Quantum computers and the future of cryptography

Katherine E. Stange

Harvard University, Open Neighbourhood Seminar, October 9th, 2019

Classical and Quantum Computers

Classical Computers

- bit: boolean value
- gate: truth table
- space complexity: number of bits
- time complexity: number of sequential gates

Quantum Computers

- qubit: ?
- n entangled qubits: ?
- gate: ?
- space complexity: ?
- time complixity: ?

One qubit

A qubit is a complex linear combination of the two classical states $|0\rangle$ and $|1\rangle$:

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

of length one:

$$a, b \in \mathbb{C}, \quad |a|^2 + |b|^2 = 1.$$

One qubit, measured

The state of a qubit is a complex linear combination of the two classical states $|0\rangle$ and $|1\rangle$:

$$a|0\rangle + b|1\rangle, \quad a, b \in \mathbb{C}, \quad |a|^2 + |b|^2 = 1.$$

We cannot know the state of the qubit. The one thing we can do is measure it:

- with probability $|a|^2$, it will become $|0\rangle$ and tell you "I'm 0!".
- with probability $|b|^2$, it will become $|1\rangle$ and tell you "I'm 1!".

Measuring destroys the state (it *collapses*) and replaces it with the answer, either $|0\rangle$ or $|1\rangle$.

You can never measure again.

Example

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

has equal probility of being measured as 0 or 1.

Two entangled qubits

The state of two (possibly entangled) qubits is a complex linear combination of all four possible classical states:

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle,$$

of length one:

$$a, b, c, d \in \mathbb{C}, \quad |a|^2 + |b|^2 + |c|^2 + |d|^2 = 1.$$

If we measure it, we obtain

- $|00\rangle$ with probability $|a|^2$
- $|01\rangle$ with probability $|b|^2$
- $|10\rangle$ with probability $|c|^2$
- $|11\rangle$ with probability $|d|^2$

Example

$$\frac{1}{\sqrt{4}}|00\rangle + \frac{1}{\sqrt{4}}|01\rangle + \frac{1}{\sqrt{4}}|10\rangle + \frac{1}{\sqrt{4}}|11\rangle$$

has equal probility of being measured as 00, 01, 10 or 11.

Example

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

has equal probility of being measured as 00, 11. It is not possible to measure 01 or 10.

Measuring one bit at a time

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$$

It's also possible to measure just the second bit. In this case, we obtain

- 0 with probability $|a|^2 + |c|^2$ and the state collapses to $ak|00\rangle + ck|10\rangle$, where k is chosen to make $|ak|^2 + |ck|^2 = 1$
- 1 with probability $|b|^2 + |d|^2$ and the state collapses to $bk|01\rangle + dk|11\rangle$, where k is chosen to make $|bk|^2 + |dk|^2 = 1$

Example

$$\frac{1}{\sqrt{4}}|00\rangle + \frac{1}{\sqrt{4}}|01\rangle + \frac{1}{\sqrt{4}}|10\rangle + \frac{1}{\sqrt{4}}|11\rangle$$

If we measure the second qubit, we get

- 0 with probability $1/2$ and the state collapses to $1/\sqrt{2}|00\rangle + 1/\sqrt{2}|10\rangle$
- 1 with probability $1/2$ and the state collapses to $1/\sqrt{2}|01\rangle + 1/\sqrt{2}|11\rangle$

Suppose we are in the first case. Then if we continue on to measure the first qubit (after the second), we get 0 or 1 with equal probability.

Example

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

If we measure the second qubit, we get

- 0 with probability 1/2 and the state collapses to $|00\rangle$
- 1 with probability 1/2 and the state collapses to $|11\rangle$

Suppose we are in the first case. Then if measure the first qubit, we get 0 for sure.

This is weird, because if we'd measured it first, we would have gotten 0 or 1 with equal probability.

So measuring the second qubit forces the first to collapse, in some sense. This is *entanglement*.

Classical and Quantum Computers

Classical Computers

- bit: boolean value
- gate: truth table
- space complexity: number of bits
- time complexity: number of sequential gates

Quantum Computers

- qubit: unit vector in $\mathbb{C}^2$, i.e. a complex linear combination of the two states 0 and 1
- n entangled qubits: a linear combination of all 2^n possible classical bit states
- gate: ?
- space complexity: ?
- time complixity: ?

Quantum Gate

A *quantum gate* is a unitary matrix acting on the vector of coefficients of the state.

Quantum Gate

A *quantum gate* is a unitary matrix acting on the vector of coefficients of the state.
(*Unitary* just means it preserves the length of vectors.)

Example

The Hadamard gate is

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

It acts like this:

$$H(a|0\rangle + b|1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} a+b \\ a-b \end{pmatrix} = \frac{a+b}{\sqrt{2}}|0\rangle + \frac{a-b}{\sqrt{2}}|1\rangle.$$

For example, it takes $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ to $|0\rangle$.

Gates

A 2×2 quantum gate can act on a one-qubit state.
A 4×4 quantum gate can act on a two-qubit state.
An 8×8 quantum gate can act on a three-qubit state.
etc.

Example

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This has the following effect:

$$|00\rangle \mapsto |00\rangle, \quad |01\rangle \mapsto |01\rangle, \quad |10\rangle \mapsto |11\rangle, \quad |11\rangle \mapsto |10\rangle.$$

One can think about the quantum gate via its action on basic states.
It is possible to perform standard arithmetic operations and normal logic on qubits, e.g.

$$|x, 0\rangle \mapsto |x, f(x)\rangle,$$

for functions $f(x)$ you might compute on a classical computer.

Classical and Quantum Computers

Classical Computers

- bit: boolean value
- gate: truth table
- space complexity: number of bits
- time complexity: number of sequential gates

Quantum Computers

- qubit: unit vector in $\mathbb{C}^2$, i.e. a complex linear combination of the two states 0 and 1
- n entangled qubits: a linear combination of all 2^n possible classical bit states
- gate: unitary matrix
- space complexity: number of qubits
- time complixity: number of gates

Classical vs. Quantum Computation

Factoring an integer n:

- Classical: sub-exponential: $O\left(e^{C(\log n)^{1/3}(\log\log n)^{2/3}}\right)$ gates (General number field sieve)
- Quantum: polynomial: $O\left((\log n)^2(\log\log n)(\log\log\log n)\right)$ gates (Shor's algorithm)

Searching through n unordered items:

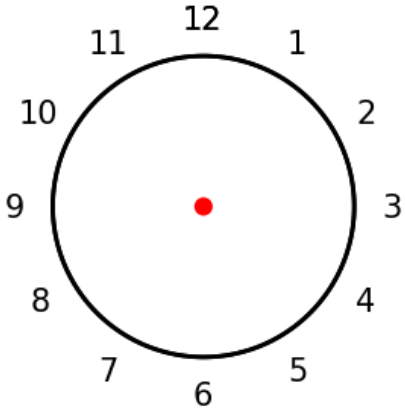
- Classical: exponential: $O(n)$ items tested
- Quantum: exponential: $O(\sqrt{n})$ items tested (Grover's Algorithm)

The power of a quantum computer

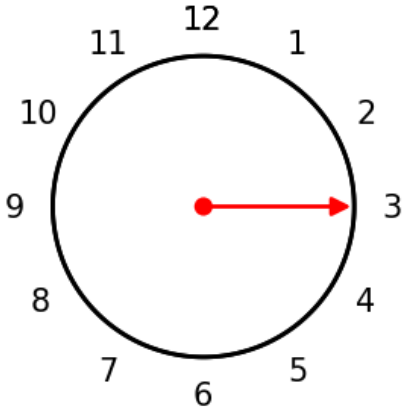
Power:
A quantum gate operates on *all the possible states simultaneously*.
(Think $|x, 0\rangle \mapsto |x, f(x)\rangle$.)

Limitation:
You can't see the result.

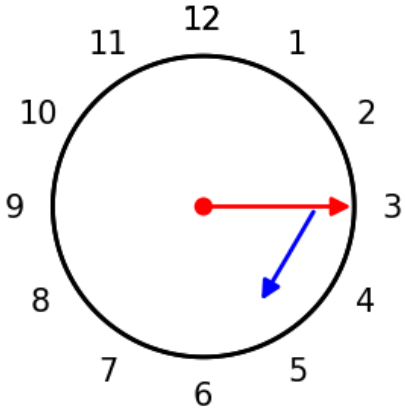
Interlude: modular arithmetic



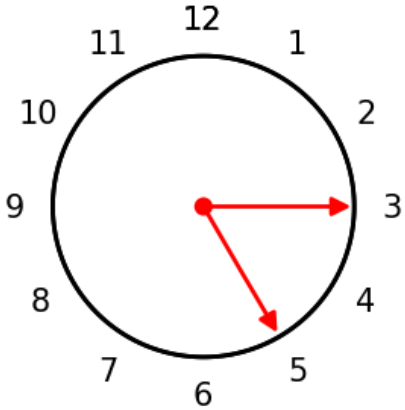
Interlude: modular arithmetic



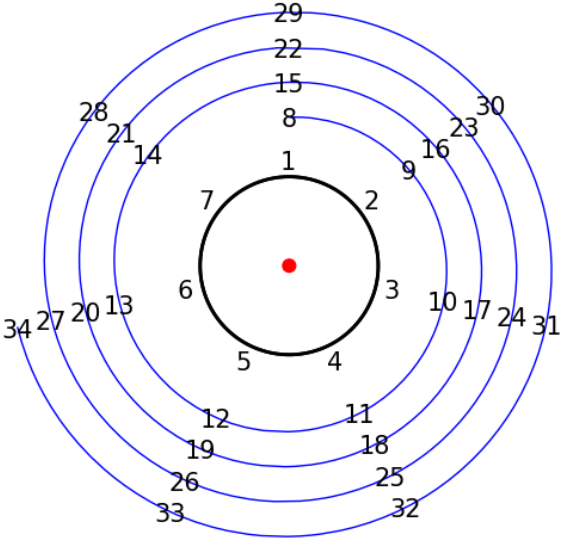
Interlude: modular arithmetic



Interlude: modular arithmetic



Interlude: modular arithmetic



Interlude: modular arithmetic

Let n be an integer (the *modulus*). We can define

$$\mathbb{Z}/n\mathbb{Z}$$

to be the set of classes of integers mod n.

Two integers are in the same class if they differ by a multiple of n.

For example, modulo 3 there are three classes:

$$\begin{aligned} 0 &: \{\ldots, -3, 0, 3, 6, 9, \ldots\} \\ 1 &: \{\ldots, -2, 1, 4, 7, 10, \ldots\} \\ 2 &: \{\ldots, -1, 2, 5, 8, 11, \ldots\} \end{aligned}$$

Modular arithmetic: addition and multiplication

$$
\begin{aligned}
0 &: \{\ldots, -3, 0, 3, 6, 9, \ldots\} \\
1 &: \{\ldots, -2, 1, 4, 7, 10, \ldots\} \\
2 &: \{\ldots, -1, 2, 5, 8, 11, \ldots\}
\end{aligned}
$$

The amazing thing is that addition and multiplication respect this, e.g.

$$1+0, 1+6, 4+3, 7+0, \ldots$$

are all in the same class.

So we can define $\mathbb{Z}/n\mathbb{Z}$ to be n items (the classes), with an addition and multiplication law.

Example: $\mathbb{Z}/3\mathbb{Z}$

$$0 : \{\ldots, -3, 0, 3, 6, 9, \ldots\}$$
$$1 : \{\ldots, -2, 1, 4, 7, 10, \ldots\}$$
$$2 : \{\ldots, -1, 2, 5, 8, 11, \ldots\}$$

Addition

	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Multiplication

	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Interlude: Fourier analysis

The space of functions

$$\{f : \{0, 1, 2, \ldots, N\} \to \mathbb{C}\}$$

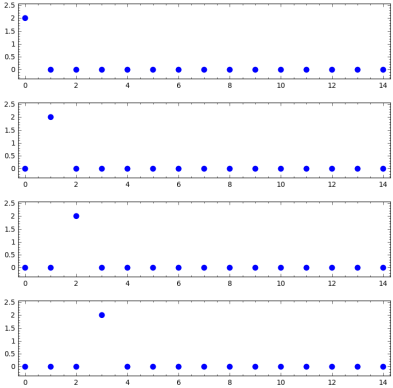
is a vector space.

We have addition and scalar multiplication.

Interlude: Fourier analysis

$$\{f : \{0, 1, 2, \ldots, N\} \to \mathbb{C}\}$$

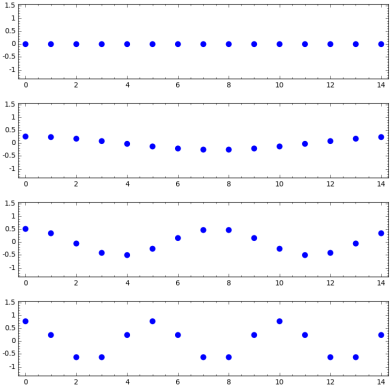
Standard basis:



Interlude: Fourier analysis

$$\{f : \{0, 1, 2, \ldots, N\} \to \mathbb{C}\}$$

Fourier basis:



Interlude: Fourier analysis

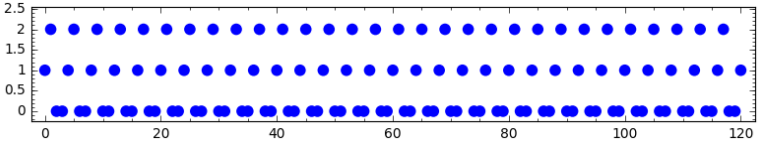
Discrete Fourier Transform:

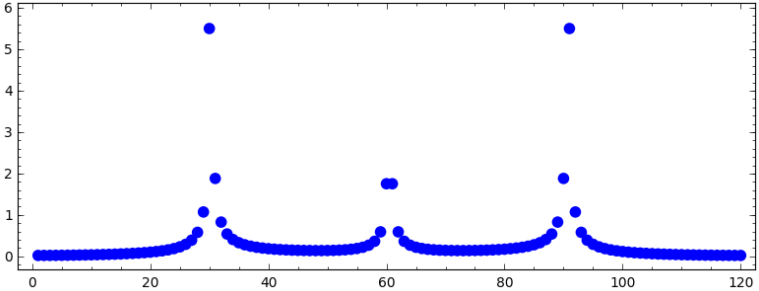
$$(x_j)_{j=1}^N \mapsto \left(\frac{1}{\sqrt{N}} \sum_{j=1}^N x_j e^{\frac{(2\pi i)jk}{N}} \right)_{k=1}^N$$

This changes between standard and Fourier bases.

Interlude: Fourier analysis

Discrete Fourier Transform detects frequency:
(N = 121, period = 4)





Quantum Fourier Transform

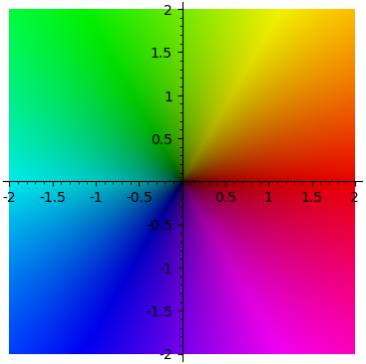
Given n qubits (2^n basic states), we define the QFT:

$$\sum_{j=1}^{N} x_j |j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=1}^{N} \left(\sum_{j=1}^{N} x_j e^{\frac{-(2\pi i)jk}{N}} \right) |k\rangle.$$

This is a unitary transformation on the vector of coefficients, i.e. a quantum gate.

QFT Demo

Domain coloring of the complex plane:



QFT Demo

DEMO

Lemma for Factoring on a Quantum Computer

In order to factor N, it is enough to know the period of the function

$$x \mapsto a^x \pmod{N}$$

for some a which is not too annoyingly trivial.

Hint: Fermat's Little Theorem says this order divides $\varphi(N)$.

Shor's algorithm for period finding

DEMO

How to factor N?

- classically reduce the problem to finding the period of a function $x \mapsto a^x \pmod N$.
- store a uniform linear combination of the states "x".
- use quantum gates to compute $x \mapsto a^x$ on all states simultaneously.
- result: a uniform linear combination of the states "$|x, a^x\rangle$".
- measure the second half: obtain a *periodic* linear combination of states.
- perform a *quantum fourier transform* to pick up the frequency.
- measure the result to read off the frequency.
- determine the period from the reading (classical).

Current public-key cryptography: the big three

- Factoring: Given $N = pq$, find p and q.
- Discrete Log mod p: Given $p, g, h = g^s \pmod{p}$, find s.
- Discrete Log on elliptic curves: Given E, P, $Q = [s]P$, find s.

Diffie-Hellman Key Exchange

Setup: p (modulus), g

Alice		Bob
Secret: a		Secret: b
$g^a \pmod p$	$\longrightarrow$	$g^a \pmod p$
$g^b \pmod p$	$\longleftarrow$	$g^b \pmod p$
Compute: $(g^b)^a \equiv g^{ab}$	$\leftarrow$ shared secret! $\rightarrow$	Compute: $(g^a)^b \equiv g^{ab}$

Diffie-Hellman Key Exchange

Setup: p (modulus), g

Alice		Bob
Secret: a		Secret: b
$g^a \pmod p$	$\longrightarrow$	$g^a \pmod p$
$g^b \pmod p$	$\longleftarrow$	$g^b \pmod p$
Compute: $(g^b)^a \equiv g^{ab}$	$\leftarrow$ shared secret! $\rightarrow$	Compute: $(g^a)^b \equiv g^{ab}$

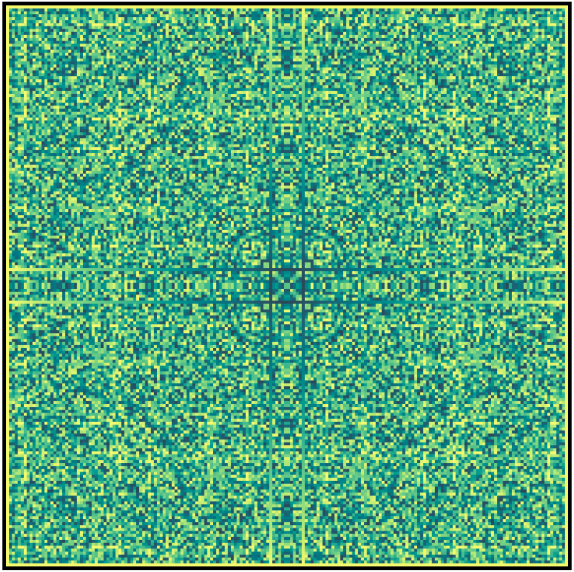
Hard Problem (Discrete Logarithm Problem)

Given $g, h = g^s \pmod p$, *find* s.

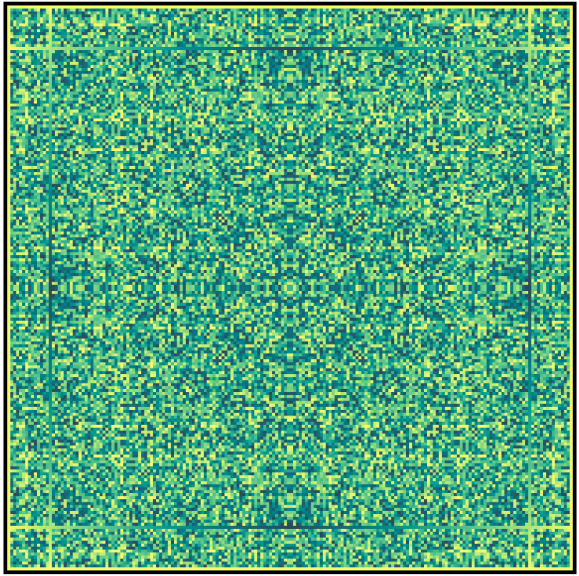
DLP Demo

DEMOS

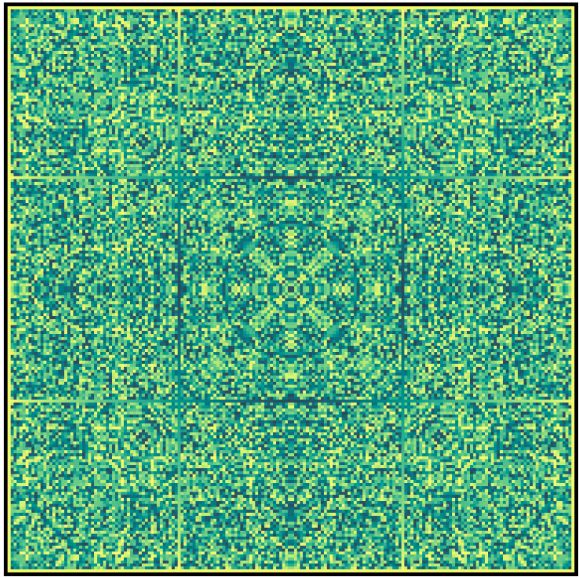
Diffie-Hellman Carpets



Diffie-Hellman Carpets



Diffie-Hellman Carpets



Adapting Shor's algorithm

In Shor's algorithm, the computer was able to find a *period* t such that $f(x) = f(x+t)$. This is an example of a *kernel* of a map.

Definition

The *kernel* of a map f is the set of inputs that give 0.

In particular, the set of periods is the kernel of the map

$$f : \mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}, \quad t \mapsto f(x+t) - f(x).$$

Shor's algorithm for discrete logarithm

Hard Problem (Discrete Logarithm Problem)

Given $g, h = g^s \pmod{p}$, *find* s.

We use a variant of Shor where we want to find the *kernel* of the function

$$f : \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/(p-1)\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}, \quad (a, b) \mapsto g^a h^{-b}.$$

The kernel of this function is exactly

$$\{(sx, x) : x\}$$

So if we can find any example of an element in this kernel, we can divide the two coordinates to obtain s.

Post-quantum candidates

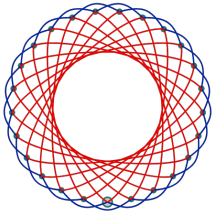
- lattice-based cryptography
- isogeny-based cryptography
- code-based cryptography
- multivariate cryptography

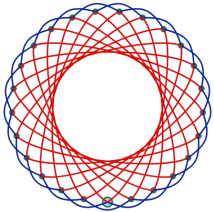
Post-quantum candidates

- lattice-based cryptography
- isogeny-based cryptography
- code-based cryptography
- multivariate cryptography

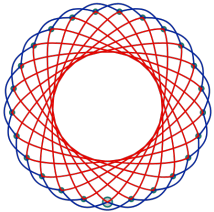
Isogeny-based cryptography: CSIDH

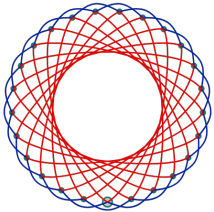
Alice





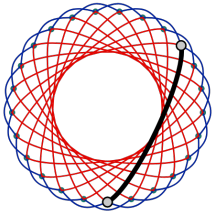
Bob

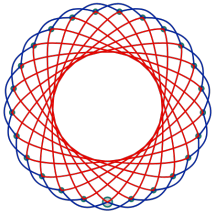




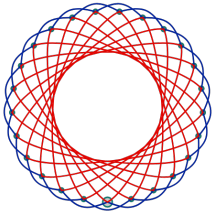
Isogeny-based cryptography: CSIDH

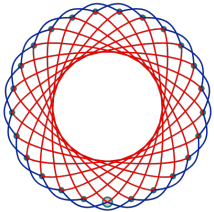
Alice





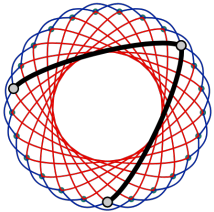
Bob

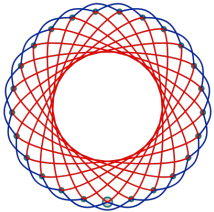




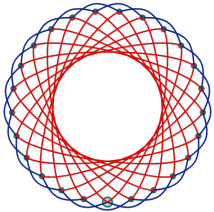
Isogeny-based cryptography: CSIDH

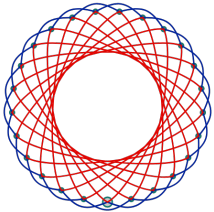
Alice





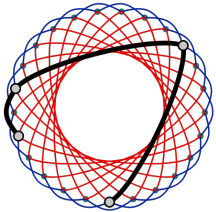
Bob

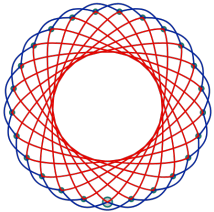




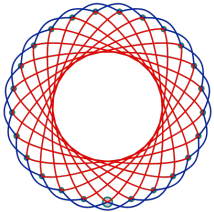
Isogeny-based cryptography: CSIDH

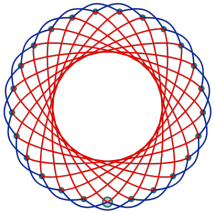
Alice





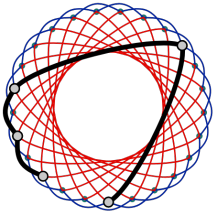
Bob

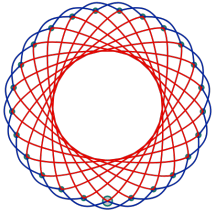




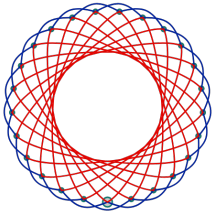
Isogeny-based cryptography: CSIDH

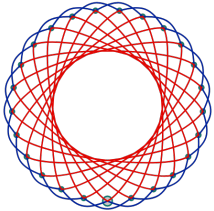
Alice





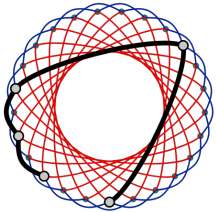
Bob

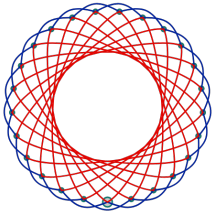




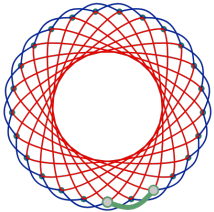
Isogeny-based cryptography: CSIDH

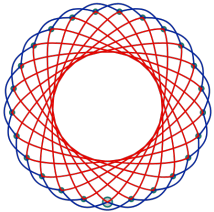
Alice





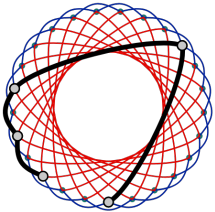
Bob

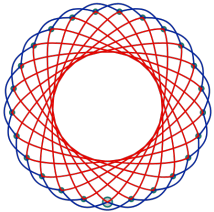




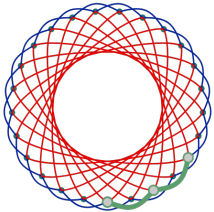
Isogeny-based cryptography: CSIDH

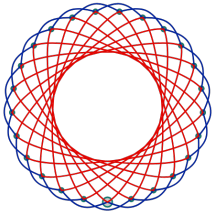
Alice





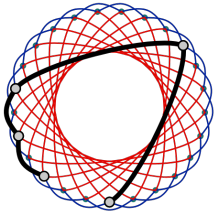
Bob

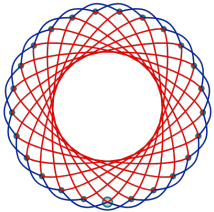




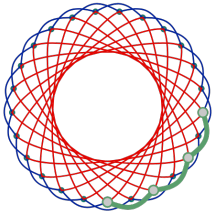
Isogeny-based cryptography: CSIDH

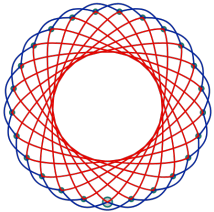
Alice





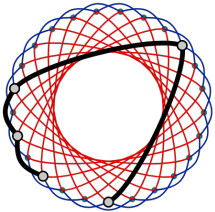
Bob

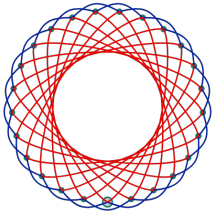




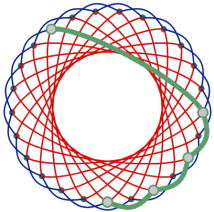
Isogeny-based cryptography: CSIDH

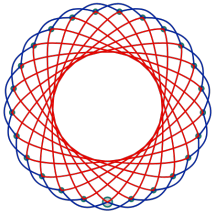
Alice





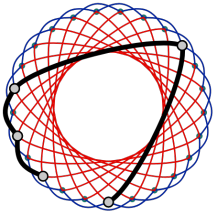
Bob

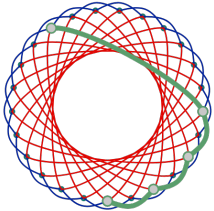




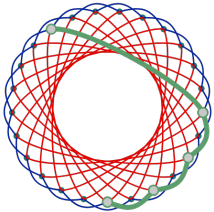
Isogeny-based cryptography: CSIDH

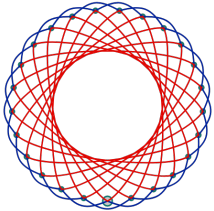
Alice





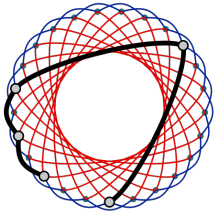
Bob

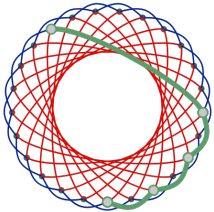




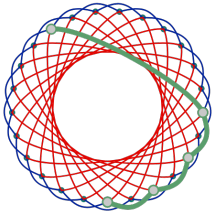
Isogeny-based cryptography: CSIDH

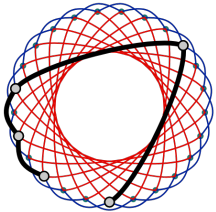
Alice





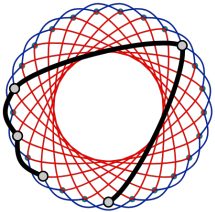
Bob

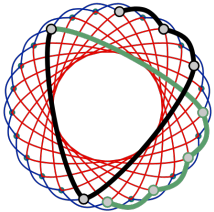




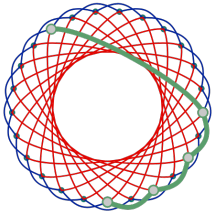
Isogeny-based cryptography: CSIDH

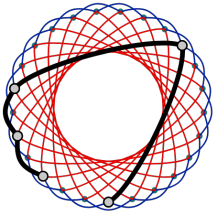
Alice





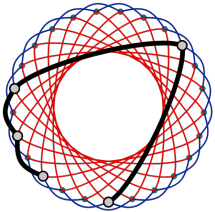
Bob

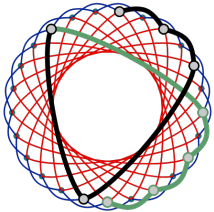




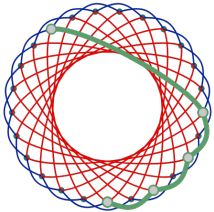
Isogeny-based cryptography: CSIDH

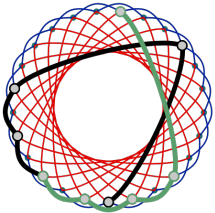
Alice





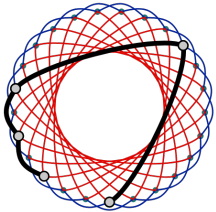
Bob

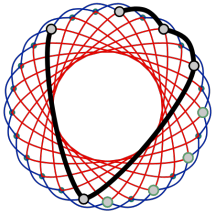




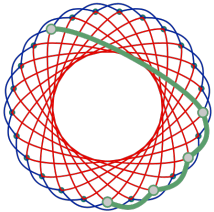
Isogeny-based cryptography: CSIDH

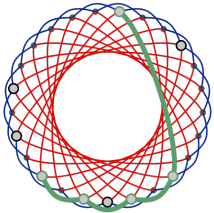
Alice





Bob



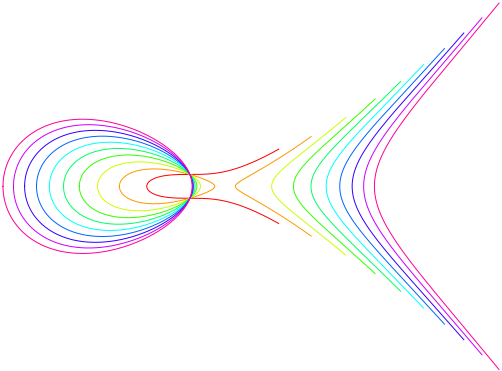


Elliptic Curves

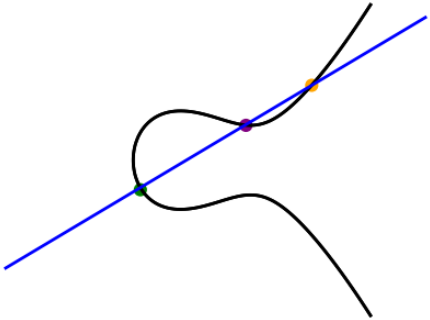
Definition

An *elliptic curve* is a smooth curve given by

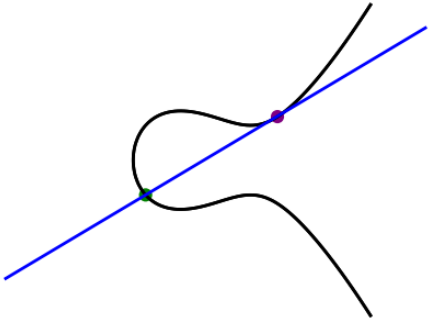
$$y^2 = x^3 + ax + b$$



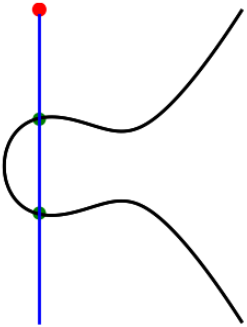
Addition law



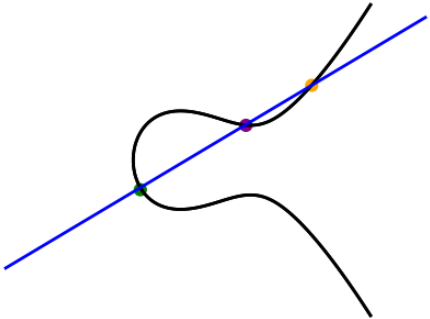
Addition law



Addition law



Addition law



Three collinear points add to 0, the point at infinity.
Addition law is given by rational functions.

Addition law

The resulting group is *finite abelian group*, and the addition law is given by rational functions!
If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, then

$$P + Q = \left(\left(\frac{y_P - y_Q}{x_P - x_Q} \right)^2 - x_P - x_Q, -y_P - \left(\frac{(y_P - y_Q)(x_R - x_P)}{x_P - x_Q} \right) \right).$$

We can use this for an Elliptic Curve Discrete Logarithm Problem: given P and a multiple $[s]P$, find s.

Isogenies

A group homomorphism between elliptic curves is called an *isogeny* if it is realized as rational functions.

Example

Modulo 11, there is an isogeny from $y^2 = x^3 + 1$ to $y^2 = x^3 + 6$, given by

$$(x, y) \mapsto \left(\frac{x^3 + 4}{x^2}, \frac{x^3 y + 3y}{x^3} \right)$$

with kernel

$$\{(0, 1), (0, 10), id\}.$$

The *degree* of the isogeny is the cardinality of its kernel. (e.g. this is a 3-isogeny)

Isogeny graphs

$G_{\ell,n}$ mod q has

- vertices = elliptic curves mod q with n points (up to isomorphism)
  - label: $j = 1728\frac{4a^3}{4a^3+27b^2}$, isomorphism invariant
- edges = E and E' connected by an ℓ-isogeny
  - label: kernel of isogeny

For CSIDH, use the union of a few small ℓ, prime modulus p, and $n = p + 1$.

Hard Problem

Given two vertices (j-invariants) construct a path between them (sequence of isogenies/kernels).

Comparison to DLP

The graph has a *group action* (by a certain *class group*).

Hard Problem (Rephrased Isogeny Problem)

Given E and $s \cdot E$, compute s.

Hard Problem (Discrete Log Problem)

Modulo p, given g and g^s, compute s.

Shor's algorithm computes the 'period' (kernel) of the map

$$(x, y) \mapsto g^x (g^s)^{-y},$$

which gives away the secret s.

In our case, no way to 'multiply': $x \cdot E \overset{?}{=} -y \cdot (s \cdot E)$

Thank you!