# Applied Cryptography and Network Security

**William Garrison**
bill@cs.pitt.edu
6311 Sennott Square

Quantum Computing

University of Pittsburgh

# Overview

Today we'll study quantum computing

- Qubits and superposition
- Entanglement
- Quantum gates
- Shor's algorithm

Some notes:

- We're not going to discuss the physics of why quantum mechanics works the way it does
- Nor will we learn how to build a quantum computer
- Instead… we'll take the quantum computing model as truth, and show what algorithms look like in that model
  - And the impact they could have

# What is a quantum computer

We don't need to understand quantum mechanics fully to get an intuition for quantum computers

- What is important is understanding what types of algorithms a quantum computer would be useful for

A conventional computer stores information in bits; a bit is 0 or 1

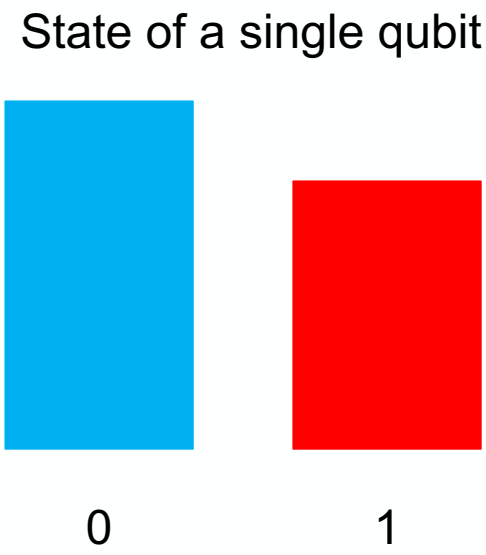
A quantum computer stores information in qubits

So… what's the difference between a qubit and a bit?

# Qubit – "Superposition"

A qubit simultaneously stores 0 and 1 (both values superposed)

The state of the qubit is how much 0 vs. how much 1




*"How much" determines the probability, if you read it, that you'll get 0 vs 1*

# Reading a qubit ("measurement")

This is also weird…

If you read the qubit, you'll get either 0 or 1 (with probability depending on how much of each is superposed)

And then the other value disappears!

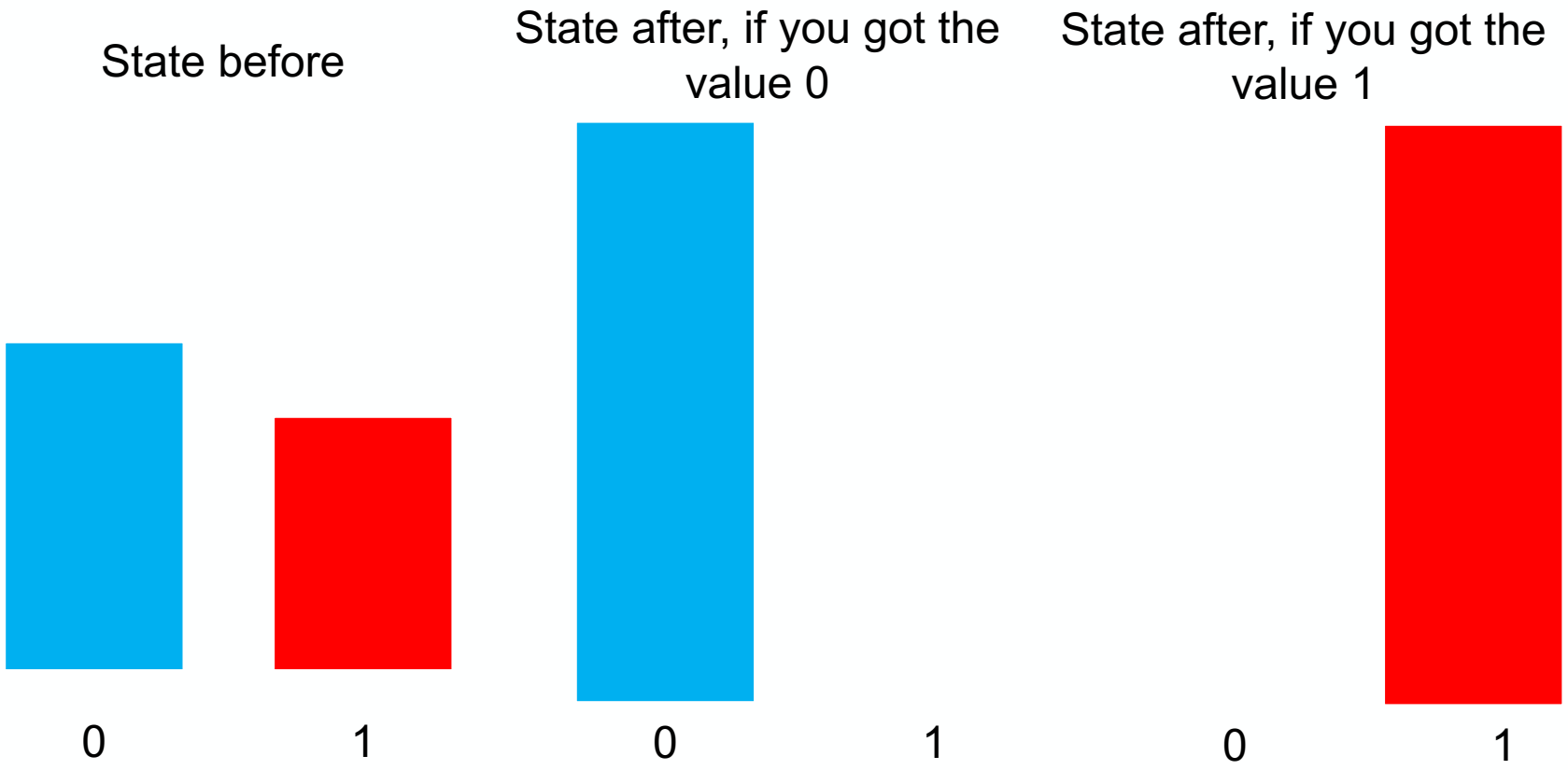
- If you read 0, it becomes completely 0

If you read a qubit twice, you'll always get the same answer the second time, because its state is no longer the superposed quantum state

You can't make a copy of a qubit

- Later, we'll see why

# Reading a qubit… say you read 0



# Entanglement

This is really weird

If a group of qubits is entangled, you can't accurately talk about the state of each one independently

Instead, the group state is a set of superposed states

- If it were 2 qubits, some superposition of 00, 01, 10, 11
- If it were 3 qubits, some superposition of 000, 001, 010, 011, 100, 101, 110, 111

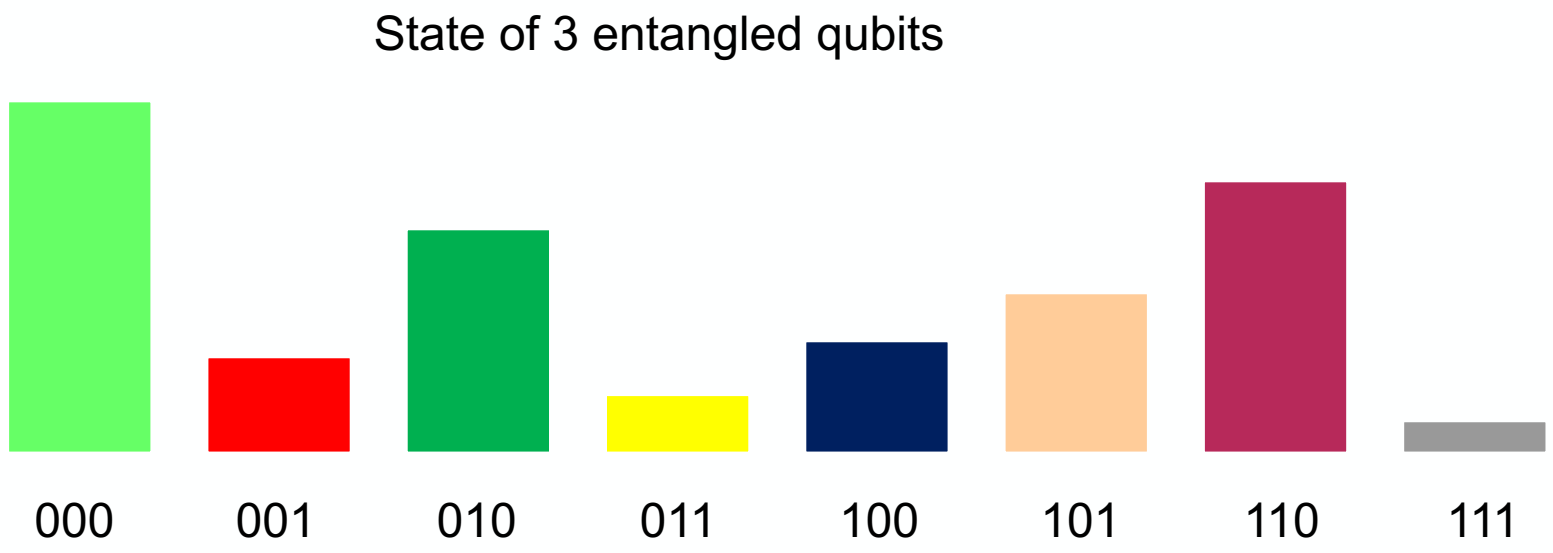
If the state of the entangled qubits is a superposition of only 00 and 11, then if you read one qubit, you'll know what the other one is

- … Even if after entangling them, you move them very far apart

# Groups of qubits

A group of n qubits has $2^n$ superposed states

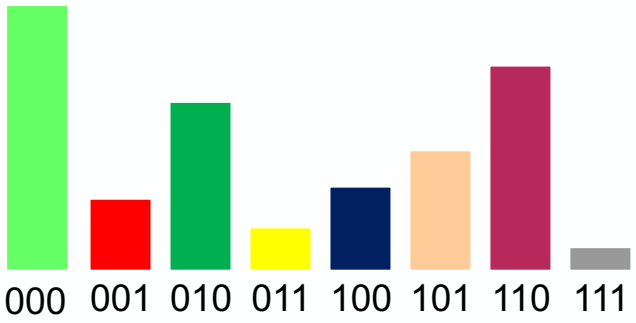
- For instance, 3 qubits (A, B, and C) holds some amount of each of 000, 001, 010, … 111

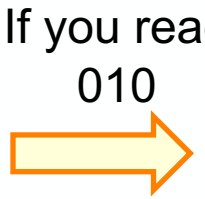


# Group of 3 qubits (A:B:C)

The amount of 000 vs 001 vs 010, etc., determine the probability that if you read the group, which value you'll get

- Once you read it, the other values go away

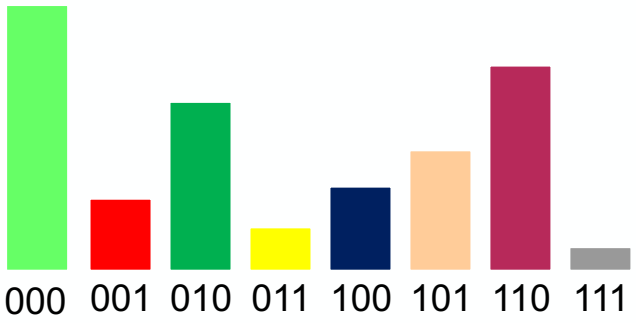


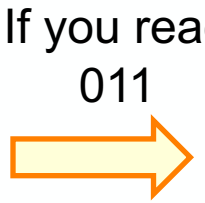



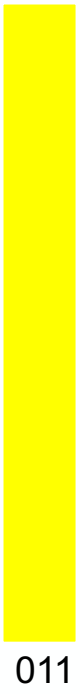

# Group of 3 qubits (A:B:C)

The amount of 000 vs 001 vs 010, etc., determine the probability that if you read the group, which value you'll get

- Once you read it, the other values go away

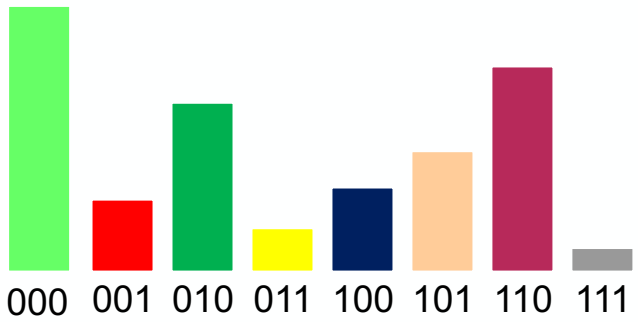


# Group of 3 qubits (A:B:C)

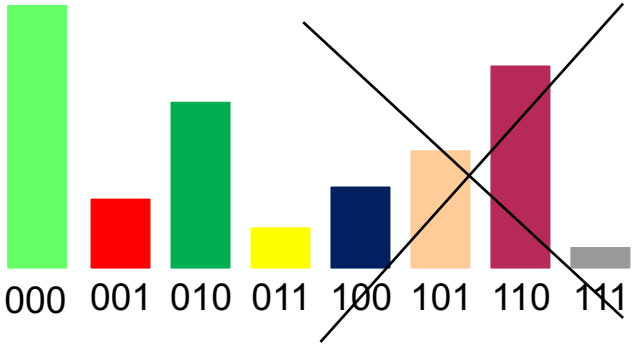
Suppose you only read a subset (say the first qubit), of the entangled qubits, and you read 0



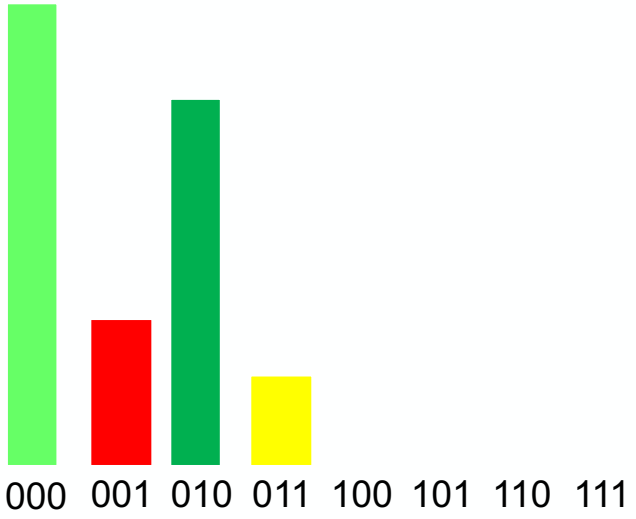

*What do you think will happen?*

# Group of 3 qubits (A:B:C)

Suppose you only read a subset (say the first qubit), of the entangled qubits

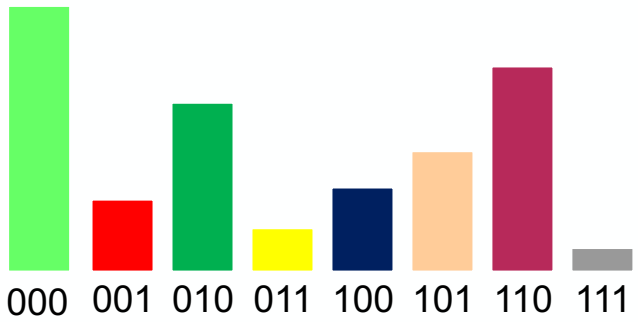



If you read 0



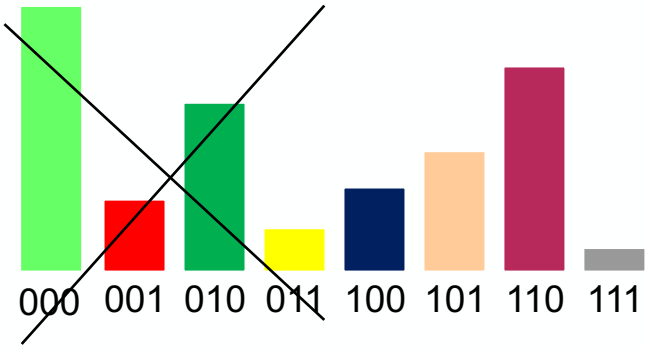
# Group of 3 qubits (A:B:C)

Suppose you only read a subset (say the first qubit), of the entangled qubits, and you read 1

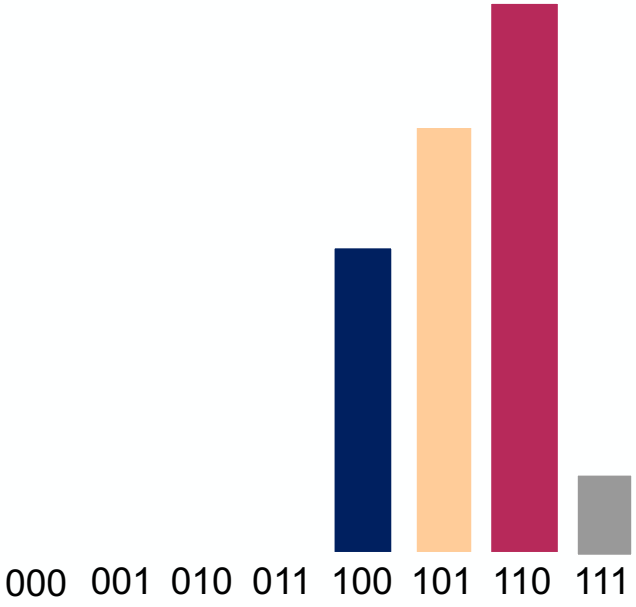


# Group of 3 qubits (A:B:C)

Suppose you only read a subset (say the first qubit), of the entangled qubits




If you read 1



# Superposition makes a quantum computer powerful, but also creates restrictions


When you compute on a set of $n$ qubits, it is logically computing on all $2^n$ superposed values simultaneously

- The output state: Add up the outputs of all the inputs, in proportion

But not as powerful as one might think, since

- You can't read it without losing the superposition (and the superposition is what makes it powerful)
- You can't copy a qubit
- You can't choose the answer you want… it's always a random selection
- The parallelism doesn't give you $2^n$ answers… it gives you one (based on the relative probabilities of the $2^n$ values)

# A quantum computer is not…

… a simple extension of Moore's Law

- i.e., it's not simply a faster conventional computer

… a non-deterministic Turing machine

- What is this?
- Only a concept, but could be simulated (in exponential time) on a conventional computer
- Defines the class NP
- (If it existed) it would operate on all possible inputs simultaneously
  - Sometimes modeled as a conventional computer with infinite parallelism
- A quantum computer sounds similar, but it is not

# How do qubits get into the desired state?


To initialize a qubit to be 0:

- Read it. If 0, mission accomplished!
- If 1, apply a NOT gate

To create superposition, apply a gate like a Hadamard gate

- What is a Hadamard gate? If you start with 0 or 1, after Hadamard, they are equally mixed

Lots of gates are possible, though they tend to only act on 1 or 2 qubits

If you want to do something with a bunch of qubits, you must create it out of a circuit of 1- and 2-qubit gates

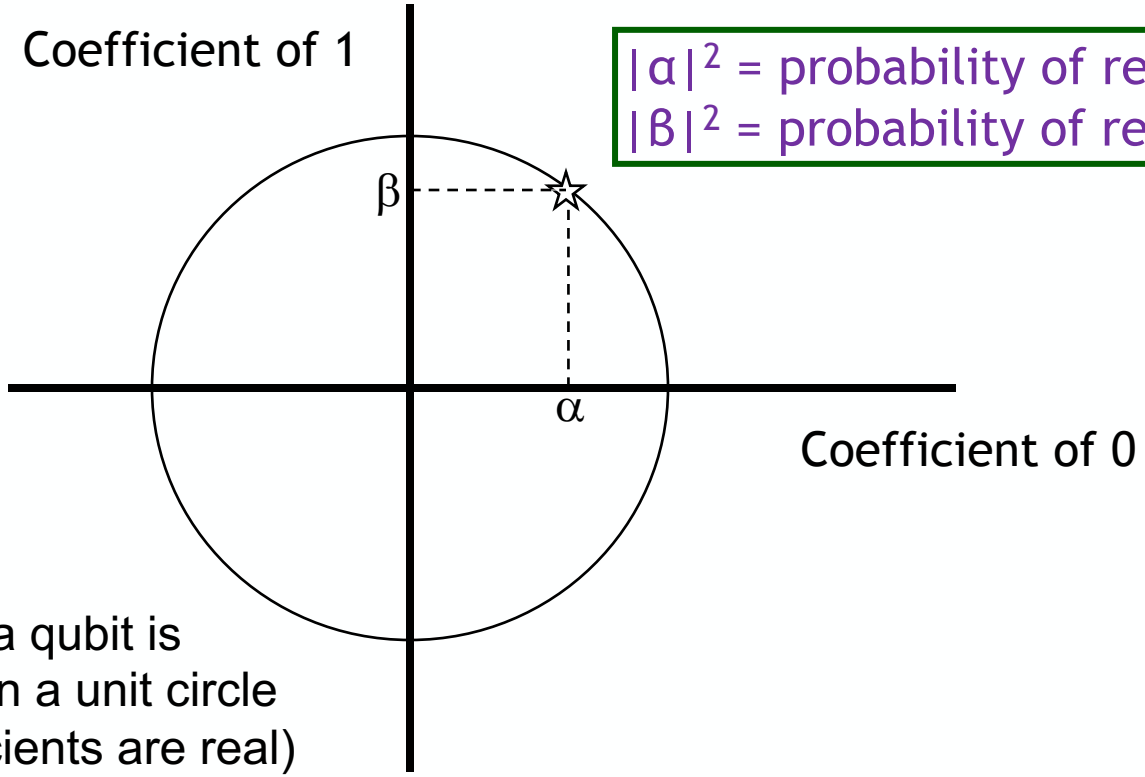
- Universal gate sets:
  - Any classical circuit can be built out of NAND gates (or XOR, AND, and NOT)
  - There are universal gate sets for quantum circuits, but they can only approximate any possible gate

# Notation for qubit states


"Ket" notation (from Dirac) expresses the "amount" of 0 vs. 1 in a qubit's state

- Single qubit: $\alpha|0\rangle + \beta|1\rangle$
- $\alpha$ and $\beta$ are probability amplitudes; $|\alpha|^2$ is the probability of getting 0 when measured
  - Note that $\alpha$ and $\beta$ can be negative or even complex, hence the use of magnitude
- $|\alpha|^2 + |\beta|^2$ must equal 1

# State of a qubit as a point on the unit circle




$|\alpha|^2$ = probability of reading 0
$|\beta|^2$ = probability of reading 1

State of a qubit is
a point on a unit circle
(if coefficients are real)

# Multi-qubit states using ket notation

Suppose you have two qubits with states:

- $\alpha|0\rangle + \beta|1\rangle$ (first qubit)
- $\gamma|0\rangle + \delta|1\rangle$ (second qubit)

The state of these two qubits can be expressed in combination:

- $\alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$
- This represents the probabilities of each of the 4 combinations of states we can observe them in
- This is useful for representing entanglement!

# Entangled qubits and their combined state

A set of entangled qubits has a combined state

- Their states depend on one another
- It is impossible to describe their collective state by describing each individually

Consider 3 entangled qubits

- Their state will be a superposition of:
  000, 001, 010, 011, 100, 101, 110, 111
- $\alpha|000\rangle + \beta|001\rangle + \gamma|010\rangle + \delta|011\rangle + \varepsilon|100\rangle + \zeta|101\rangle + \eta|110\rangle + \theta|111\rangle$
- $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 + |\varepsilon|^2 + |\zeta|^2 + |\eta|^2 + |\theta|^2 = 1$

If $n$ qubits are not entangled, their state can be expressed compactly with $2n$ coefficients

- If they are entangled, it requires $2^n$, one for each combined state

# Entanglement is a big part of what makes quantum mechanics so powerful


An entangled set of $n$ qubits holds a superposition of $2^n$ different values

- All at the same time!

Thus, a quantum computer can compute on all $2^n$ possibilities in parallel

Without entanglement, a quantum computer is no more powerful than a conventional computer!

# A photon (a “particle” of light) is one example of an item that can be a qubit

The polarization of the photon’s wave behavior can be described as a quantum state

- Vertical polarization – up/down – can be represented as 1
- Horizontal polarization – left/right – can be represented as 0
- Angles in between:
  - $\alpha|0\rangle + \beta|1\rangle$

One can measure the state of a photon using a polarizing filter

- If it’s aligned with the photon, the photon will always pass through
- If it’s 90 degrees off, the photon will never pass through
- If it’s an angle in between, the photon will pass through with some probability
  - And after, it will be polarized to align with the filter!

# Polarizers are slightly weird measuring devices


The photon either goes through the filter or disappears

There are other types of qubits and measuring devices that can set the qubit state without eliminating the qubit

Some candidate ways to represent qubits

- Any two-level quantum-mechanical system can be a qubit!
- Spin for an electron
- Charge for an electron
- Spin for a nucleus
- Spin for a trapped ion
- Charge of a quantum dot pair
- …

We won't go any deeper into the physical implementations here

# If we can build reliable qubits, we can use quantum gates to manipulate them


A gate is an abstraction; it describes something we to do a qubit to read and/or change its state

- Gates must be implemented for a specific form of qubit

All quantum gates must obey a few physical rules

- Linearity: If we input a superposition of states, the output must match the weighted sum of the outputs we would get if we input the pure states
- Unitarity: If we input a normalized state (where probabilities add to 1), the output state must also be normalized

These are laws of quantum physics, not invented constraints added to the abstraction

# Linearity: Superposition must be preserved

If we input a superposition of states, the output will match the weighted sum of the outputs we would get if we input the pure states

In other words, if:

- … you know how the gate would operate on state X and state Y…
- … and the input state is a superposition $\alpha|X\rangle + \beta|Y\rangle$…

… then the output of the gate will be α times its output for X plus β times its output for Y

# Linearity Example

"CNOT" (conditional not)

- Invert the 2nd bit if the 1st bit is 1
- Truth table
  - 00 → 00
  - 01 → 01
  - 10 → 11
  - 11 → 10

Suppose the input state of two qubits is

- $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$

Result:

- $\alpha|00\rangle + \beta|01\rangle + \delta|10\rangle + \gamma|11\rangle$

# Linearity is the reason that a qubit cannot be copied

Let's say we want to copy qubit $\alpha|0\rangle + \beta|1\rangle$

We can XOR its value into a qubit initialized to 0…

- Use CNOT

But the result is not two independent qubits with state $\alpha|0\rangle + \beta|1\rangle$

- i.e., we don't get $\alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle$

What we actually get: Two entangled qubits!

- $\alpha|00\rangle + \beta|11\rangle$

# Unitarity: Probabilities must continue to add up to 1

This might feel like an obvious thing to point out, so let's look at a counter-example

Proposed operation: "Zeroize" a qubit

- |0> → |0>; and |1> → |0>
- Any reason we can't build a gate to do this?

Start from state 0.6 |0> + 0.8 |1>

- This is normalized, $0.6^2 + 0.8^2 = 0.36 + 0.64 = 1$
- If we apply the hypothetical zeroize, what do we get?
- 0.6 |0> + 0.8 |0> = 1.4 |0>
  - $1.4^2 = 1.96 > 1$ !!!

Thus, zeroize is not a gate that could be built

# Surely we can't build a quantum computer without a way to set something to 0?!


There are ways to achieve nonunitary results…

- … but they necessarily break superposition

To implement a form of zeroize, we can:

- Measure the qubit
- If it's 0, do nothing else
- If it's 1, apply a NOT gate

But note that the act of measurement will collapse the state to the pure observed state!

- There is no longer superposition, since it was measured

# Why is unitarity important?

One consequence of unitarity is that all operations must be reversible

- Given the output, it is possible to perfectly reconstruct the input
- There must be a one-to-one correspondence between each state and its successor
- Note: This doesn't describe the physical implementation that would be required to implement the reverse gate

Unitarity guides our circuit design

- If you want to do something non-reversible, you need to use measurement and collapse superposition…
- … or find an equivalent reversible operation, perhaps using ancilla (extra) qubits
  - Ancillas aren't meaningful to the result, but allow reversal

# An example algorithm built of qubits and quantum gates: Shor's algorithm

Shor's algorithm can factor integers

- It can also be adapted to solve discrete log

Why is this a concern for cryptography?

- ?

Consider: $a^x \bmod n$ is a periodic function

- If you compute $a^x \bmod n$ for increasing values of $x$, you'll find repeated results at regular intervals (say, $D$)
- If we know $D$, we can find a nontrivial square root of 1 mod $n$
  - i.e., a value $y$ such that $y^2 \equiv 1 \pmod{n}$, where $y$ is not 1 or -1
- If we know a nontrivial square root of 1 mod $n$, we can factor $n$

# From periodicity to a nontrivial square root

For any exponent $x$, $a^x \bmod n = a^{x+kD} \bmod n$ for any $k$

If we know $D$, we can find a square root of 1 mod $n$

- $1 = a^0 \bmod n = a^D \bmod n$
- Thus, $a^{D/2} \bmod n$ is a square root of 1!
- But is it nontrivial? What if it's also 1?
  - If $a^{D/2} \bmod n$ is also 1, then $a^{D/4} \bmod n$ is also a square root of 1
  - … and so on, until we find a nontrivial square root
- One corner case: If $a^{D/2} \bmod n = -1$
  - In this case, we're stuck. Choose another $a$ and try again.
  - About half of $a$ values will result in a nontrivial square root

# From a non-trivial square root to factoring n=pq

Assume than $n$ is an RSA modulus, so $n = pq$

If $y^2 \bmod n = 1$, then $(y^2 - 1) \bmod n = 0$

- $(y+1)(y-1) = 0 \bmod n$
- $(y+1)(y-1) = k * n$

This leaves two possibilities:

1. One of $y+1$ and $y-1$ is $0 \bmod n$
2. One is a multiple of $p$, and the other a multiple of $q$

Since $y$ is a nontrivial square root, the first case is impossible

- Thus, $\gcd(y+1, n)$ or $\gcd(y-1, n)$ will yield $p$ or $q$!

Main point: If we can find the period $D$, we can factor $n$

# Shor's algorithm, or how to factor a k-bit RSA modulus by finding its period

Initialize $2k$ qubits for the exponent $x$

- For 2048-bit RSA, this is 4096 qubits
- Call these the exponent qubits

Initialize $k$ more qubits for the value $a^x \bmod n$

- 2048 bits in our example
- Call these the answer qubits

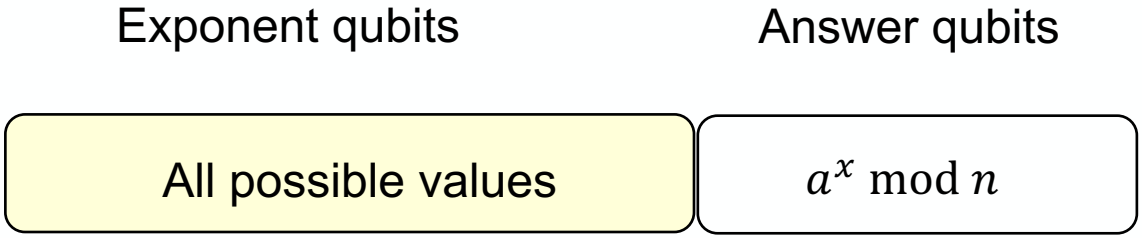
Initialize all $3k$ qubits to 0

Apply Hadamard gates to the $2k$ exponent qubits

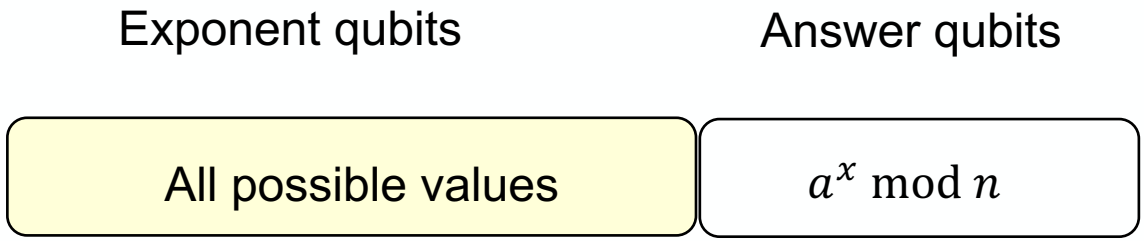
Now, the exponent qubits are superposed to all $2^{2k}$ possible values

Use a quantum circuit to calculate $a^x \bmod n$ and XOR the result into the answer qubits, entangling all $3k$ qubits

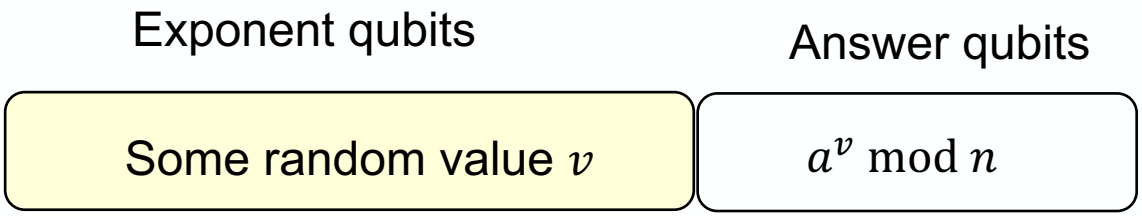
# What is the state after these steps?



# If you read all 3k qubits at this point…




… what would you get?




*Not at all useful… so we're not going to do that!*

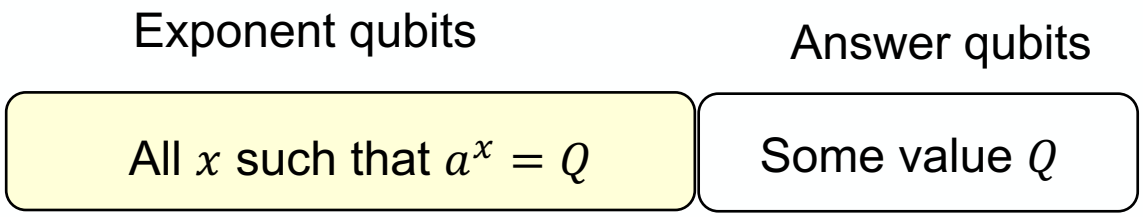
# So… what do we do instead?

Instead, only read the answer qubits

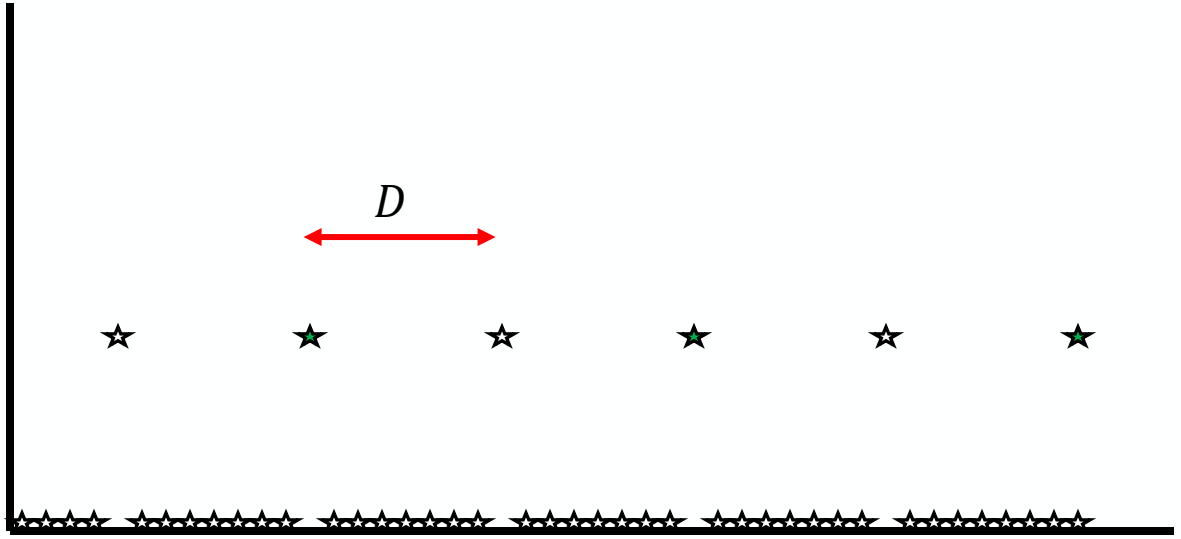
In fact, we don't even care what the value is!

- But, say it's $Q$
- How will this affect the exponent qubits?

The (entangled) exponent qubits will now only have values of $x$ for which $a^x \bmod n = Q$



# Amplitude spikes in exponent qubits are equally spaced




*If you could simply read the spikes, you could derive D…*

*… but you can only measure once, yielding an unhelpful value*

# So how can we use this to determine $D$?

Let's say that $N$ is the total number of superposed values in the exponent qubits

- $N = 2^{2k}$, or $2^{4096}$ for 2048-bit RSA

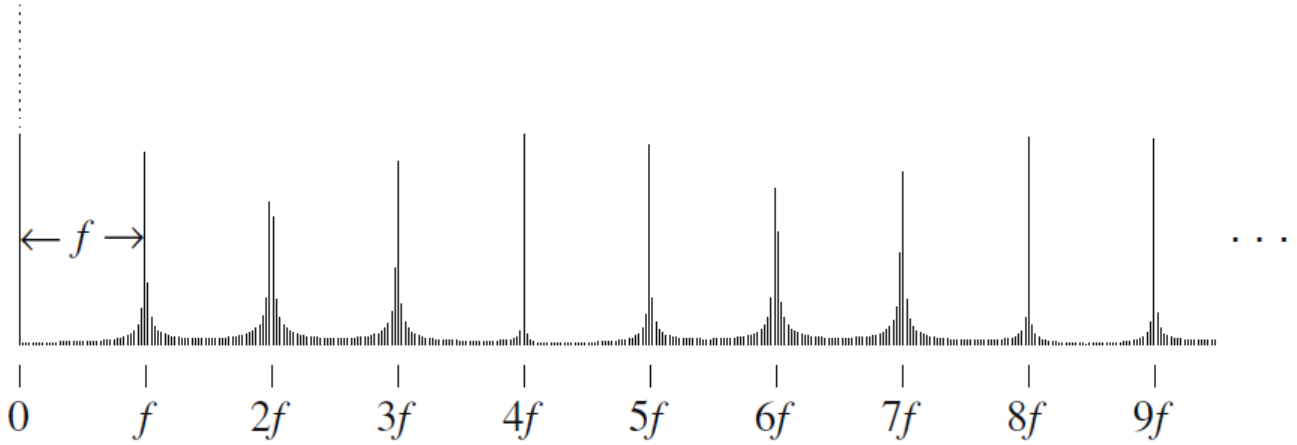
Shor, inspired by discrete Fourier Transforms, noted that there was a way to convert the state to a more useful form

- Rather than the first spike being at an unknown location, align it to 0
- The spikes now occur at every $N/D$, starting with 0

There are a lot of details to this step that we won't discuss in detail

- The previous state is called the "time graph", and the transformed result the "frequency graph"
- The text describes an intuitive way to do this transformation
  - … but as the authors admit, it's not how it would be done by a quantum computer
- Note that D will not exactly divide N, so spikes will be "fuzzy"

# The "frequency graph", where $f = N/D$




Now when we read the exponent qubits, with high probability we'll get a value (very close to) a multiple of $f$

- Say the value we read is $v$
- We next expand $N/v$ as a continued fraction to get rational approximations
- The numerator of one of the approximations is (usually) $D$
- Using $2k$ exponent qubits was chosen to make it very likely that our $v$ is close enough to an integer multiple of $f$

*We've already shown that we can factor $n$ once we know $D$*

# The impact of Grover's algorithm

The textbook also covers Grover's algorithm

- Here, we will summarize the main points

Grover's algorithm improves brute-force search by a square-root factor compared to conventional limits

- Requires about $2^{n/2}$ steps rather than $2^n$ for an $n$-bit value
- This is the main issue quantum computing presents for symmetric-key cryptography and hashing

In this case, we have some good news

- If this becomes feasible, we can compensate by doubling key sizes and hash outputs
- Compared to conventional approaches, Grover's algorithm is much less parallelizable
  - And researchers have shown this is an impossibility, not an issue we can overcome

# Conclusions

Quantum computing has the potential to be very impactful

- … but it is often overblown and misunderstood

We now have a (somewhat surface-level) understanding of:

- Qubits and superposition
- Entanglement
- How quantum computers exploit quantum effects
- How quantum algorithms can break conventional public-key cryptography

Research is active in ways to utilize quantum computing as well as preserve strong cryptography when QC is realized

Next time: Post-quantum cryptography