}) = \textcolor{blue}{01} \\ g(\textcolor{blue}{10}) = \textcolor{blue}{01} & g(\textcolor{red}{11}) = \textcolor{red}{00} \end{array}$$

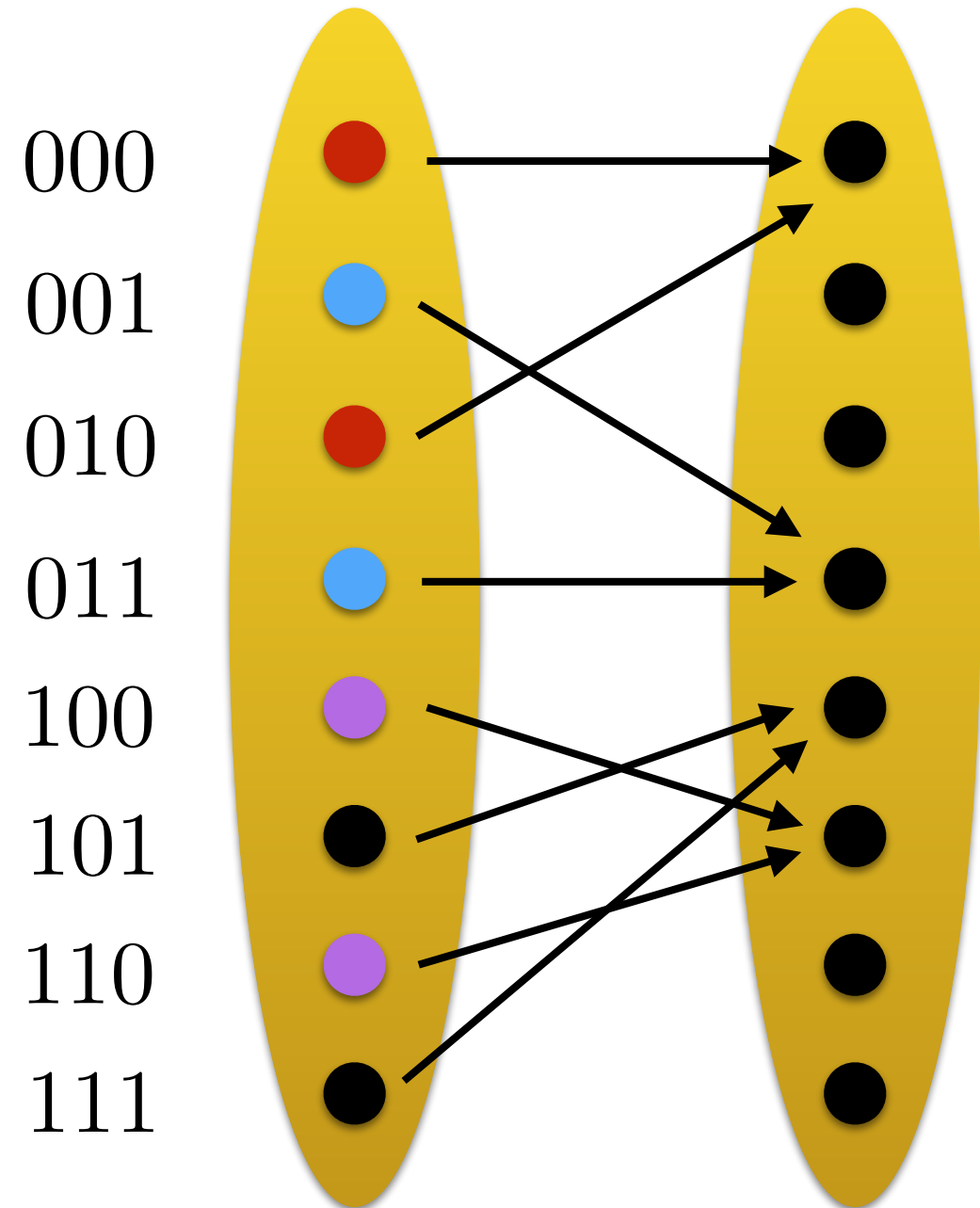
Simon's problem

1-to-1 function



Essentially just a permutation

Simon function



$s = 010$

Simon's problem

We're going to solve this by trying to find a **collision**

$$(x, y) \text{ with } x \neq y, f(x) = f(y)$$

Finding a collision means the function is a Simon function

Otherwise, it's 1-to-1

How many queries classically, deterministically?

$$2^{N-1} + 1$$

Probabilistic vs quantum

Finding collisions

Say we have a 2-to-1 function

How many queries to find a collision
with high probability? (say $> 2/3$)

Let $M = 2^N$

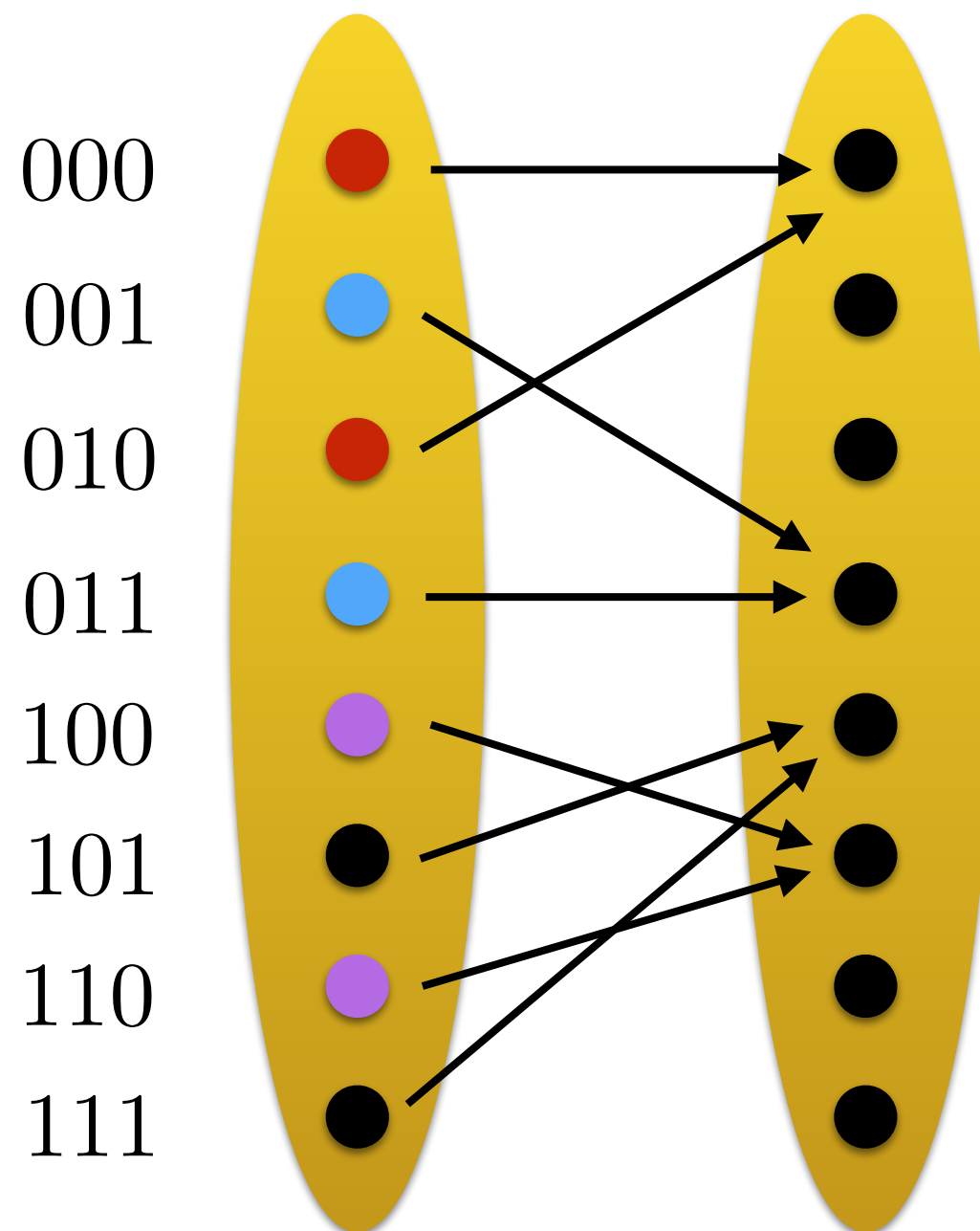
How many collision pairs are there?

Finding collisions

Say we have a 2-to-1 function

$$\text{Let } M = 2^N$$

How many collision pairs are there?



Finding collisions

Say we have a 2-to-1 function

$$\text{Let } M = 2^N$$

How many collision pairs are there?

$$M/2$$

Take k (different) random values from $\{0, 1\}^N$

There will be $\binom{k}{2} = \frac{k(k-1)}{2}$ different pairs

What is the probability of finding a collision?

$$\approx \frac{k(k-1)}{M}$$

Finding collisions

What is the probability of finding a collision?

$$\approx \frac{k(k-1)}{M}$$

We want this to be large

So we take $k = O(\sqrt{M})$

Thus, we will need on the order of $2^{N/2}$ queries

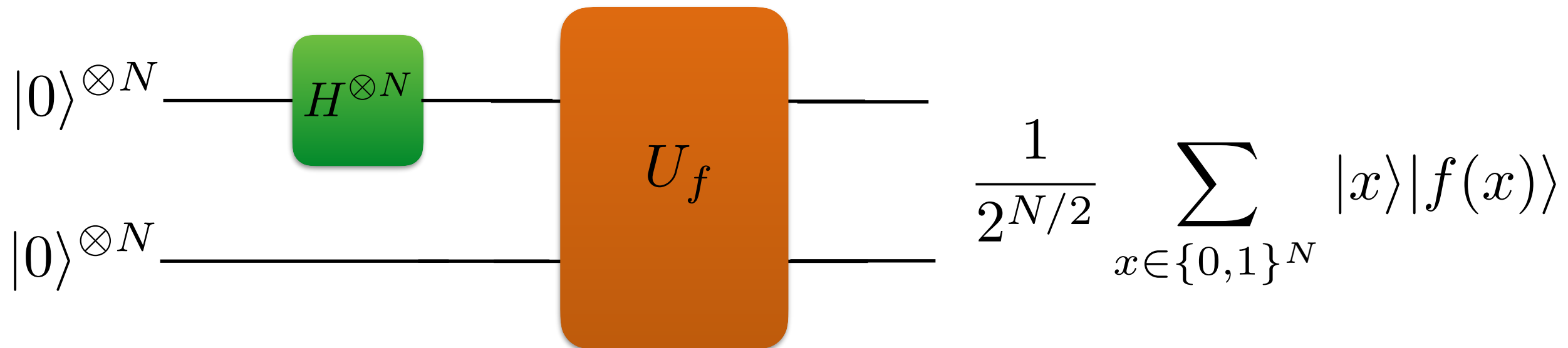
Birthday paradox

Can be shown that this is optimal

Simon's algorithm

What about quantumly?

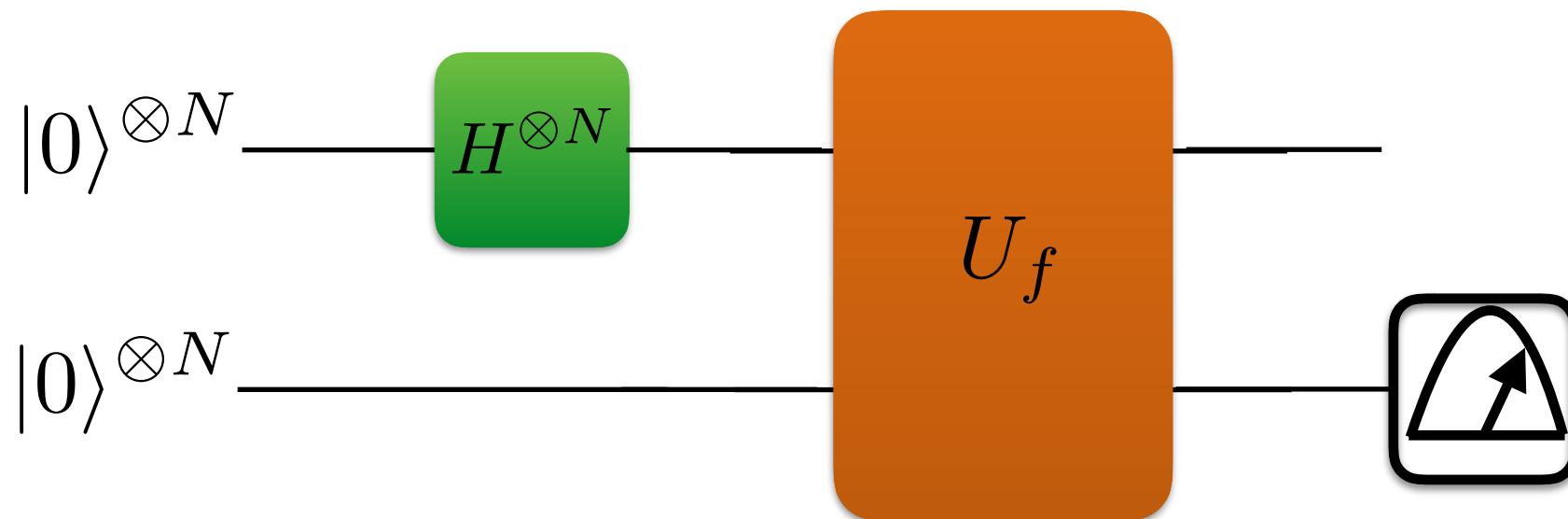
Let's start by creating a superposition of all inputs



Suppose the function is a Simon function

What happens if we measure the second register?

Simon's algorithm



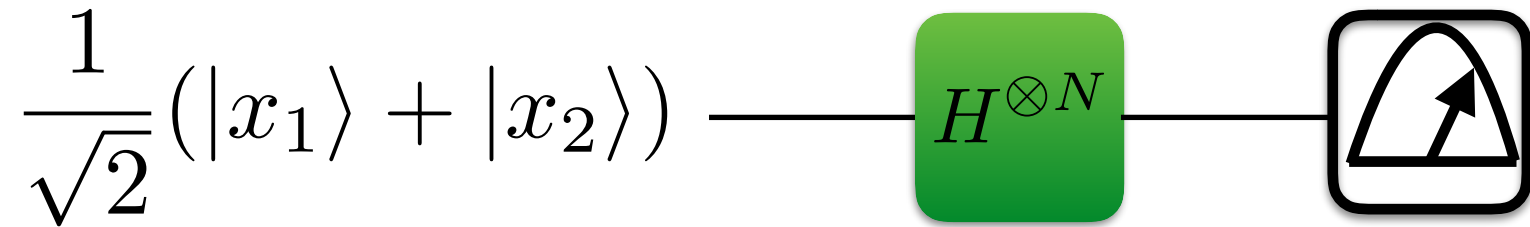
$$\frac{1}{2^{N/2}} \sum_{x \in \{0,1\}^N} |x\rangle |f(x)\rangle$$

Suppose the measurement outcome is $|z\rangle$

What will the first register be?

$$\frac{1}{\sqrt{2}}(|x_1\rangle + |x_2\rangle) \text{ such that } x_1 = x_2 \oplus s, f(x_1) = f(x_2) = z$$

Simon's algorithm



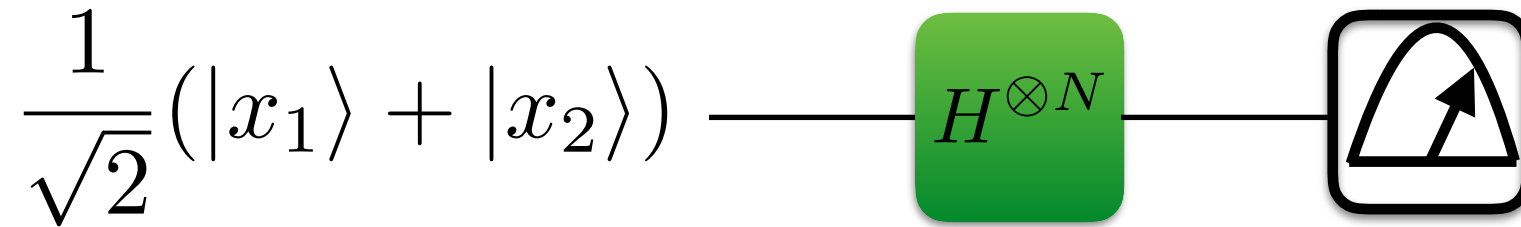
A useful identity

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

Where...

$$x \cdot y = x^1 y^1 \oplus x^2 y^2 \oplus \dots \oplus x^N y^N$$

Simon's algorithm



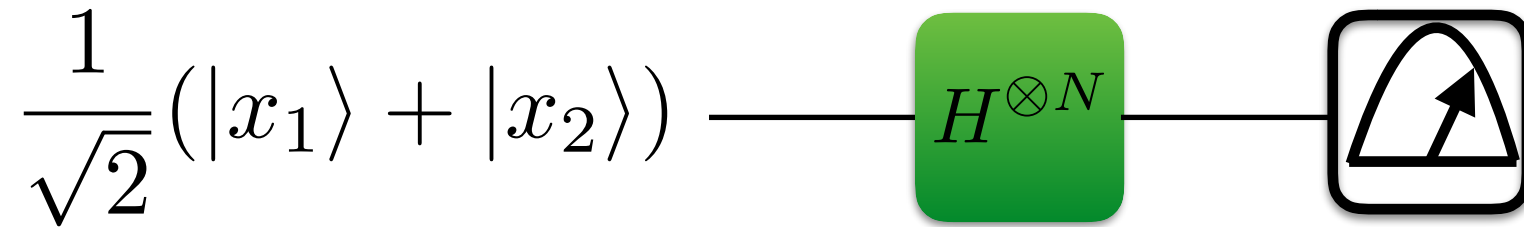
Before measurement, the state will be

$$\frac{1}{2^{(N+1)/2}} \sum_{y \in \{0,1\}^N} ((-1)^{x_1 \cdot y} + (-1)^{x_2 \cdot y}) |y\rangle$$

$$x_2 = x_1 \oplus s$$

$$\frac{1}{2^{(N+1)/2}} \sum_{y \in \{0,1\}^N} ((-1)^{x_1 \cdot y} + (-1)^{x_1 \cdot y \oplus s \cdot y}) |y\rangle$$

Simon's algorithm



$$\frac{1}{2^{(N+1)/2}} \sum_{y \in \{0,1\}^N} ((-1)^{x_1 \cdot y} + (-1)^{x_1 \cdot y \oplus s \cdot y}) |y\rangle$$

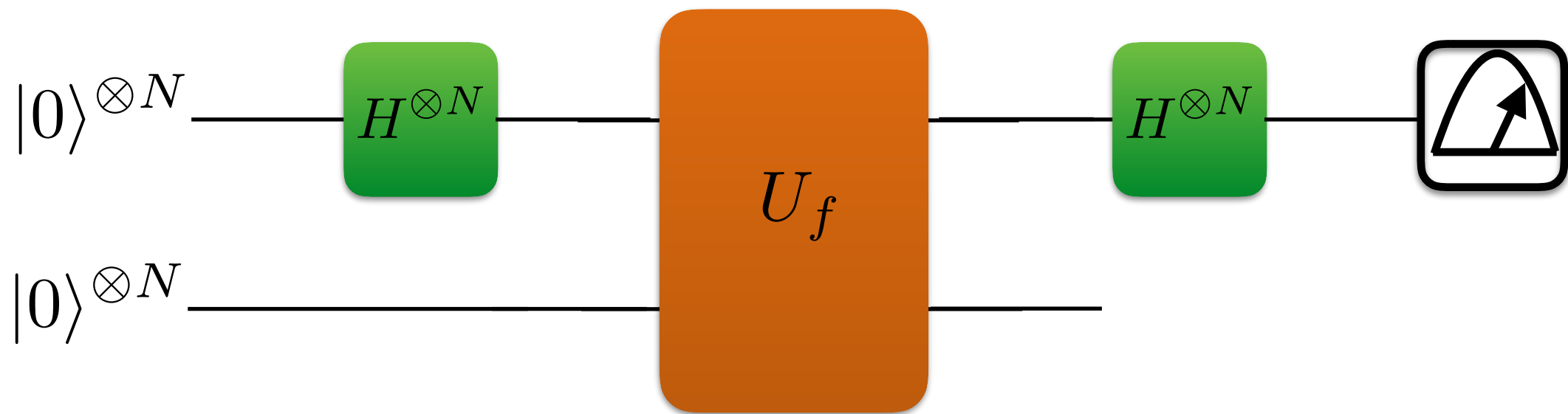
$$\frac{1}{2^{(N+1)/2}} \sum_{y \in \{0,1\}^N} (-1)^{x_1 \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle$$

The only y 's we can get are those for which

$$s \cdot y = 0$$

Simon's algorithm

To recap, for a Simon function...



We obtain y , such that $s \cdot y = 0$

Do it again, and again, and again...

$$y_1, y_2, \dots, y_k$$

$$y_1^1 s^1 \oplus y_1^2 s^2 \oplus \dots \oplus y_1^N s^N = 0$$

$$y_2^1 s^1 \oplus y_2^2 s^2 \oplus \dots \oplus y_2^N s^N = 0$$

...

$$y_k^1 s^1 \oplus y_k^2 s^2 \oplus \dots \oplus y_k^N s^N = 0$$

Simon's algorithm

$$y_1^1 s^1 \oplus y_1^2 s^2 \oplus \dots \oplus y_1^N s^N = 0$$

$$y_2^1 s^1 \oplus y_2^2 s^2 \oplus \dots \oplus y_2^N s^N = 0$$

...

$$y_k^1 s^1 \oplus y_k^2 s^2 \oplus \dots \oplus y_k^N s^N = 0$$

Can be shown that if we do this $O(k)$ times
whp we will get k linearly independent equations

Do it $O(N)$ times!

Solve the system

We've found s , with only $O(N)$ queries!

Simon's algorithm

When we have a Simon function we will
find s whp

When we have a 1-to-1 function, we will
just get random y 's

Hence, we will get a random s

We only need to check whether

$$f(0^N) = f(s)$$

If so, then algorithm says it's a Simon function
otherwise, 1-to-1 function

Simon's algorithm

Once again, the key was interference

$$\frac{1}{2^{(N+1)/2}} \sum_{y \in \{0,1\}^N} (-1)^{x_1 \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle$$

Perfect **destructive** interference when $s \cdot y = 1$

Perfect **constructive** interference when $s \cdot y = 0$

Interference reveals structure about s

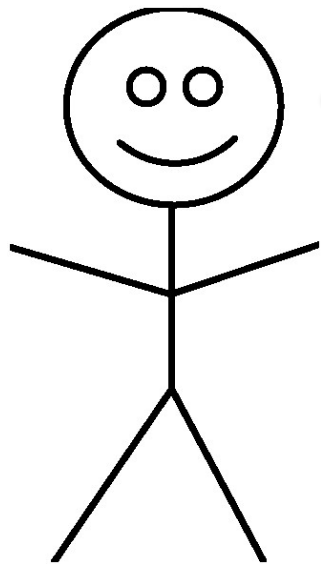
Like finding the period of a diffraction grating

Let's break some classical crypto!

Public-key cryptography

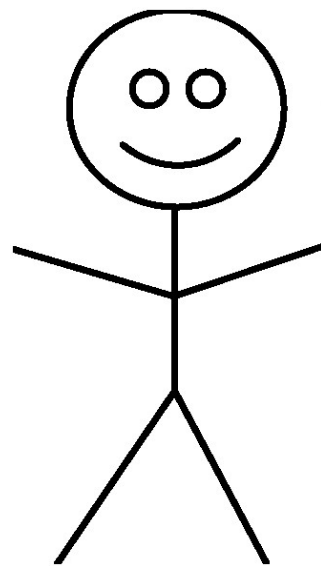
Public-key cryptography

I want to buy



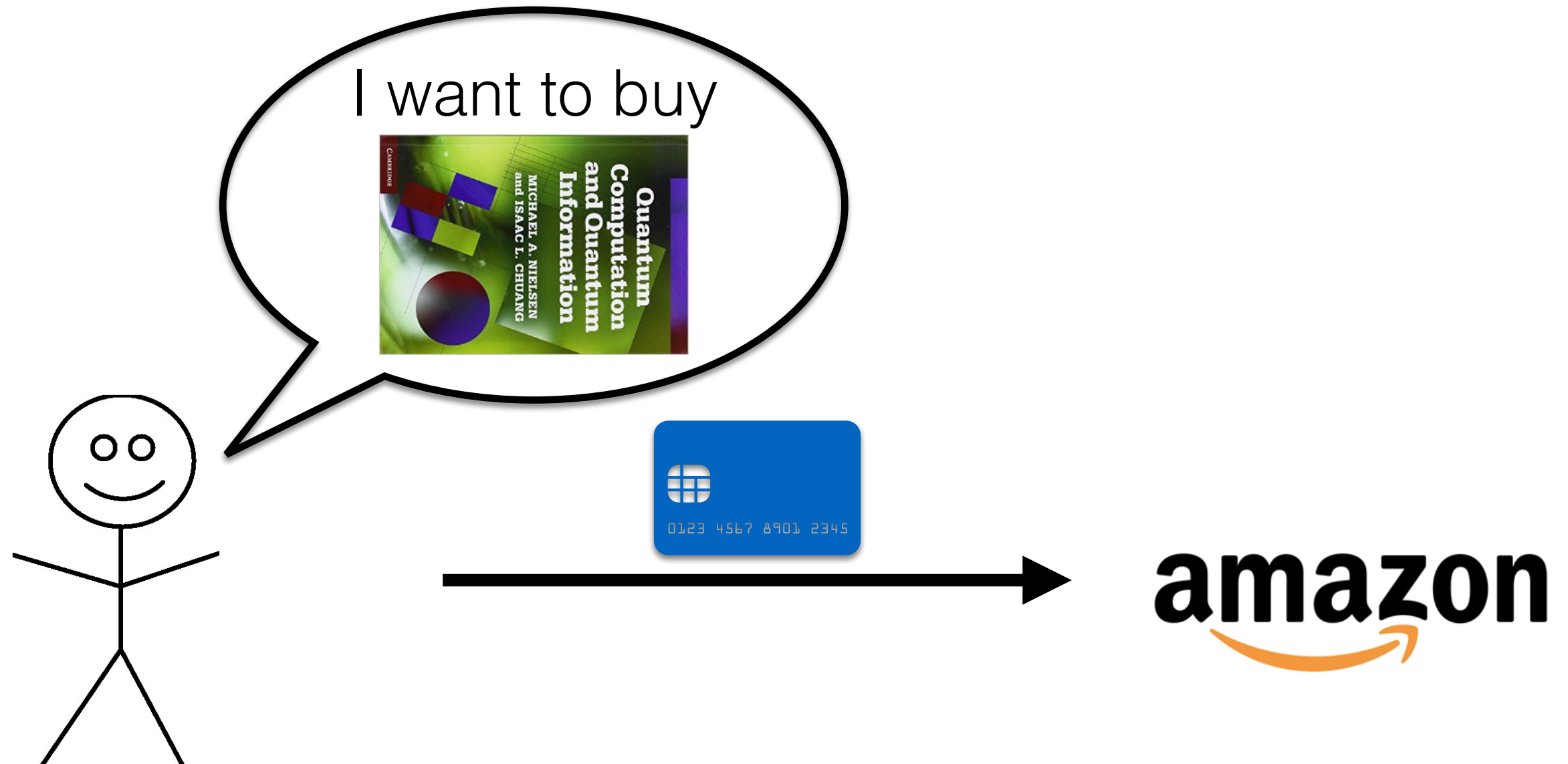
Public-key cryptography

I want to buy



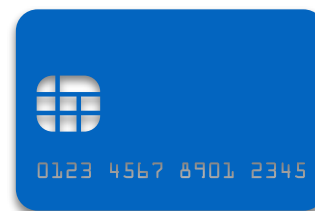
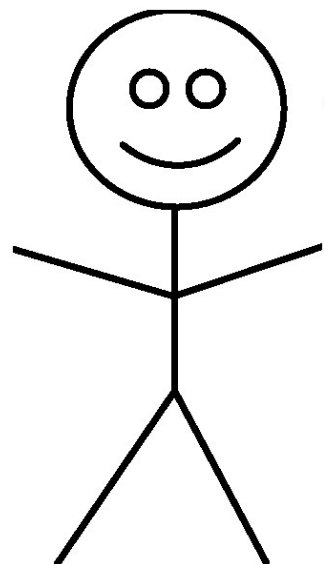
amazon

Public-key cryptography



Public-key cryptography

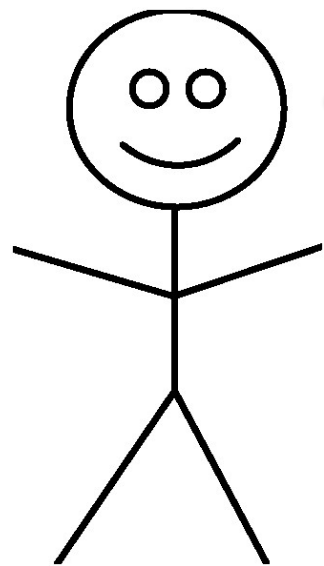
I want to buy



amazon

Public-key cryptography

I want to buy

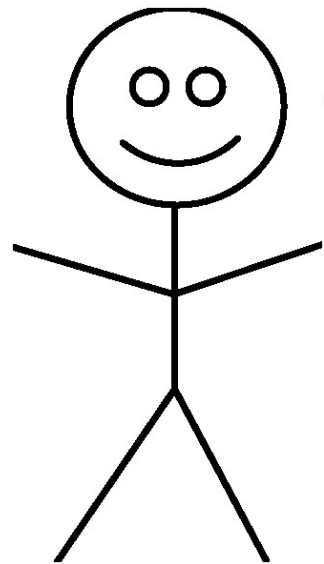


amazon

(PK, SK)

Public-key cryptography

I want to buy



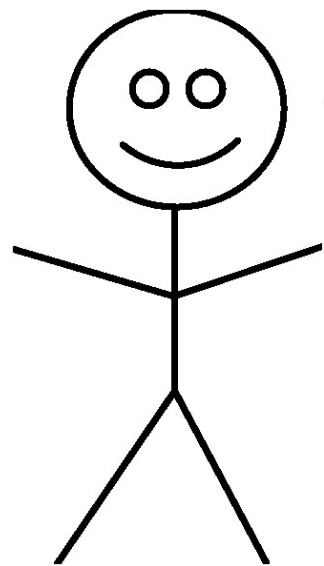
PK

amazon

(PK, SK)

Public-key cryptography

I want to buy



PK

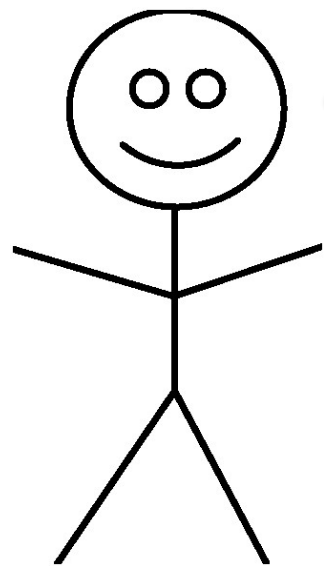
amazon

(PK, SK)

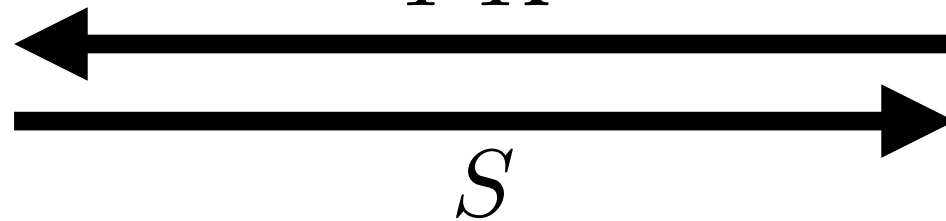
$$S = Enc(PK, \text{Card details})$$

Public-key cryptography

I want to buy



PK



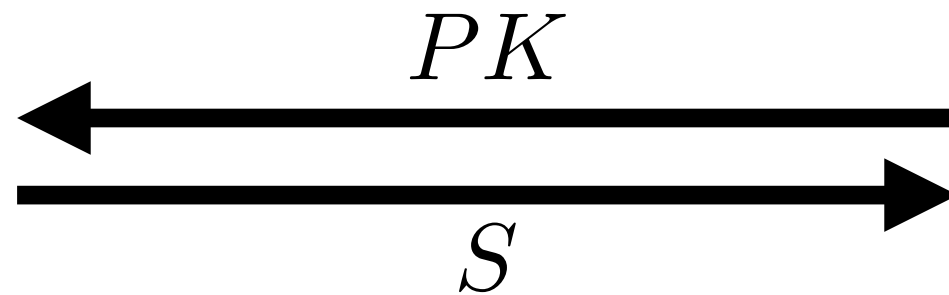
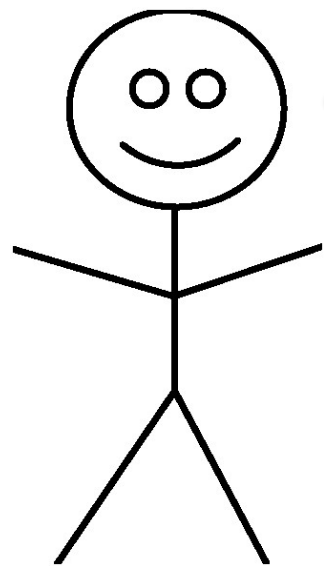
amazon

(PK, SK)

$$S = Enc(PK, \text{Card details})$$

Public-key cryptography

I want to buy



amazon

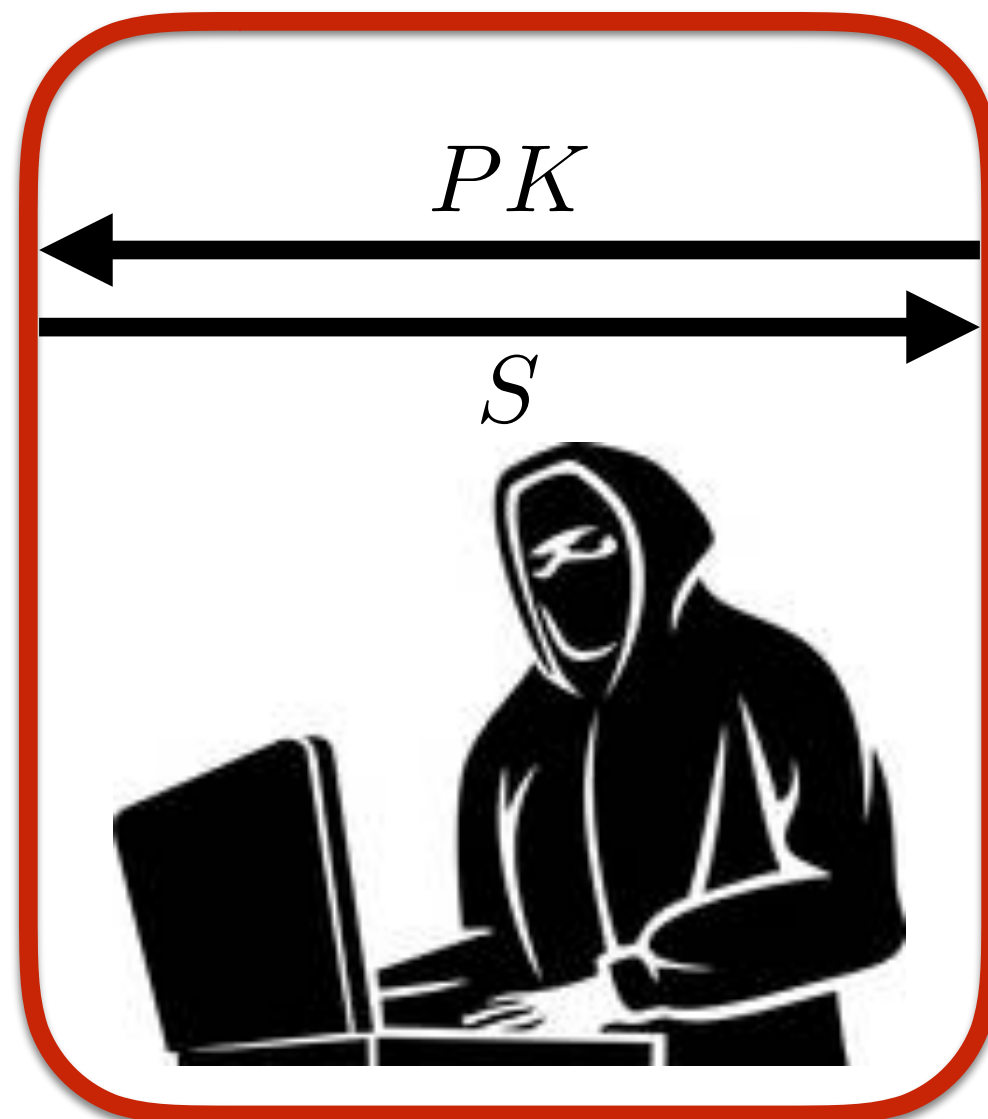
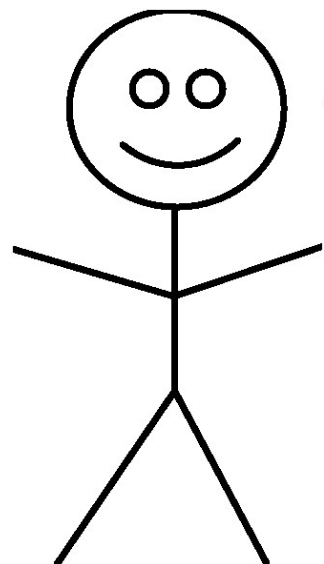
(PK, SK)

$$S = Enc(PK, \text{Card details})$$

$$Dec(SK, S) \rightarrow \text{Card details}$$

Public-key cryptography

I want to buy



amazon
 (PK, SK)

Public-key cryptography

Many public-key crypto systems

RSA, El Gamal, ECC, lattice-based

All of them are based on the computational hardness of certain problems

For the attacker to find SK he would have to solve a “hard problem”

Factoring, computing the discrete logarithm, learning with errors

Factoring

Input: $n = p \cdot q$

p, q prime numbers

Output: (p, q)

“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

Note that the number of bits required to represent our input is

$$N = \log(n)$$

Factoring

Classical approaches

Naive approach: try all factors $O(\sqrt{n}) = O(2^{N/2})$

Pollard's rho algorithm
(uses birthday paradox) $O(2^{N/4})$

General number field sieve $2^{O(\sqrt[3]{N})}$

Sub-exponential time, but still inefficient

No known classical poly-time algorithm

But we have a quantum algorithm, due to Peter Shor

Some group theoretic preliminaries

Integers co-prime with n form a group under multiplication

$$G = \{x \in \mathbb{Z} | x > 0, \gcd(x, n) = 1\}$$

For $n = p \cdot q$ Euler showed that

$$|G| = (p - 1)(q - 1)$$

Knowing the order of the group and n
lets us find p, q

Goal: find the order of G

Some group theoretic preliminaries

Let $g \in G$

Smallest k such that $g^k \bmod n = 1$ is called the **order** of g

Lagrange showed that the order of an element always divides the order of the group

Idea: take a bunch of group elements and compute the lcm of their orders

What will that be, with high probability?

$$|G|$$

A constant number of elements is sufficient

Factoring

We've reduced factoring to finding the order of an element in G

Order finding

Input: $n = p \cdot q, \quad g$

Output: smallest k , s.t. $g^k \bmod n = 1$

Remember that $N = \log(n)$

We want an algorithm that is polynomial in N

Shor's algorithm

Idea and an anecdote

$$f : G \rightarrow G$$

$$f(x) = g^x \bmod n$$

Since $g^k \bmod n = 1$

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

Find k !

Sound familiar?

Shor's algorithm

Consider the state

$$\sum_x |x\rangle |f(x)\rangle$$

Suppose we measure the second register and get $|f(z)\rangle$

What will the first register be?

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

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

Here, we're going to use the
Quantum Fourier Transform (QFT)

Quantum Fourier Transform

Discrete Fourier Transform

$$x_1, x_2, \dots, x_M \xrightarrow{DFT} y_1, y_2, \dots, y_M$$

$$y_j = \frac{1}{\sqrt{M}} \sum_{k=1}^M \omega^{jk} x_k \quad \omega = e^{2\pi i/M}$$

Now imagine the x's are amplitudes

$$QFT \left(\sum_{k=1}^M x_k |k\rangle \right) = \sum_{k=1}^M y_k |k\rangle$$

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

Shor's algorithm

What happens if we apply QFT on the state

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

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

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

If we measure this state, probability of seeing y is

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

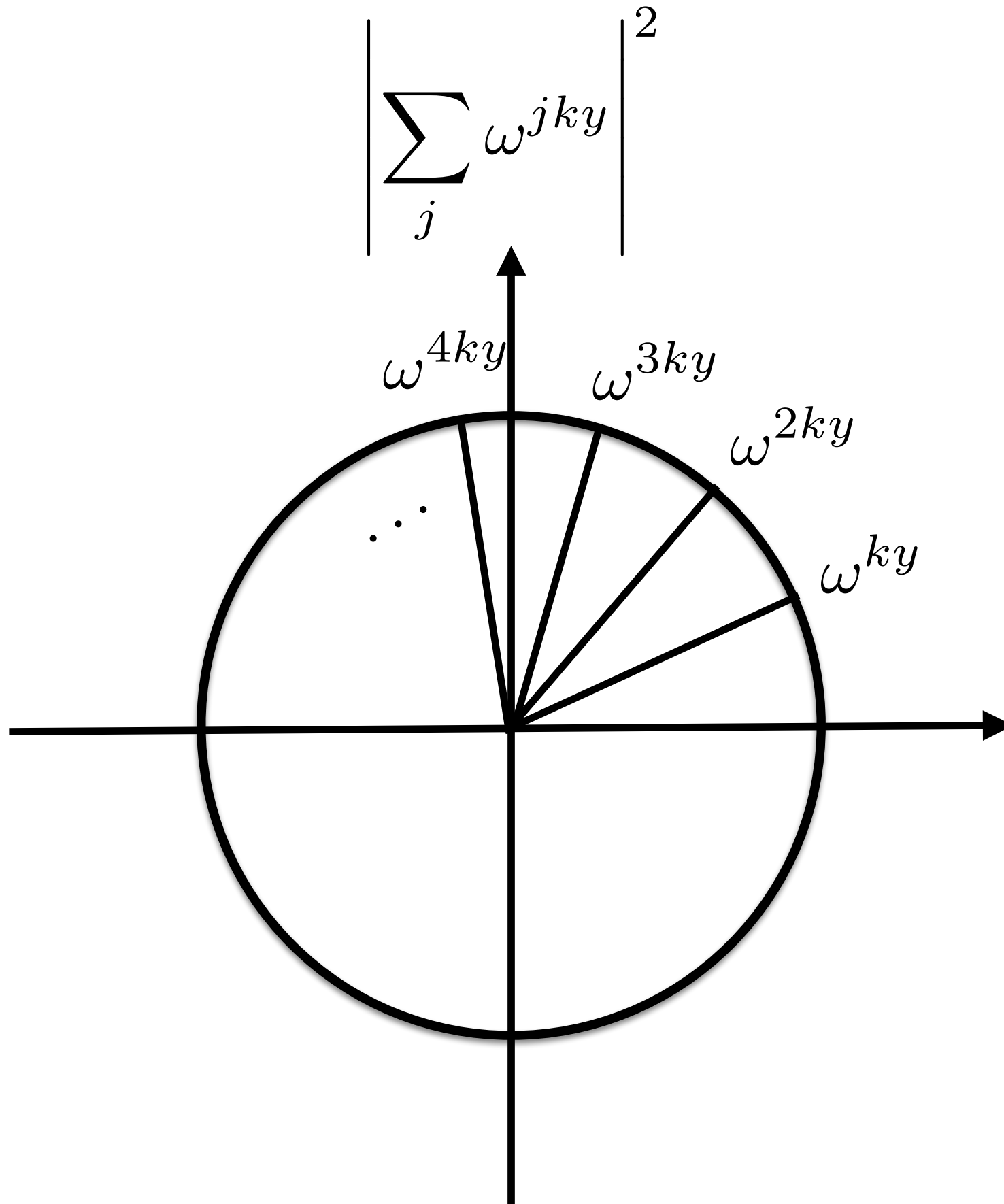
Shor's algorithm

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

When $\omega^{ky} = 1$ the sum is maximal

What about the other cases?

Shor's algorithm



Shor's algorithm

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

When $\omega^{ky} = 1$ the sum is maximal

What about the other cases?

The sum will approach 0
(interference strikes again)

We are likely to see y 's for which $\omega^{ky} = 1$

This means that y will be close to $\frac{m2^N}{k}$

For some integer m

Shor's algorithm

Repeat the following multiple times

Create $\sum_x |x\rangle |f(x)\rangle$

Measure second register

Apply QFT to first register

Measure first register

$$\frac{m_1 2^N}{k}, \frac{m_2 2^N}{k}, \frac{m_3 2^N}{k} \dots$$

Use *modular multiplicative inverse* and *GCD* to find k

Recapping Shor's algorithm

Factoring \leq Order finding

Order finding \equiv Period finding

Do pretty much the same as in Simon's algorithm

Complexity will be $O(N^3)$

Hadamard replaced with QFT

In Simon we were looking for periods over binary strings

In Shor we were looking for periods over integers mod n

Seems like something more general is at play

Hidden Subgroup Problem

Input: $(G, +), f : G \rightarrow G$

G is a group and f is a function such that

There exists a subgroup H of G and...

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

Output: efficient description of H

By efficient we mean a list of generators

We are assuming G was also presented
as a list of generators

Hidden Subgroup Problem

Simon's problem

$$G = \{0, 1\}^N, + = \oplus, H = \{s, 0^N\}$$

Order finding


$$G = \{0, 1, \dots, n-1\}, + = + \bmod n, H = \{x \in G \mid g^x \bmod n = 1\}$$

For some element g for which we want to find the order

Note that this G is not the same group
as the one that g is from

Note that both groups are **abelian**!

Abelian Hidden Subgroup Problem

Generalise Shor  Efficient quantum algorithm for AHSP

No known efficient classical algorithm for AHSP

Computing the discrete logarithm is also an instance of AHSP

This compromises security of
Diffie-Hellman, RSA, El Gamal, ECC and others

What about the non-abelian case?

Non-abelian Hidden Subgroup Problem

Quantum computers can **almost** solve it

[https://people.eecs.berkeley.edu/~vazirani/s09quantum/notes/
lecture11.pdf](https://people.eecs.berkeley.edu/~vazirani/s09quantum/notes/lecture11.pdf)

No one knows how to overcome the “almost”

People have looked at specific groups

Dihedral group, $2^{O(\sqrt{N})}$ algorithm due to Kuperberg

<https://arxiv.org/abs/quant-ph/0302112>

Hope for quantum-secure cryptosystems?

References and resources

Birthday paradox

https://en.wikipedia.org/wiki/Birthday_problem

Pollard's rho algorithm

https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm

Simon & Shor

https://www.youtube.com/watch?v=EQKt_hyk2h8

<https://www.scottaaronson.com/blog/?p=208>

Lecture notes on public-key crypto and quantum algorithms

<http://stellar.mit.edu/S/course/6/sp15/6.045/materials.html>