Isogeny-based Cryptography
I & II

Lorenz Panny

Academia Sinica

Post-Quantum Crypto Mini-School 2022, Taipei, 14 July 2022

Big picture 🔎🔎

- Isogenies are a source of exponentially-sized graphs.

Big picture 🔍🔍

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.

Big picture 🔍🔍

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.

Big picture 🔎🔎

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)

Big picture 🔍🔍

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)
- Enough structure to navigate the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths.

Big picture 🔎🔎

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No efficient* algorithms to recover paths from endpoints.
  (*Both* classical and quantum!)
- Enough structure to navigate the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths.

Finding graphs with *almost* all of these properties is easy —
but getting all at once seems rare.

Crypto on graphs?

Diffie–Hellman key exchange 1976

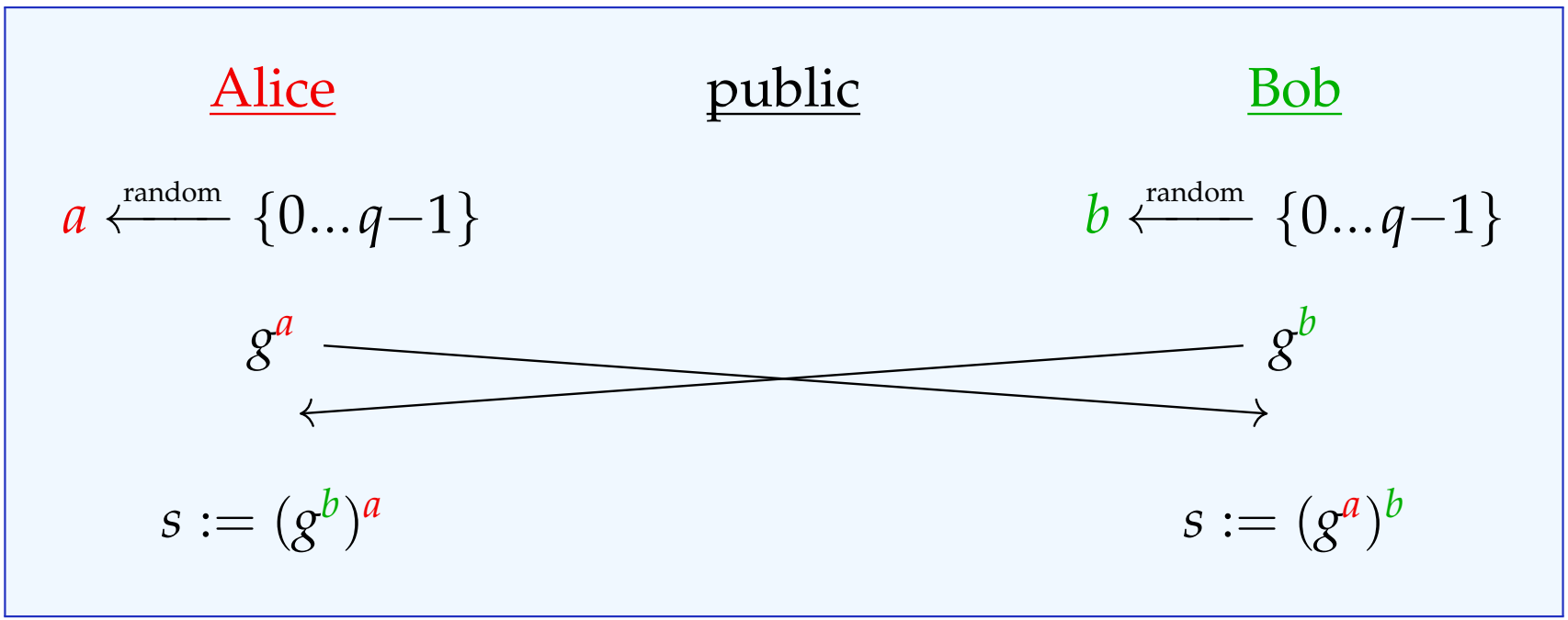
Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q

Diffie–Hellman key exchange 1976

Public parameters:

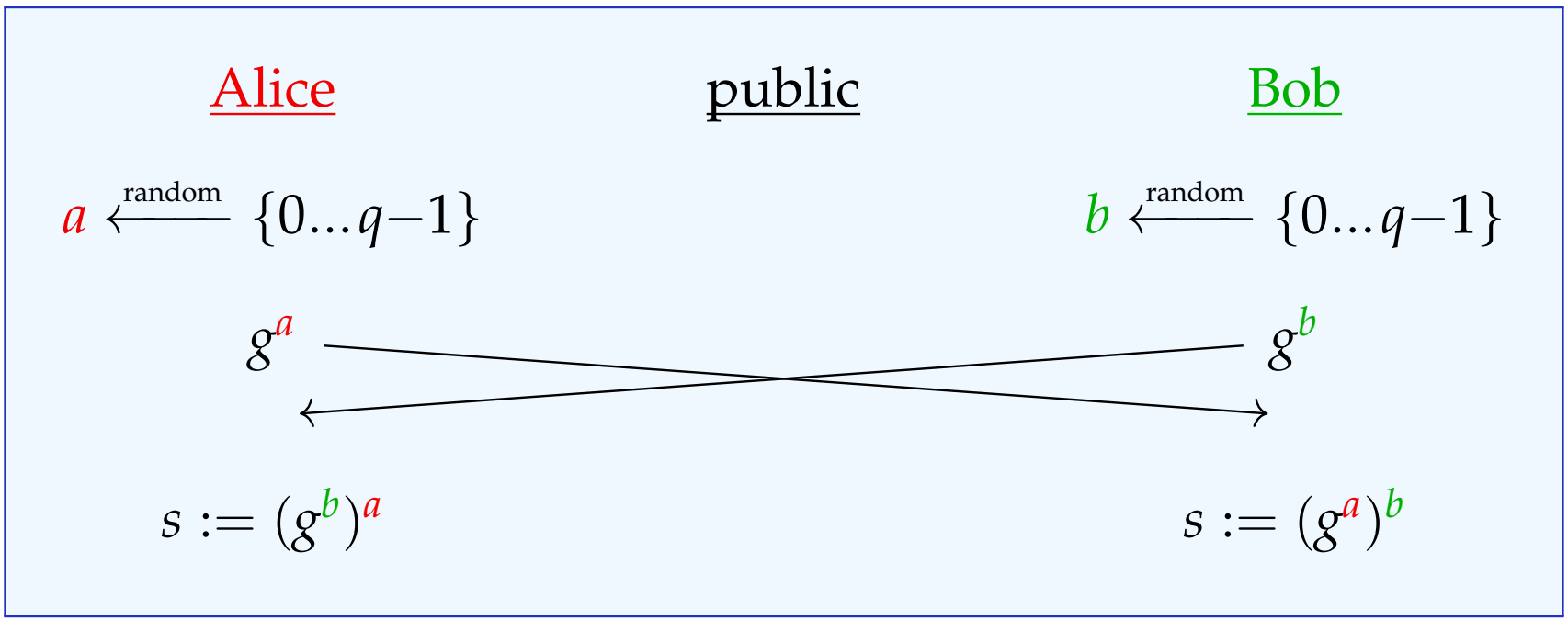
- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q



Diffie–Hellman key exchange 1976

Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q



Fundamental reason this works: $\cdot^a$ and $\cdot^b$ are commutative!

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Is this a good idea?

Diffie–Hellman: Bob vs. Eve

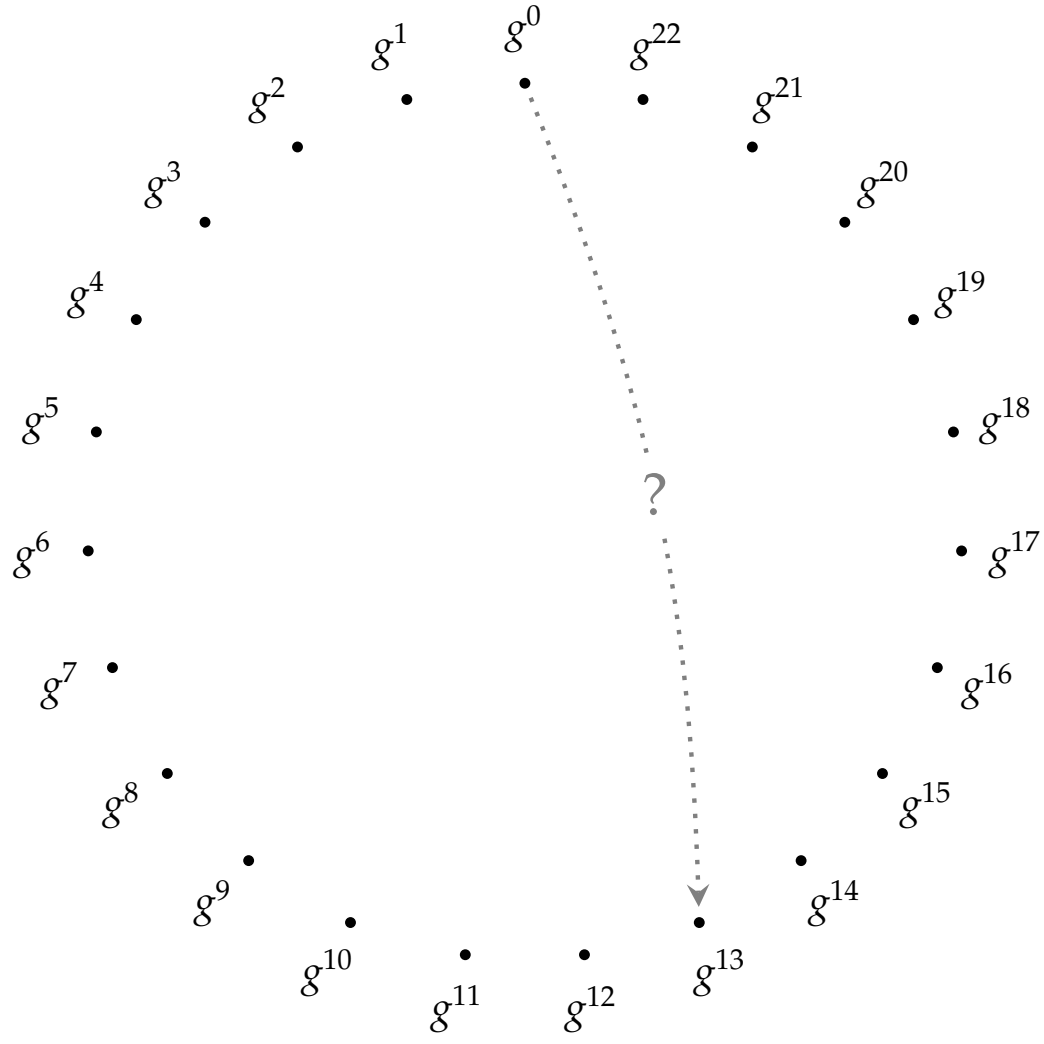
Bob	**Attacker** Eve
1. Set $t \leftarrow g$.	1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$.	2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$.	3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$.	4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
...	...
$b-2$. Set $t \leftarrow t \cdot g$.	$b-2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-2$.
$b-1$. Set $t \leftarrow t \cdot g$.	$b-1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-1$.
b. Publish $B \leftarrow t \cdot g$.	b. Set $t \leftarrow t \cdot g$. If $t = B$ return b.
	$b+1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+1$.
	$b+2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+2$.
	...

Diffie–Hellman: Bob vs. Eve

Bob	**Attacker** Eve
1. Set $t \leftarrow g$.	1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$.	2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$.	3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$.	4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
...	...
$b-2$. Set $t \leftarrow t \cdot g$.	$b-2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-2$.
$b-1$. Set $t \leftarrow t \cdot g$.	$b-1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-1$.
b. Publish $B \leftarrow t \cdot g$.	b. Set $t \leftarrow t \cdot g$. If $t = B$ return b.
	$b+1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+1$.
	$b+2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+2$.
	...

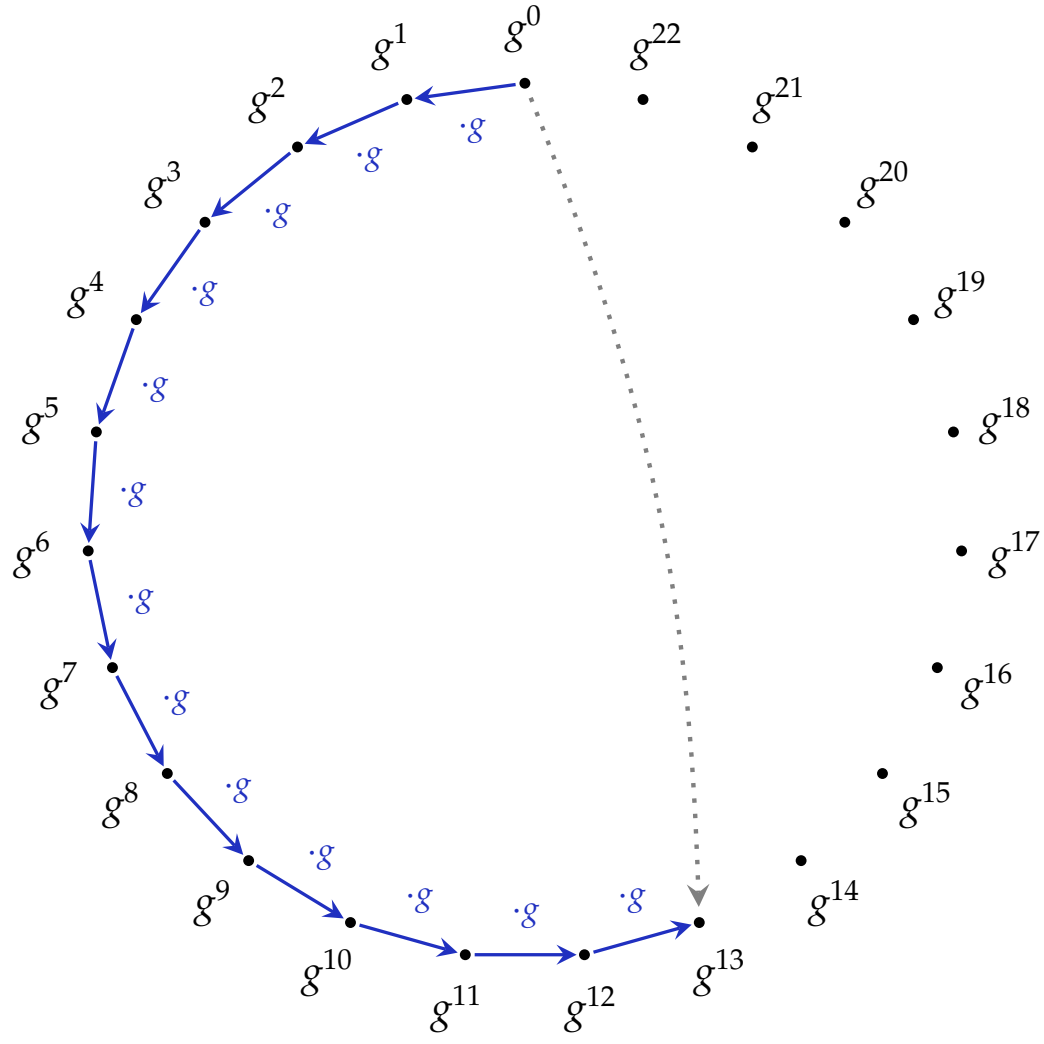
Effort for both: $O(\#G)$. Bob needs to be smarter.

(This attacker is also kind of dumb, but that doesn't matter for my point here.)



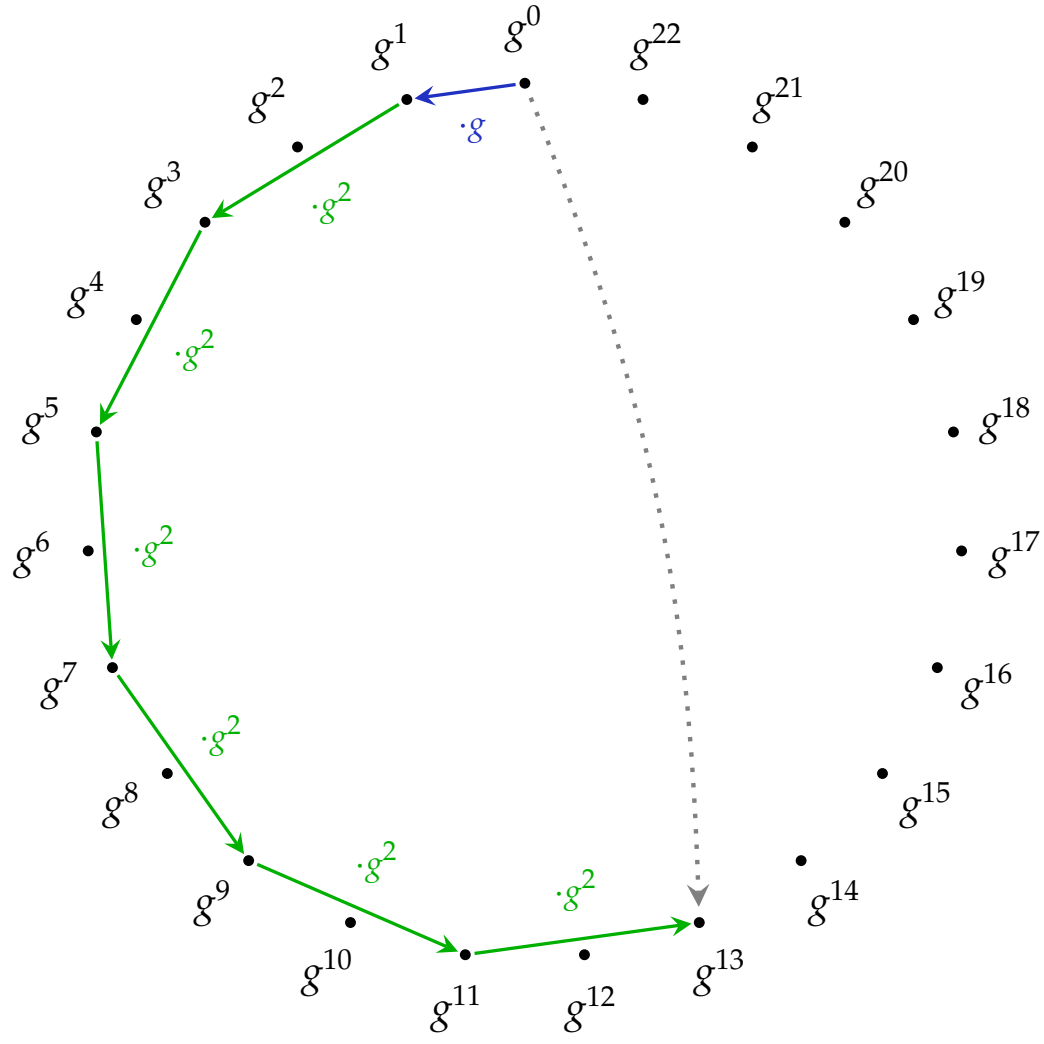
Bob computes his public key g^{13} from g.

multiply



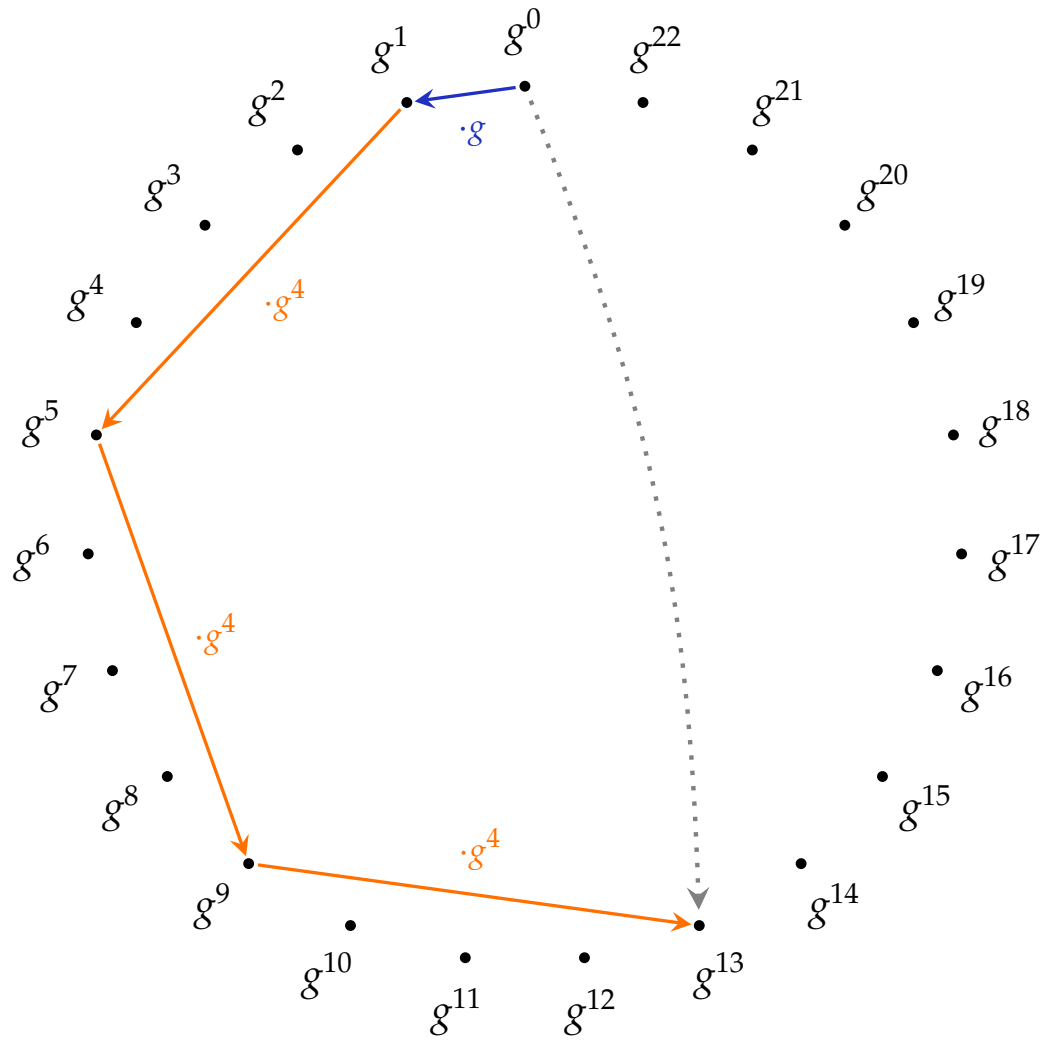
Bob computes his public key g^{13} from g.

Square-and-multiply



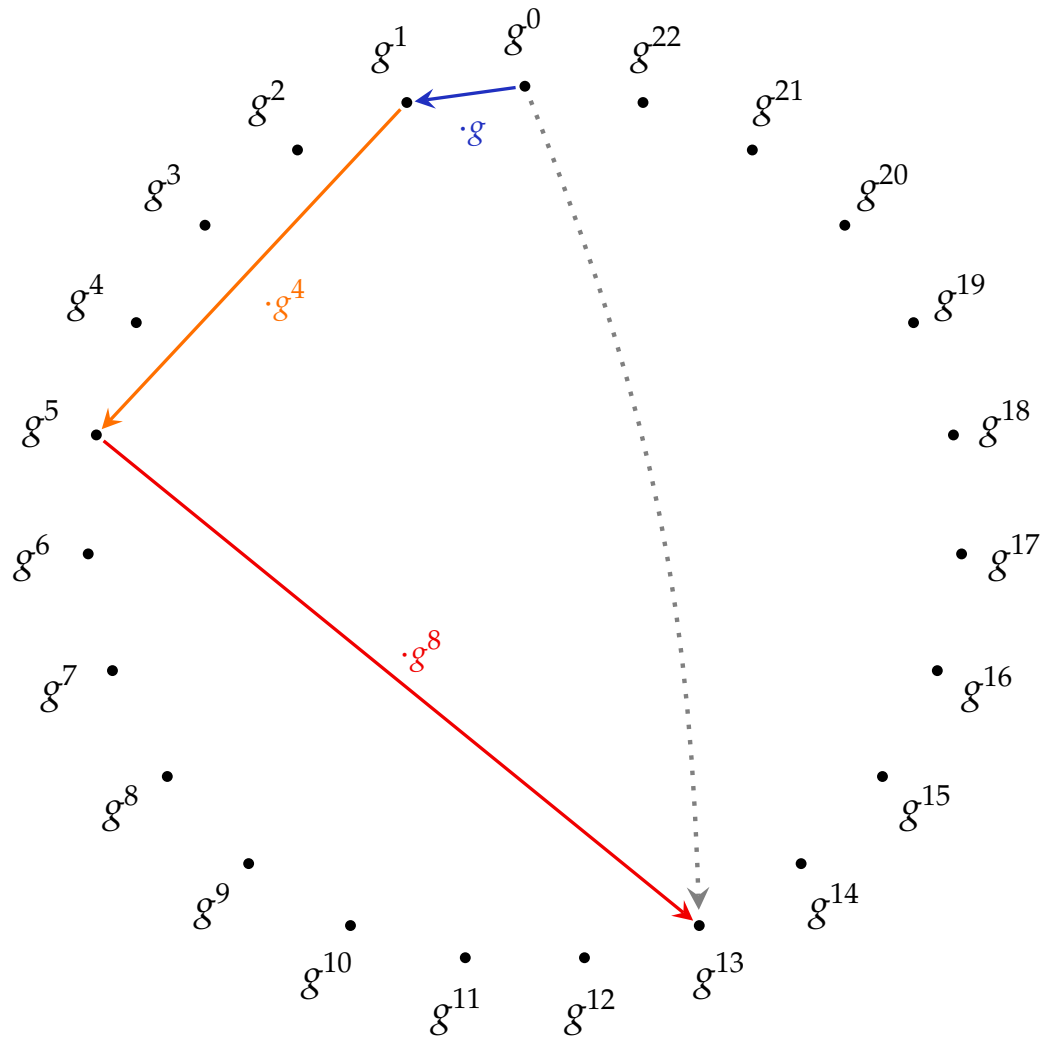
Bob computes his public key g^{13} from g.

Square-and-multiply-and-square-and-multiply



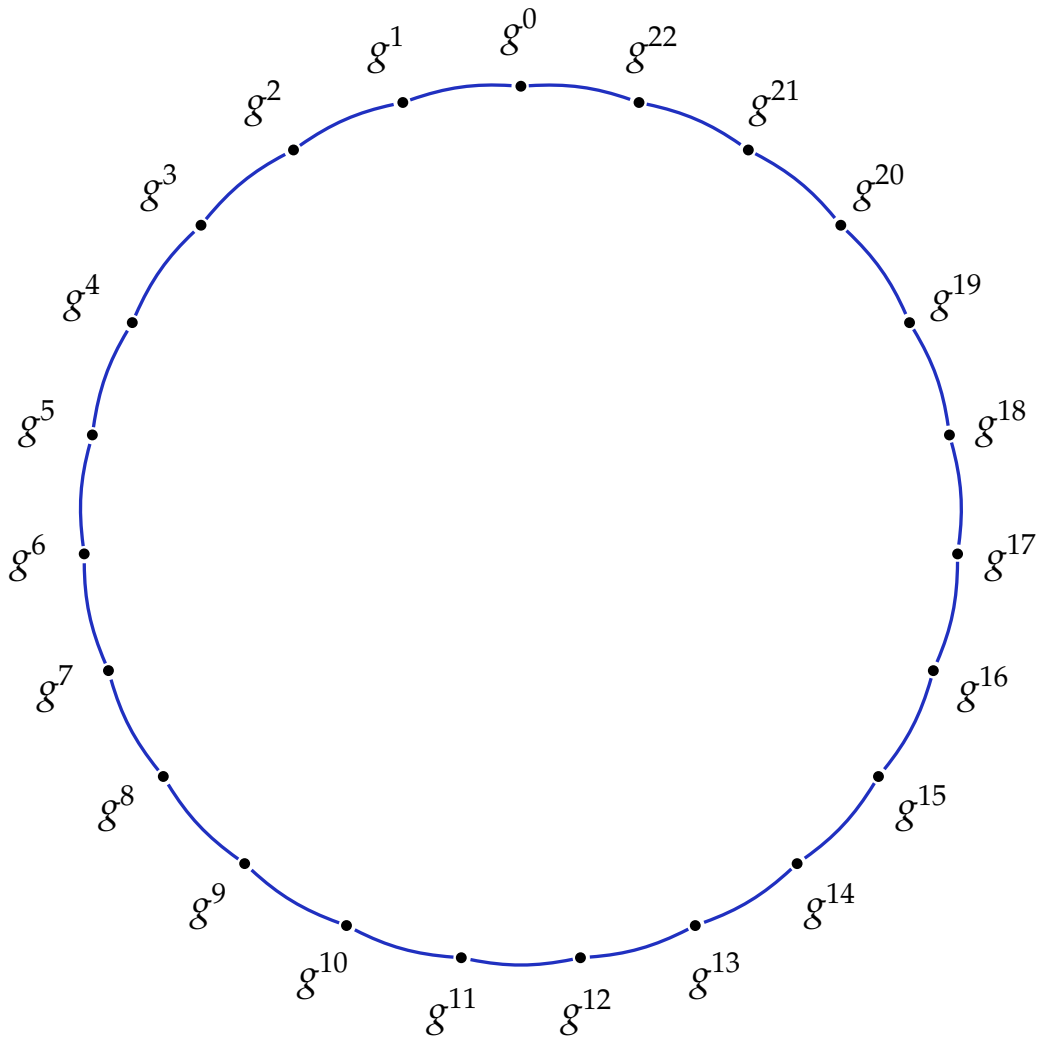
Bob computes his public key g^{13} from g.

Square-and-multiply-and-square-and-multiply-and-squ

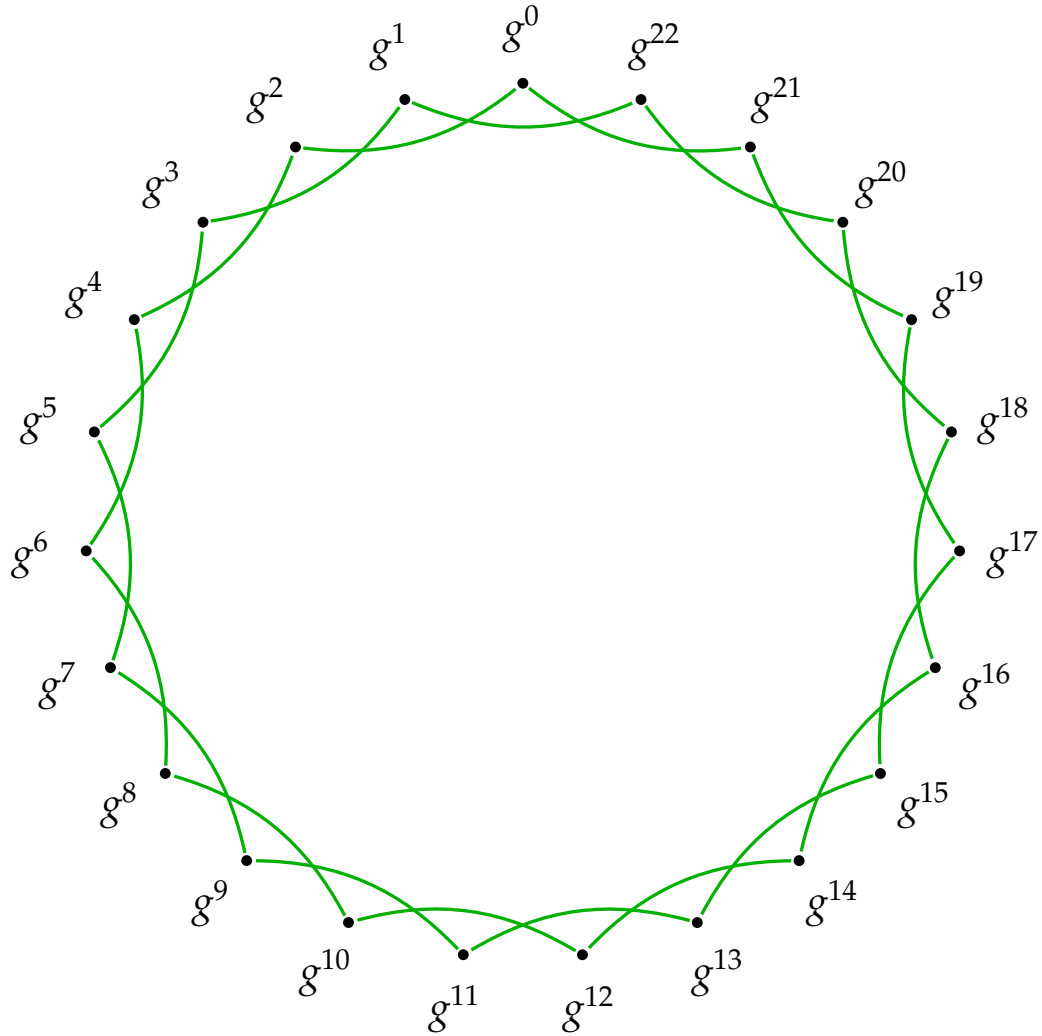


Bob computes his public key g^{13} from g.

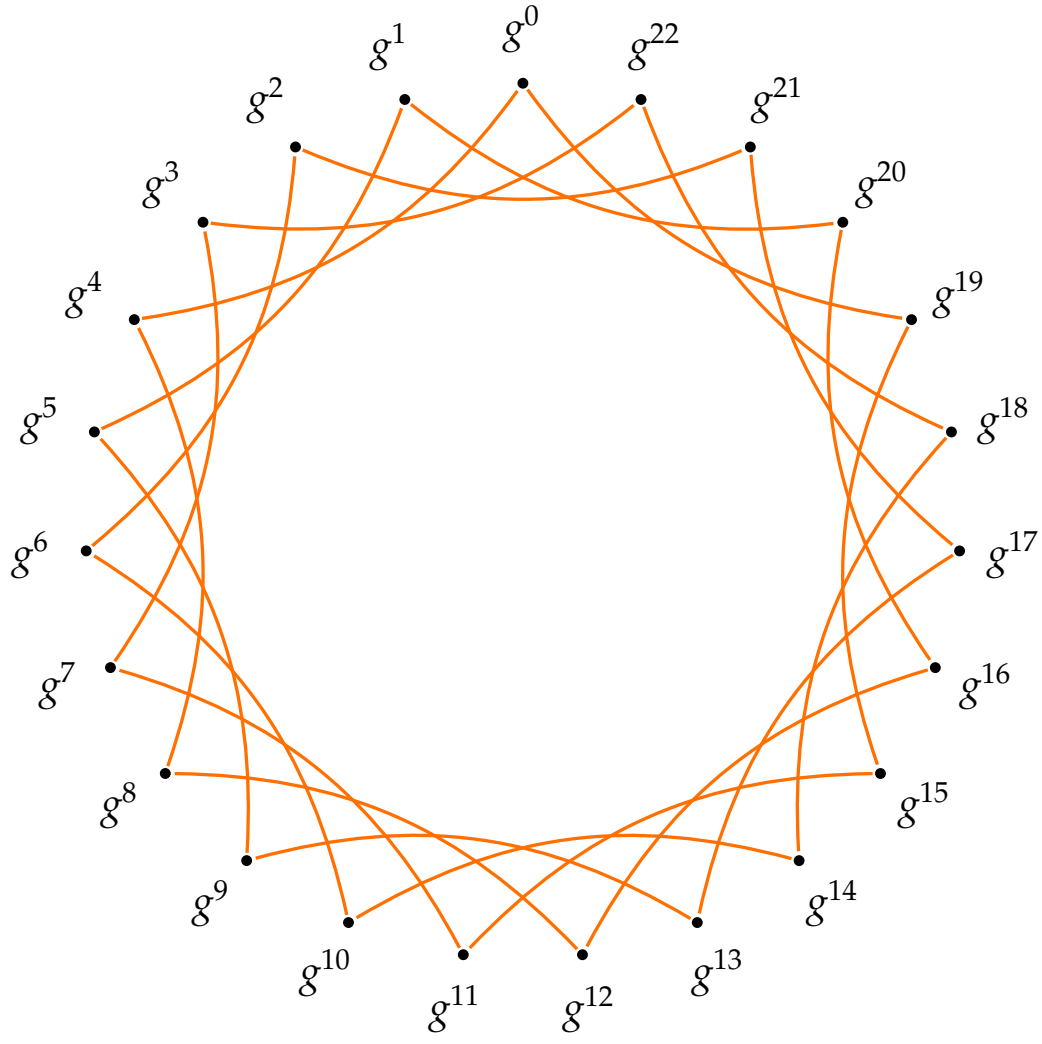
Square-and-multiply as graphs



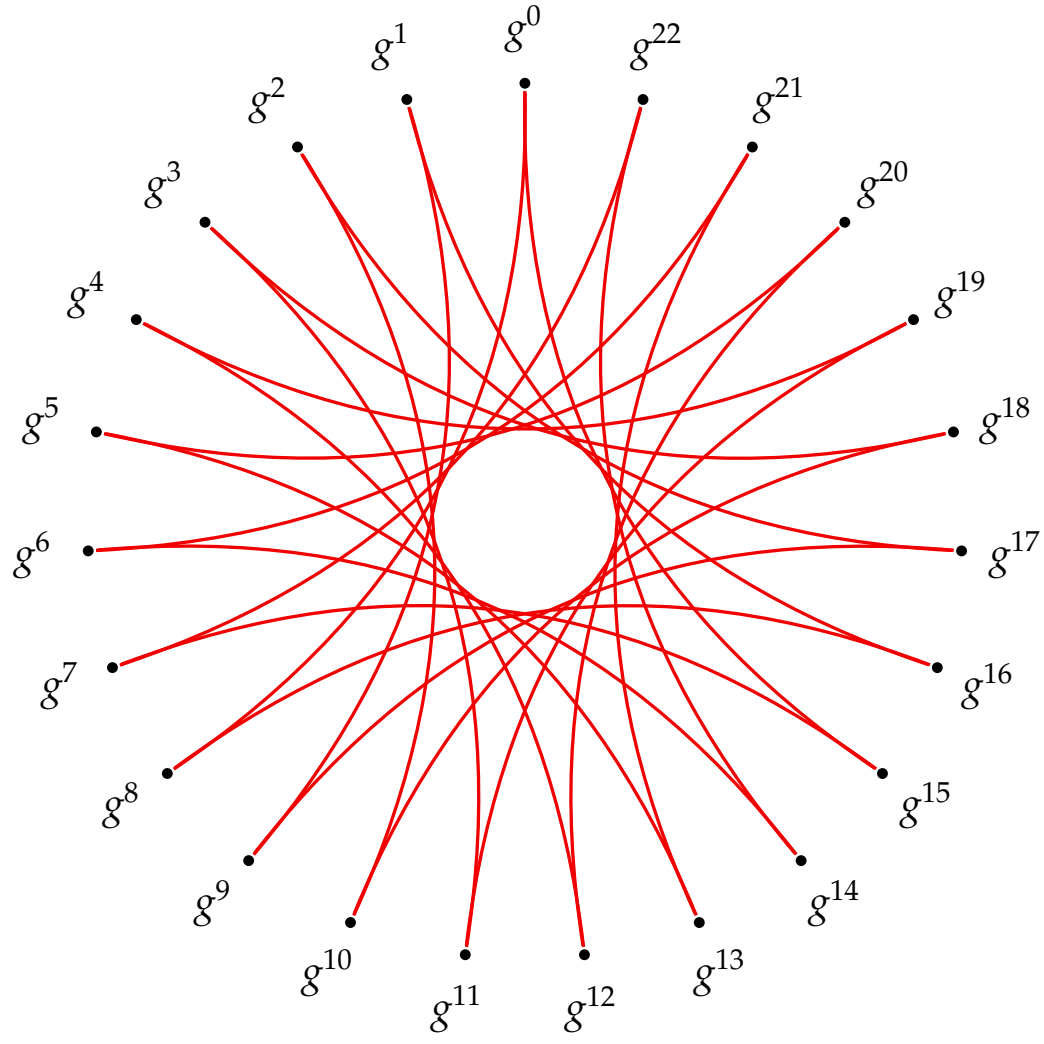
Square-and-multiply as graphs



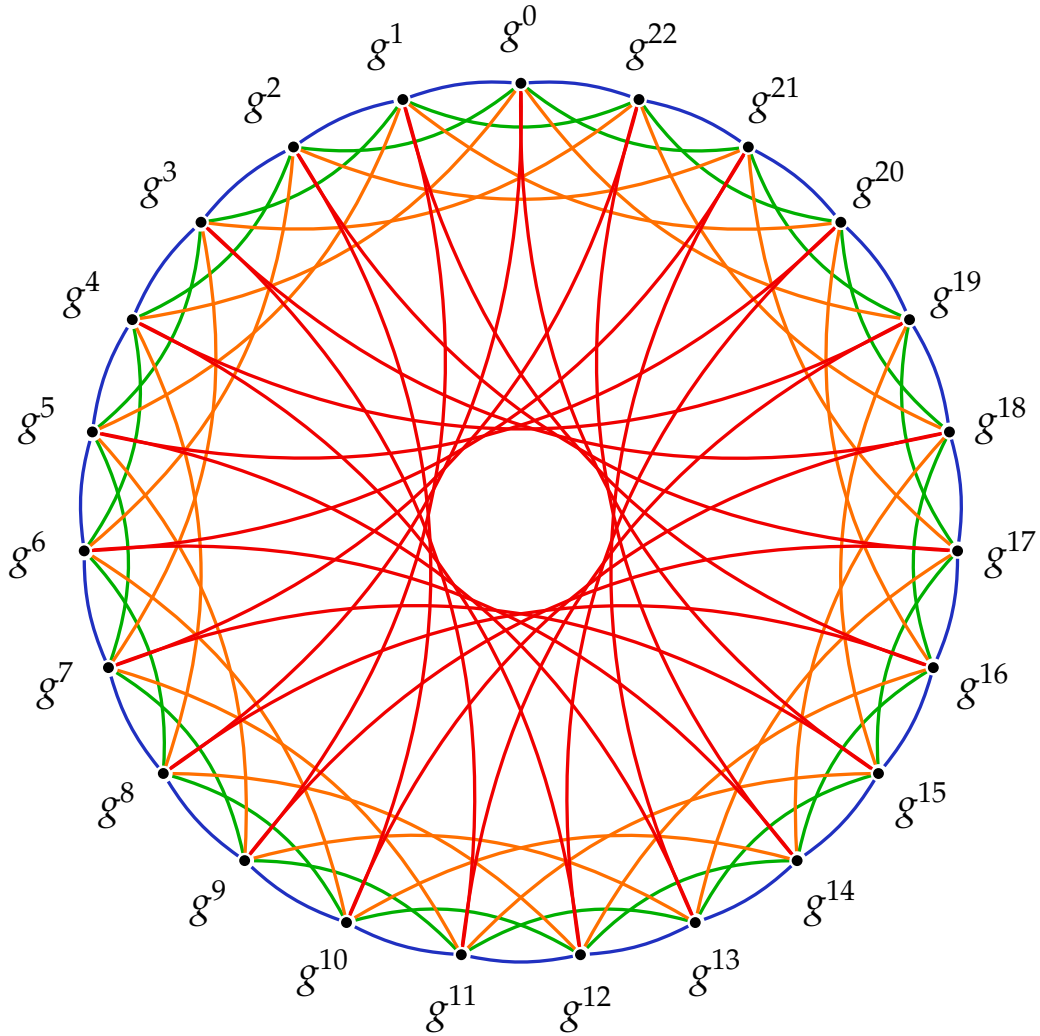
Square-and-multiply as graphs



Square-and-multiply as graphs



Square-and-multiply as a graph



Crypto on graphs?

We've been doing it all the time!

The fast mixing requirement

Fast mixing: paths of length log(# nodes) to everywhere.

The fast mixing requirement

Fast mixing: paths of length $\log(\#\,\text{nodes})$ to everywhere.

With square-and-multiply, computing $\alpha \mapsto g^\alpha$ takes $\Theta(\log \alpha)$.

The fast mixing requirement

Fast mixing: paths of length $\log(\#\,\text{nodes})$ to everywhere.

With square-and-multiply, computing $\alpha \mapsto g^\alpha$ takes $\Theta(\log \alpha)$.

For well-chosen groups, computing $g^\alpha \mapsto \alpha$ takes $\Theta(\sqrt{\#G})$.

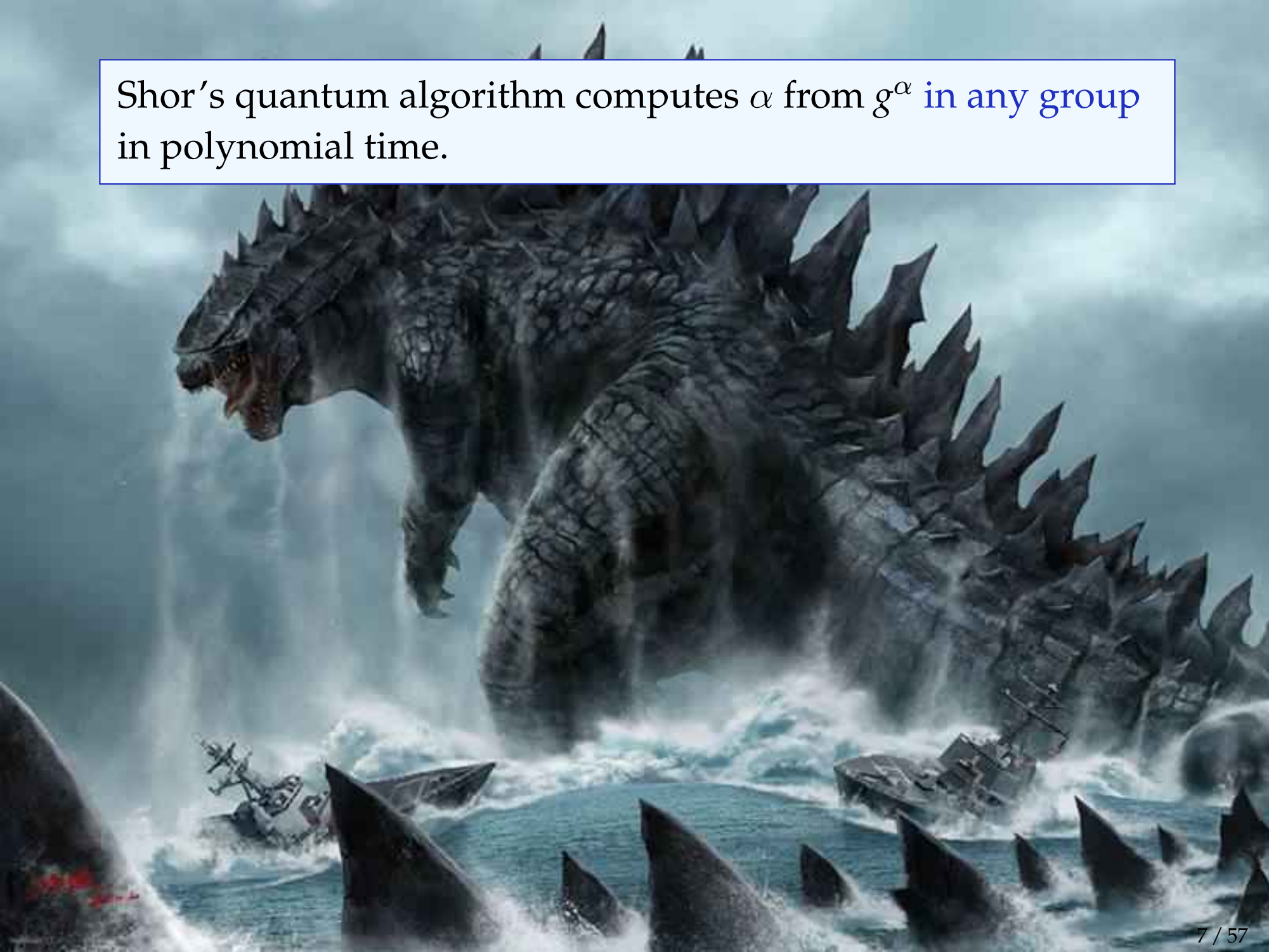
The fast mixing requirement

Fast mixing: paths of length $\log(\#\,\text{nodes})$ to everywhere.

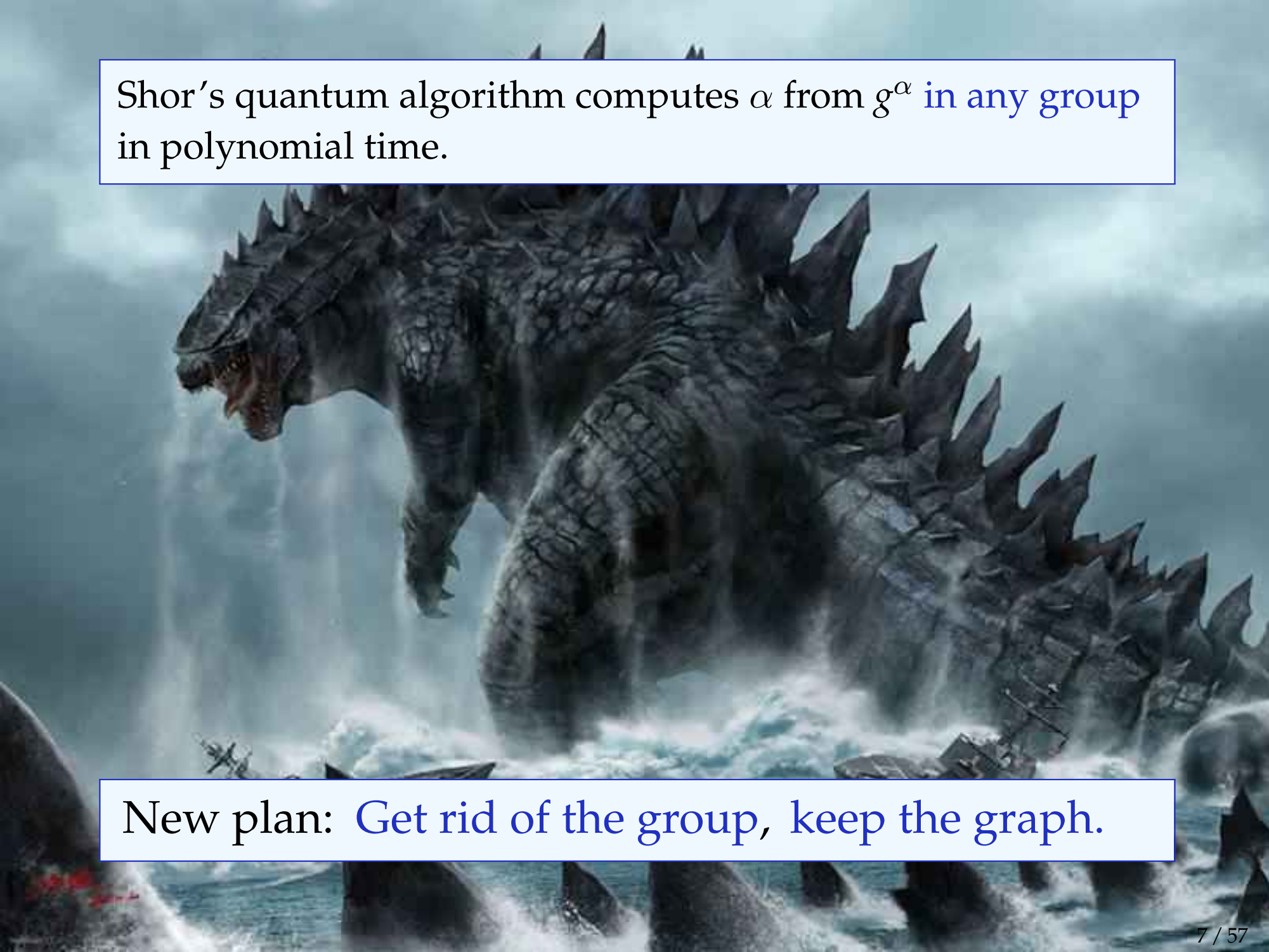
With square-and-multiply, computing $\alpha \mapsto g^\alpha$ takes $\Theta(\log \alpha)$.

For well-chosen groups, computing $g^\alpha \mapsto \alpha$ takes $\Theta(\sqrt{\#G})$.

$\rightsquigarrow$ Exponential separation!



Shor's quantum algorithm computes α from g^α in any group in polynomial time.



Shor's quantum algorithm computes α from g^α in any group in polynomial time.

New plan: Get rid of the group, keep the graph.

The upshot

In some cases,

isogeny graphs

can replace DLP-based constructions post-quantumly.

The upshot

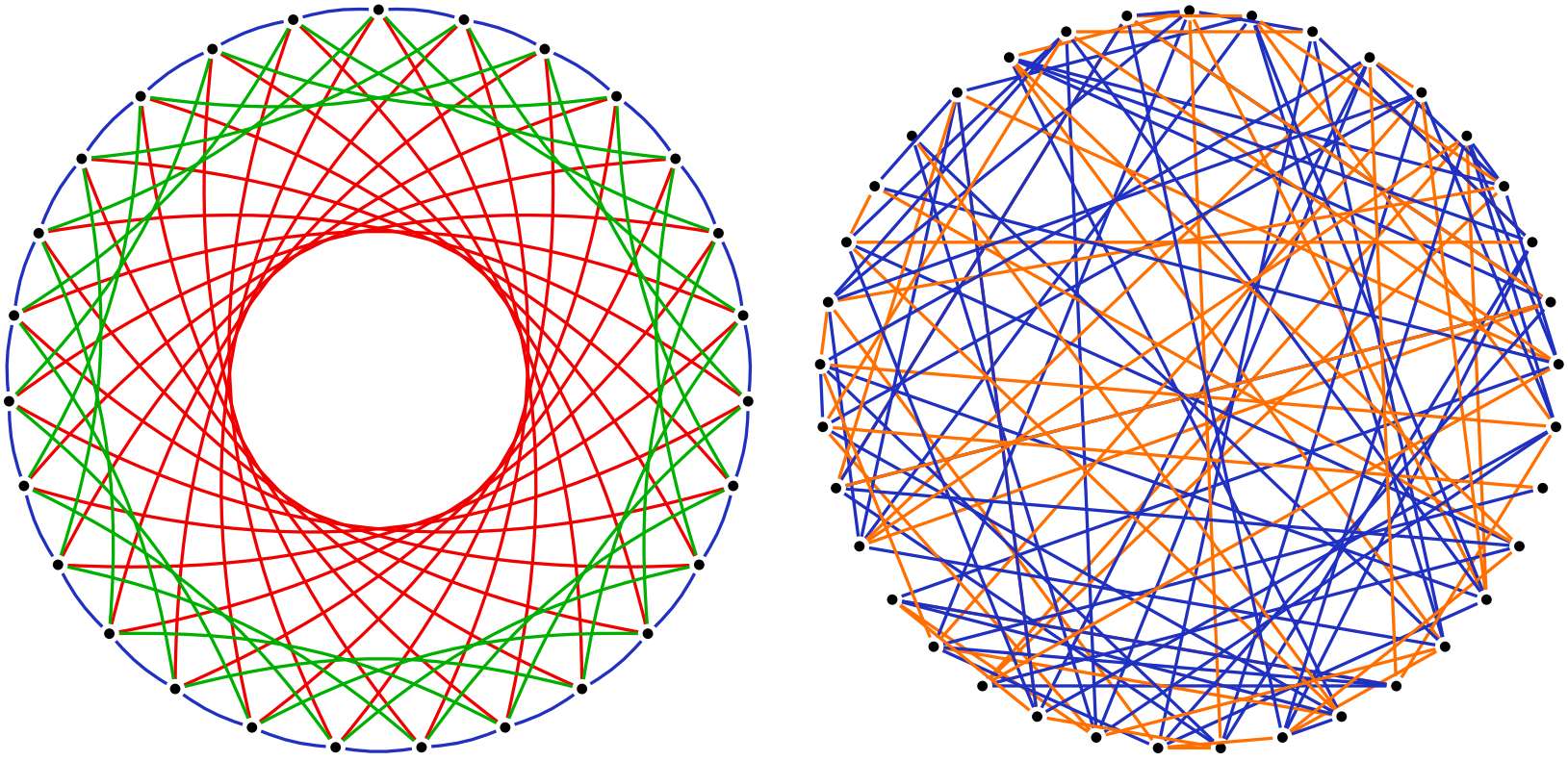
In some cases,

isogeny graphs

can replace *some* DLP-based constructions post-quantumly.

The beauty and the beast

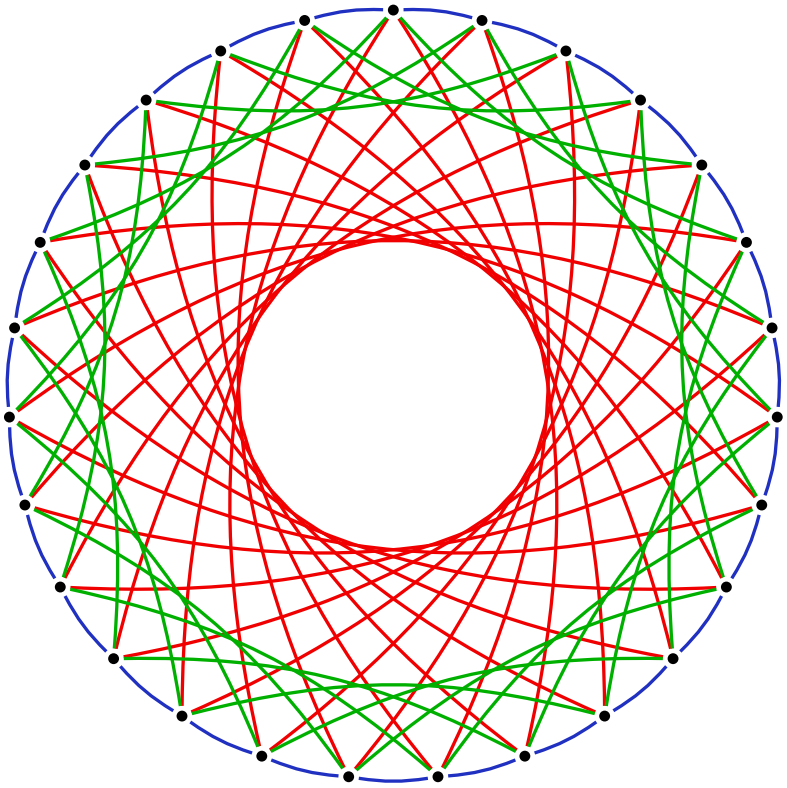
Components of particular isogeny graphs look like this:

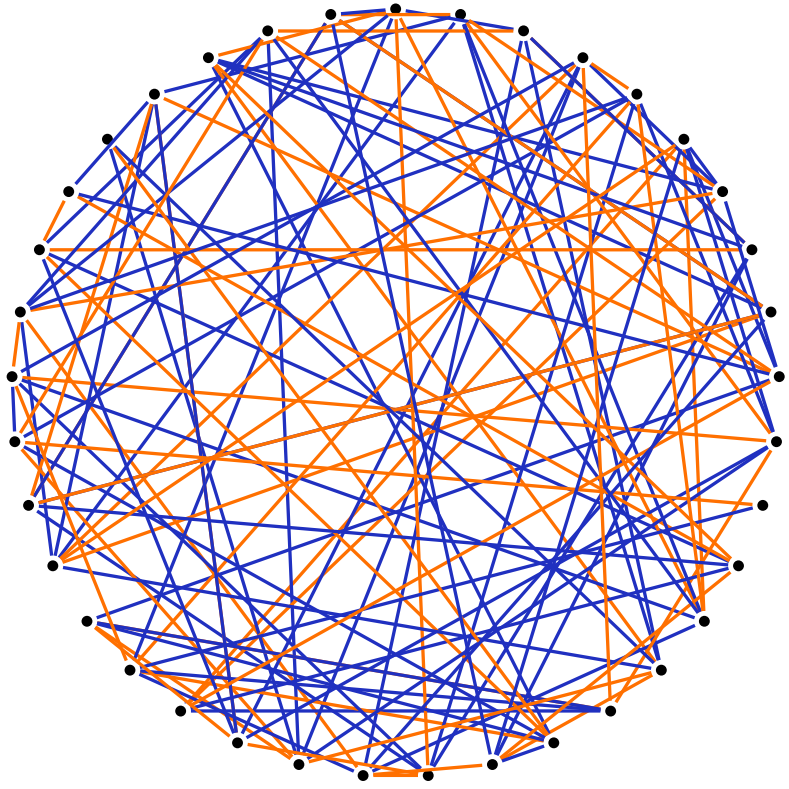


Which of these is good for crypto?

The beauty and the beast

Components of particular isogeny graphs look like this:

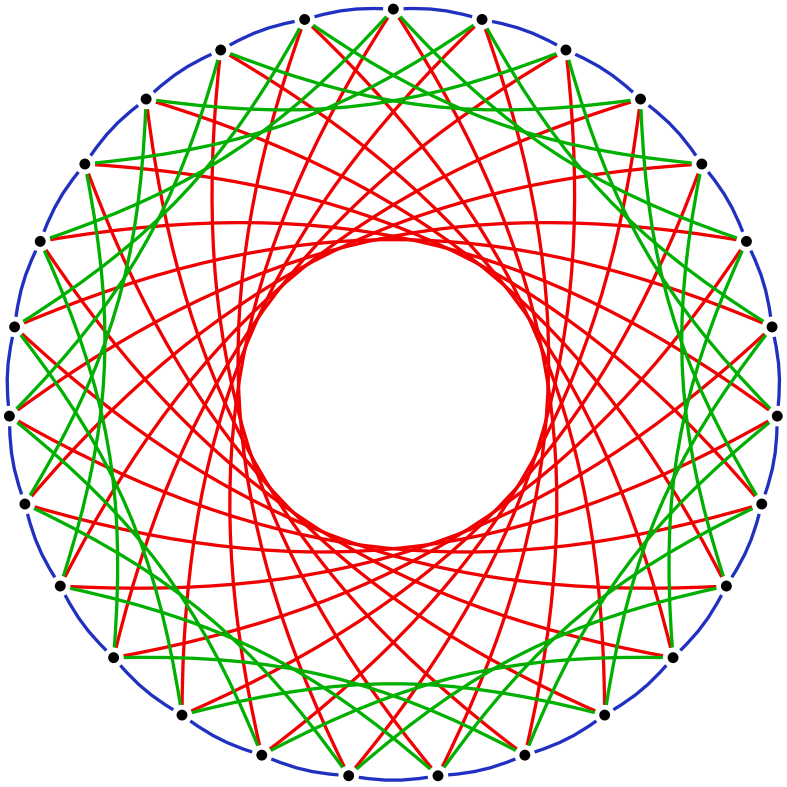




Which of these is good for crypto? **Both.**

The beauty and the beast

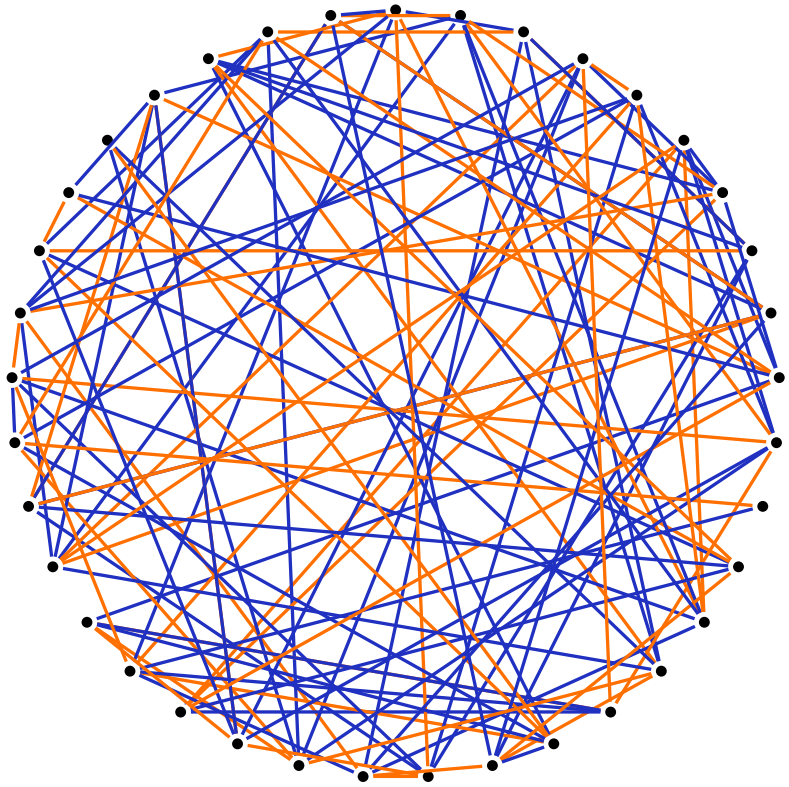
At this time, there are $\geq$ two distinct families of systems:



$\mathbb{F}_p$

CSIDH ['siː,saɪd]

https://csidh.isogeny.org



$\mathbb{F}_{p^2}$

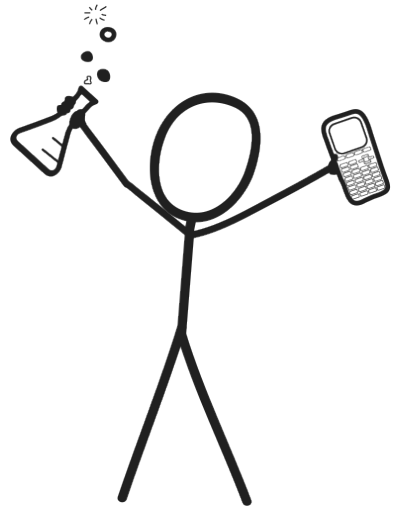
SIDH

https://sike.org

Plan for this lecture

- High-level overview for intuition. ✓
- Elliptic curves & isogenies, algorithms.
- The SIDH/SIKE key-agreement protocol.
- The CSIDH non-interactive key-exchange.

Stand back!



We're going to do math.

Elliptic curves

An elliptic curve over a field F of characteristic $\notin \{2, 3\}$ is* an equation of the form

$$E: \; y^2 = x^3 + ax + b$$

with $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$.

Elliptic curves

An elliptic curve over a field F of characteristic $\notin \{2, 3\}$ is* an equation of the form

$$E: \; y^2 = x^3 + ax + b$$

with $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$.

A point on E is a solution (x, y), or the "fake" point ∞.

Elliptic curves

An elliptic curve over a field F of characteristic $\notin \{2, 3\}$ is* an equation of the form

$$E: \; y^2 = x^3 + ax + b$$

with $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$.

A point on E is a solution (x, y), or the "fake" point ∞.

E is an abelian group: we can "add" points.

Elliptic curves

An elliptic curve over a field F of characteristic $\notin \{2, 3\}$ is* an equation of the form

$$E: \; y^2 = x^3 + ax + b$$

with $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$.

A point on E is a solution (x, y), or the "fake" point ∞.

E is an abelian group: we can "add" points.

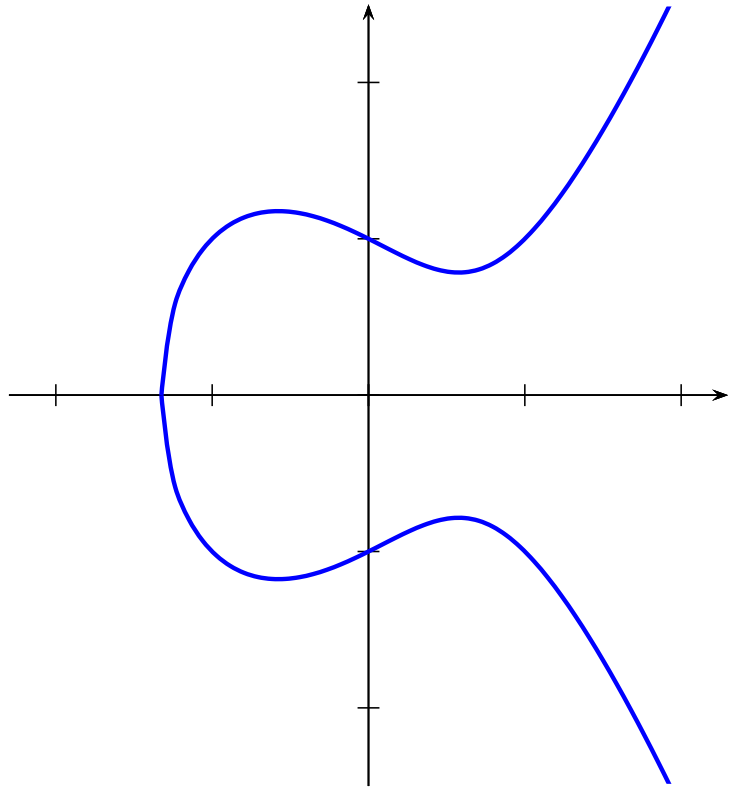
- The neutral element is ∞.
- The inverse of (x, y) is $(x, -y)$.
- The sum of (x_1, y_1) and (x_2, y_2) is

$$\left(\lambda^2 - x_1 - x_2, \; \lambda(2x_1 + x_2 - \lambda^2) - y_1\right)$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $x_1 \neq x_2$ and $\lambda = \frac{3x_1^2 + a}{2y_1}$ otherwise.

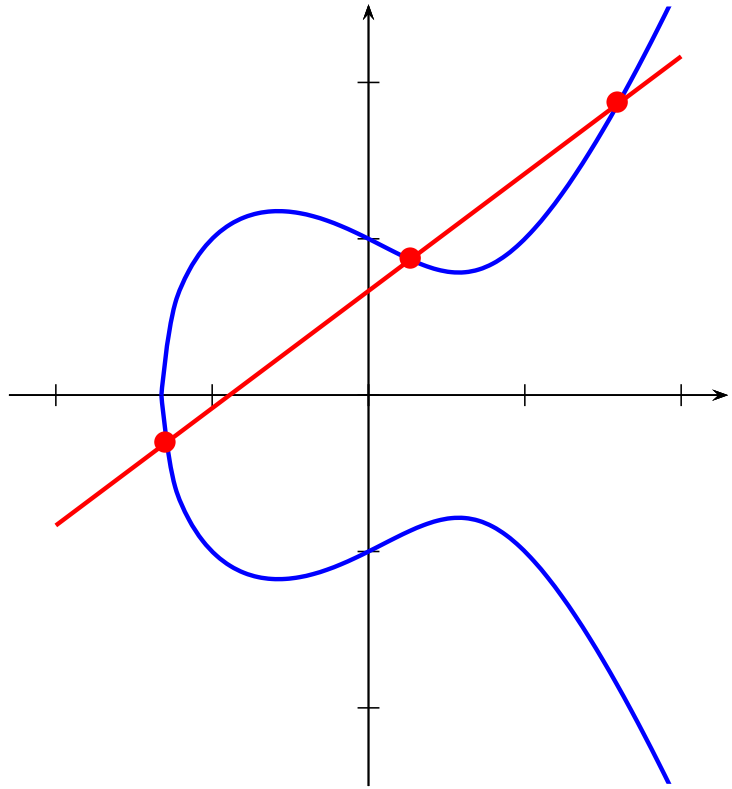
do **not** remember these formulas!

Elliptic curves (picture over $\mathbb{R}$)



The elliptic curve $y^2 = x^3 - x + 1$ over $\mathbb{R}$.

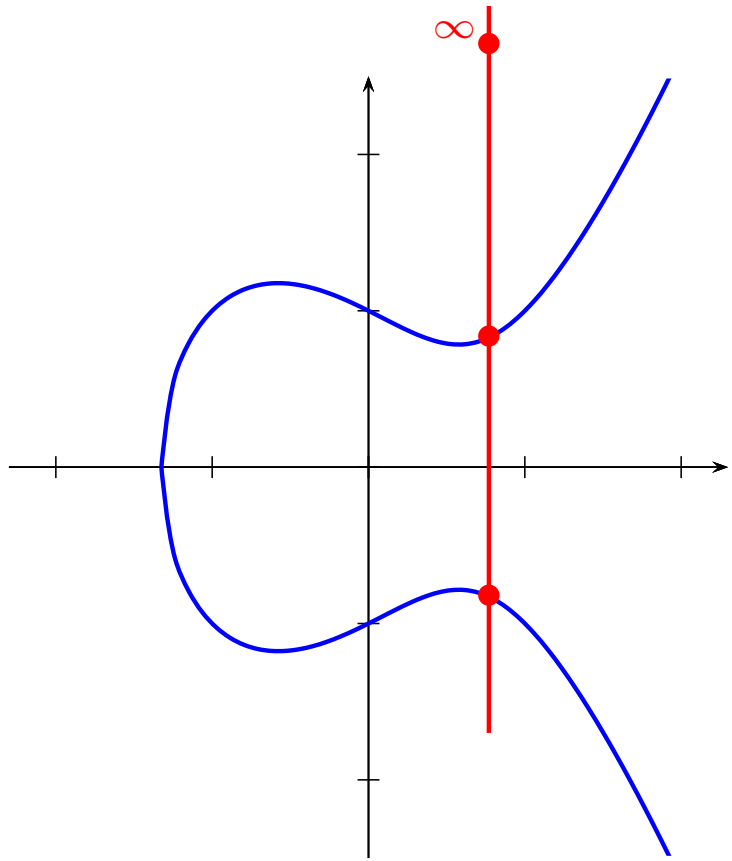
Elliptic curves (picture over $\mathbb{R}$)



Addition law:

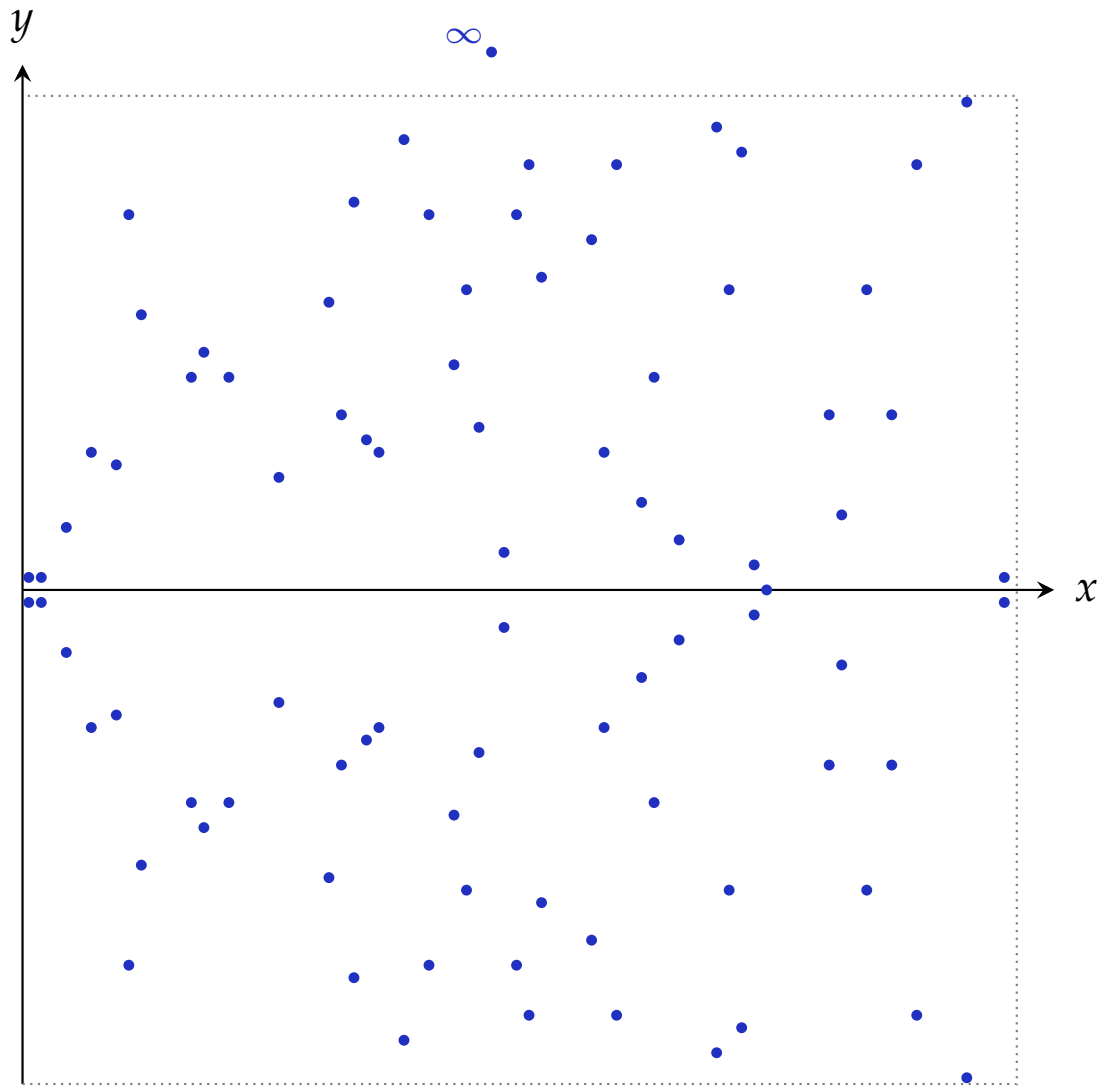
$$P + Q + R = \infty \iff \{P, Q, R\} \text{ on a straight line.}$$

Elliptic curves (picture over $\mathbb{R}$)



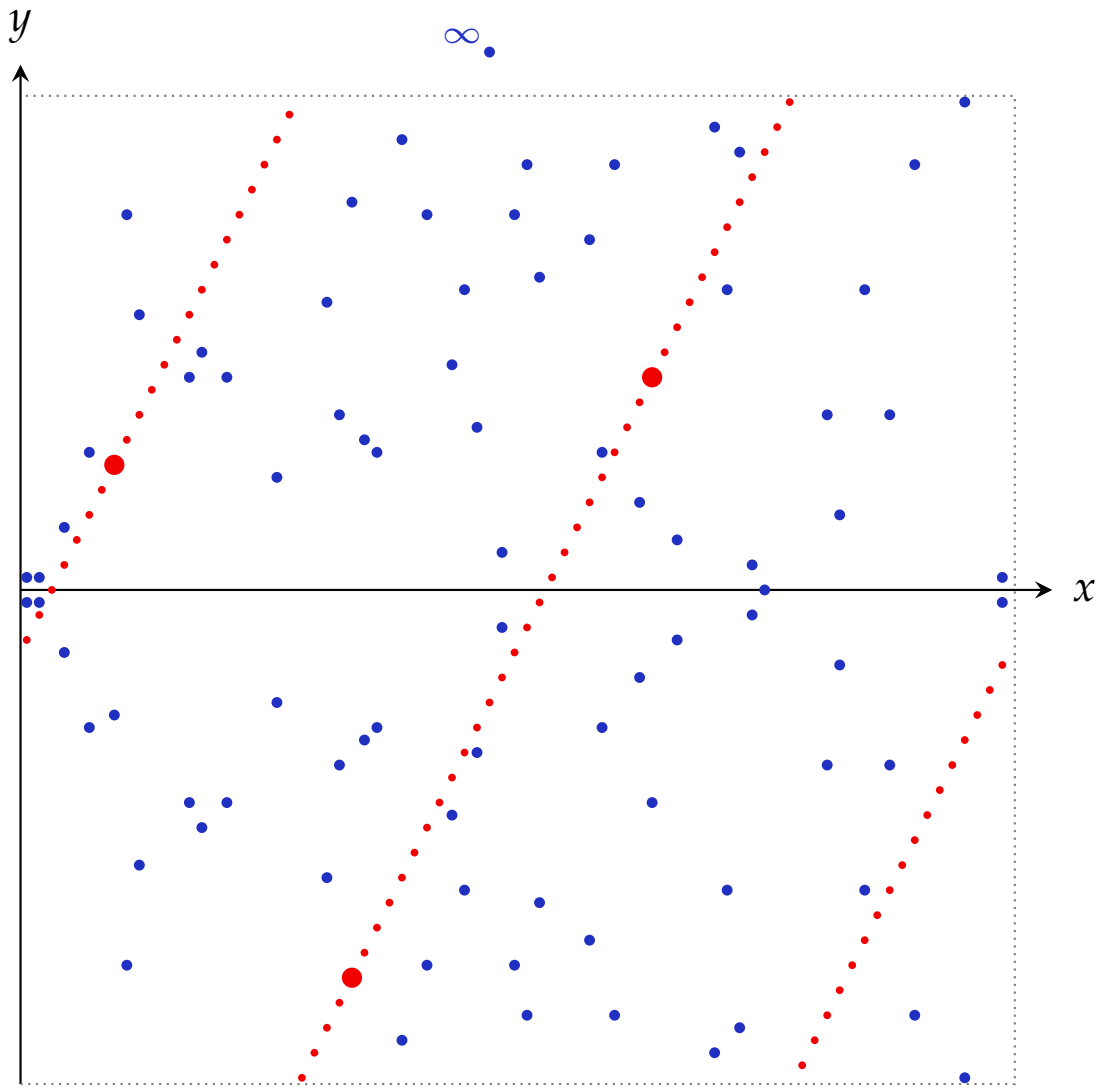
The *point at infinity* ∞ lies on every vertical line.

Elliptic curves (picture over $\mathbb{F}_p$)



The same curve $y^2 = x^3 - x + 1$ over the finite field $\mathbb{F}_{79}$.

Elliptic curves (picture over $\mathbb{F}_p$)



The addition law of $y^2 = x^3 - x + 1$ over the finite field $\mathbb{F}_{79}$.

In Sage:

```
sage: E = EllipticCurve(GF(101), [5,6,7,8,9])
sage: E
Elliptic Curve defined by
            y^2 + 5*x*y + 7*y = x^3 + 6*x^2 + 8*x + 9
            over Finite Field of size 101
```

In Sage:

```
sage: E = EllipticCurve(GF(101), [5,6,7,8,9])
sage: E
Elliptic Curve defined by
            y^2 + 5*x*y + 7*y = x^3 + 6*x^2 + 8*x + 9
            over Finite Field of size 101
sage: P = E(3, 18)        # constructing points
sage: Q = E(8, 75)
sage: P + Q               # point addition
(73 : 24 : 1)
```

In Sage:

```
sage: E = EllipticCurve(GF(101), [5,6,7,8,9])
sage: E
Elliptic Curve defined by
            y^2 + 5*x*y + 7*y = x^3 + 6*x^2 + 8*x + 9
            over Finite Field of size 101
sage: P = E(3, 18)        # constructing points
sage: Q = E(8, 75)
sage: P + Q               # point addition
(73 : 24 : 1)
sage: P - P
(0 : 1 : 0)               # point at infinity
```

ECDH (not post-quantum)

Public parameters:
an elliptic curve E and a point $P \in E$ of large prime order ℓ.

ECDH (not post-quantum)

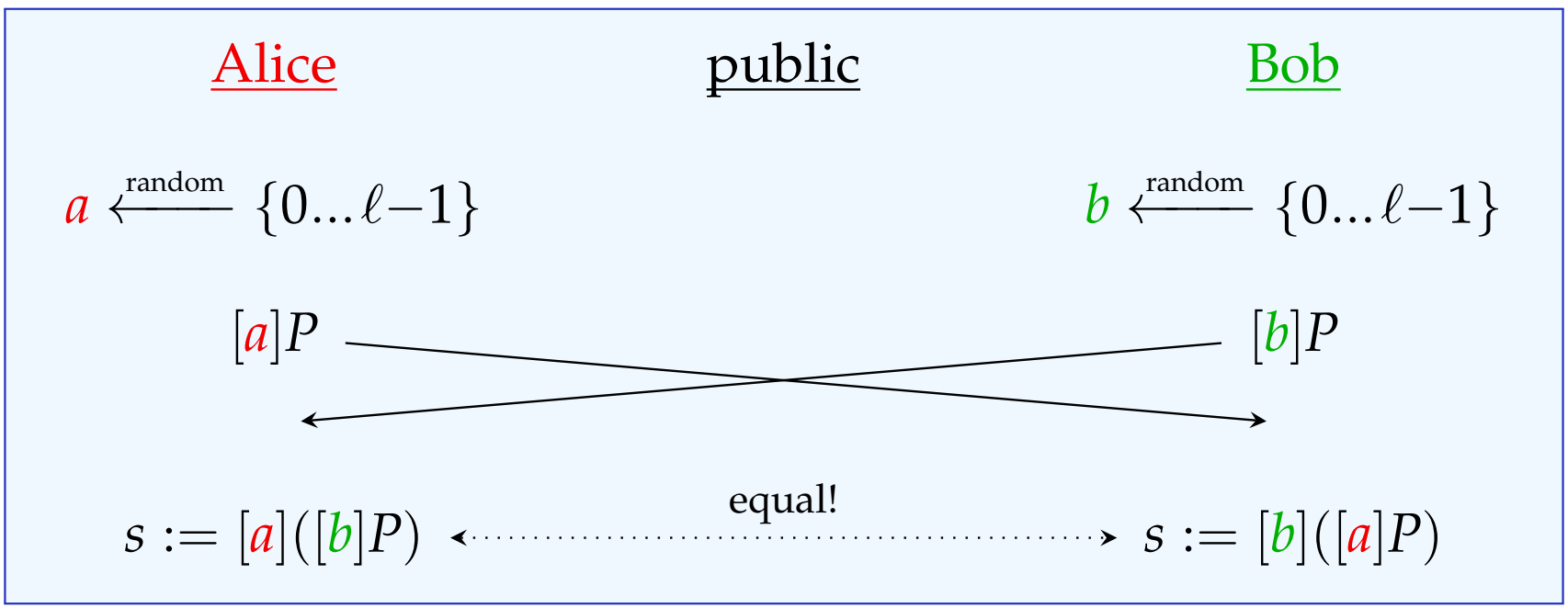
Public parameters:
an elliptic curve E and a point $P \in E$ of large prime order ℓ.

Define scalar multiplication $[n]P := \underbrace{P + \cdots + P}_{n \text{ times}}$. (Use double-and-add!)

ECDH (not post-quantum)

Public parameters:
an elliptic curve E and a point $P \in E$ of large prime order ℓ.

Define scalar multiplication $[n]P := \underbrace{P + \cdots + P}_{n \text{ times}}$. (Use double-and-add!)



Fields of definition

Let k be a field.

An elliptic curve/point/isogeny is defined over k
if the coefficients in its equation/formula lie in k.

We write E/k for "E is defined over k".

Fields of definition

Let k be a field.

An elliptic curve/point/isogeny is defined over k
if the coefficients in its equation/formula lie in k.

We write E/k for "E is defined over k".

For E/k, write $E(k)$ for the set of points of E defined over k.

Note: Simply writing E means $E(\bar{k})$, i.e., points over *all* extension fields.

In Sage:

Everything happens over the specified field of definition:

In Sage:

Everything happens over the specified field of definition:

```
sage: E = EllipticCurve(GF(101), [0,5,0,1,0])
sage: E
Elliptic Curve defined by y^2 = x^3 + 5*x^2 + x
                  over Finite Field of size 101
sage: F.<t> = GF(101^2)
sage: E(11, 69*t + 64)
ValueError: 69*t + 64 is not in the image of #...
sage: EE = E.change_ring(F)
sage: EE(11, 69*t + 64)
(11 : 69*t + 64 : 1)
```

Point counting

There is a polynomial-time algorithm to compute the number of points on an elliptic curve defined over $\mathbb{F}_q$. [Schoof 1985]

Point counting

There is a polynomial-time algorithm to compute the number of points on an elliptic curve defined over $\mathbb{F}_q$. [Schoof 1985]

In Sage:

```
sage: E = EllipticCurve(GF(1000000007), [5,5,5,5,5])
sage: E.count_points()
1000060294
sage: E.order()
1000060294
sage: E.cardinality()
1000060294
```

Isogenies

Isogenies

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Isogenies

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Reminder:

A rational function is $f(x, y)/g(x, y)$ where f, g are polynomials.

A group homomorphism φ satisfies $\varphi(P + Q) = \varphi(P) + \varphi(Q)$.

Isogenies

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Reminder:

A rational function is $f(x, y)/g(x, y)$ where f, g are polynomials.

A group homomorphism φ satisfies $\varphi(P + Q) = \varphi(P) + \varphi(Q)$.

The kernel of an isogeny $\varphi\colon E \to E'$ is $\{P \in E \mid \varphi(P) = \infty\}$.

The degree of a separable* isogeny is the size of its kernel.

Isogenies (examples)

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Isogenies (examples)

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Example #1: $(x, y) \mapsto \left(\frac{x^3-4x^2+30x-12}{(x-2)^2}, \frac{x^3-6x^2-14x+35}{(x-2)^3} \cdot y \right)$ defines a degree-3 isogeny of the elliptic curves

$$\{y^2 = x^3 + x\} \longrightarrow \{y^2 = x^3 - 3x + 3\}$$

over $\mathbb{F}_{71}$. Its kernel is $\{(2, 9), (2, -9), \infty\}$.

Isogenies (examples)

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Example #2: For any a and b, the map $\iota : (x, y) \mapsto (-x, \sqrt{-1} \cdot y)$ defines a degree-1 isogeny of the elliptic curves

$$\{y^2 = x^3 + ax + b\} \longrightarrow \{y^2 = x^3 + ax - b\}.$$

It is an *isomorphism*; its kernel is $\{\infty\}$.

Isogenies (examples)

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Example #3: For each $m \neq 0$, the multiplication-by-m map

$$[m] : E \to E$$

is a degree-m^2 isogeny. If $m \neq 0$ in the base field, its kernel is

$$E[m] \cong \mathbb{Z}/m \times \mathbb{Z}/m.$$

Isogenies (examples)

An isogeny of elliptic curves is a non-zero map $E \to E'$ that is:

- given by rational functions.
- a group homomorphism.

Example #4: For $E/\mathbb{F}_q$ where $q = p^r$ with p prime,

$$\pi : (x, y) \mapsto (x^q, y^q)$$

is a degree-q isogeny, the *Frobenius endomorphism*.

The kernel of $\pi - 1$ is precisely the set of rational points $E(\mathbb{F}_q)$.

In Sage:

```
sage: E = EllipticCurve(GF(101), [1,0])
sage: mu = E.multiplication_by_m_isogeny(5)
```

In Sage:

```
sage: E = EllipticCurve(GF(101), [1,0])
sage: mu = E.multiplication_by_m_isogeny(5)
sage: mu
Isogeny of degree 25
    from Elliptic Curve defined by y^2 = x^3 + x
                    over Finite Field of size 101
    to Elliptic Curve defined by y^2 = x^3 + x
                    over Finite Field of size 101
```

In Sage:

```
sage: E = EllipticCurve(GF(101), [1,0])
sage: mu = E.multiplication_by_m_isogeny(5)
sage: mu
Isogeny of degree 25
    from Elliptic Curve defined by y^2 = x^3 + x
                    over Finite Field of size 101
    to Elliptic Curve defined by y^2 = x^3 + x
                    over Finite Field of size 101
sage: mu.rational_maps()
((-4*x^25 - 4*x^23 - ... + 35*x^5 + 45*x^3 + x)
    / (x^24 + 45*x^22 + ... - 11*x^4 - 4*x^2 - 4),
 (-21*x^36*y + 32*x^34*y + ... - 29*x^2*y - 4*y)
    / (x^36 + 17*x^34 - ... - 44*x^4 + 19*x^2 - 21))
```

The isogeny relation

Isogenies between distinct curves are "rare".
We say E and E' are *isogenous* if there exists an isogeny $E \to E'$.

The isogeny relation

Isogenies between distinct curves are "rare".
We say E and E' are *isogenous* if there exists an isogeny $E \to E'$.

Each isogeny $\varphi\colon E \to E'$ has a unique dual isogeny $\widehat{\varphi}\colon E' \to E$ characterized by $\widehat{\varphi} \circ \varphi = [\deg \varphi]$ and $\varphi \circ \widehat{\varphi} = [\deg \varphi]$.

The isogeny relation

Isogenies between distinct curves are "rare".
We say E and E' are *isogenous* if there exists an isogeny $E \to E'$.

Each isogeny $\varphi\colon E \to E'$ has a unique dual isogeny $\widehat{\varphi}\colon E' \to E$ characterized by $\widehat{\varphi} \circ \varphi = [\deg \varphi]$ and $\varphi \circ \widehat{\varphi} = [\deg \varphi]$.

Tate's theorem:
$E, E'/\mathbb{F}_q$ are isogenous over $\mathbb{F}_q$ if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$.

(Recall that Schoof's algorithm can check this efficiently!)

The isogeny relation

Isogenies between distinct curves are "rare".
We say E and E' are *isogenous* if there exists an isogeny $E \to E'$.

Each isogeny $\varphi\colon E \to E'$ has a unique dual isogeny $\widehat{\varphi}\colon E' \to E$ characterized by $\widehat{\varphi} \circ \varphi = [\deg \varphi]$ and $\varphi \circ \widehat{\varphi} = [\deg \varphi]$.

Tate's theorem:
$E, E'/\mathbb{F}_q$ are isogenous over $\mathbb{F}_q$ if and only if $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$.

(Recall that Schoof's algorithm can check this efficiently!)

$\Longrightarrow$ Bottom line: Being isogenous is an equivalence relation.
Over finite fields, we can easily test it.

Isogeny graphs

Consider a field k and let $S \not\ni \mathrm{char}(k)$ be a set of primes.

The S-isogeny graph over k consists of

- vertices given by elliptic curves over k;
- edges given by ℓ-isogenies, $\ell \in S$, over k;

up to k-isomorphism.

Isogeny graphs

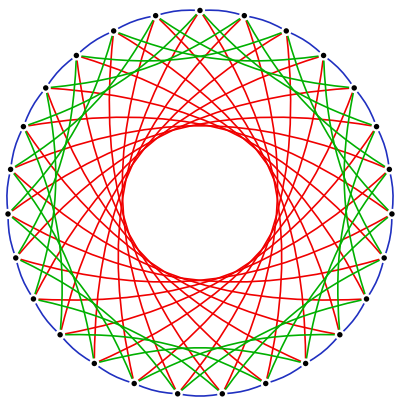
Consider a field k and let $S \not\ni \mathrm{char}(k)$ be a set of primes.

The S-isogeny graph over k consists of

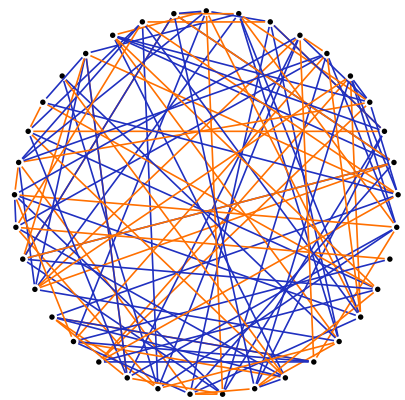
- vertices given by elliptic curves over k;
- edges given by ℓ-isogenies, $\ell \in S$, over k;

up to k-isomorphism.

Example components containing $E\colon y^2 = x^3 + x$:



$k = \mathbb{F}_{419},\ S = \{3, 5, 7\}$



$k = \mathbb{F}_{431^2},\ S = \{2, 3\}.$

Isogenies and kernels

For any finite subgroup G of E, there exists a unique[1] separable* isogeny $\varphi_G : E \to E'$ with kernel G.

The curve E' is denoted by E/G. (cf. quotient groups)

If G is defined over k, then φ_G and E/G are also defined over k.

[1] (up to isomorphism of E')

Computing isogenies: Vélu's formulas (1971)

Let G be a finite subgroup of an elliptic curve E. Then

$$P \mapsto \left(x(P) + \sum_{Q \in G \setminus \{\infty\}} \big(x(P+Q) - x(Q) \big), \right.$$
$$\left. y(P) + \sum_{Q \in G \setminus \{\infty\}} \big(y(P+Q) - y(Q) \big) \right)$$

defines an isogeny of elliptic curves with kernel G.

Computing isogenies: Vélu's formulas (1971)

Let G be a finite subgroup of an elliptic curve E. Then

$$P \mapsto \Bigg(x(P) + \sum_{Q \in G \setminus \{\infty\}} \big(x(P+Q) - x(Q) \big), \\ y(P) + \sum_{Q \in G \setminus \{\infty\}} \big(y(P+Q) - y(Q) \big) \Bigg)$$

defines an isogeny of elliptic curves with kernel G.

This leads to formulas for

- computing the defining equation of E/G;
- evaluating the isogeny $E \to E/G$ at a point.

Computing isogenies: Vélu's formulas (1971)

Let G be a finite subgroup of an elliptic curve E. Then

$$P \mapsto \left(x(P) + \sum_{Q \in G \setminus \{\infty\}} \big(x(P+Q) - x(Q)\big), \right.$$
$$\left. y(P) + \sum_{Q \in G \setminus \{\infty\}} \big(y(P+Q) - y(Q)\big) \right)$$

defines an isogeny of elliptic curves with kernel G.

This leads to formulas for

- computing the defining equation of E/G;
- evaluating the isogeny $E \to E/G$ at a point.

Complexity: $\Theta(\#G) \rightsquigarrow$ only suitable for small degrees.

$\sqrt{\text{élu}}$ [ePrint 2020/341] reduces the cost to $\tilde{\Theta}(\sqrt{\#G})$.

In Sage:

```
sage: E = EllipticCurve(GF(419), [1,0])
sage: E
Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
sage: K = E(80,30)
sage: K.order()
7
```

In Sage:

```
sage: E = EllipticCurve(GF(419), [1,0])
sage: E
Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
sage: K = E(80,30)
sage: K.order()
7
sage: phi = E.isogeny(K)
sage: phi
Isogeny of degree 7
    from Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
    to Elliptic Curve defined by y^2 = x^3 + 285*x + 87
                          over Finite Field of size 419
```

In Sage:

```
sage: E = EllipticCurve(GF(419), [1,0])
sage: E
Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
sage: K = E(80,30)
sage: K.order()
7
sage: phi = E.isogeny(K)
sage: phi
Isogeny of degree 7
    from Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
    to Elliptic Curve defined by y^2 = x^3 + 285*x + 87
                          over Finite Field of size 419
sage: phi(K)
(0 : 1 : 0)      # φ(K) = ∞  ⟹  K lies in the kernel
```

In Sage:

```
sage: E = EllipticCurve(GF(419), [1,0])
sage: E
Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
sage: K = E(80,30)
sage: K.order()
7
sage: phi = E.isogeny(K)
sage: phi
Isogeny of degree 7
    from Elliptic Curve defined by y^2 = x^3 + x
                          over Finite Field of size 419
    to Elliptic Curve defined by y^2 = x^3 + 285*x + 87
                          over Finite Field of size 419
sage: phi(K)
(0 : 1 : 0)      # φ(K) = ∞  ⟹  K lies in the kernel
sage: phi.rational_maps()
((x^7 + 129*x^6 - ... + 25)/(x^6 + 129*x^5 - ... + 36),
 (x^9*y - 16*x^8*y - ... + 70*y)/(x^9 - 16*x^8 + ...))
```

Predictable groups: Supersingular curves

Vélu operates in the field where the points in G live.

$\rightsquigarrow$ We need to make sure extensions stay small for desired $\#G$.

Predictable groups: Supersingular curves

Vélu operates in the field where the points in G live.

$\rightsquigarrow$ We need to make sure extensions stay small for desired $\#G$.

Let $p \geq 5$ be prime.

- $E/\mathbb{F}_p$ is *supersingular* if and only if $\#E(\mathbb{F}_p) = p+1$.
- In that case, $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(p+1) \times \mathbb{Z}/(p+1)$.

Predictable groups: Supersingular curves

Vélu operates in the field where the points in G live.

$\rightsquigarrow$ We need to make sure extensions stay small for desired $\#G$.

Let $p \geq 5$ be prime.

- $E/\mathbb{F}_p$ is *supersingular* if and only if $\#E(\mathbb{F}_p) = p+1$.
- In that case, $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(p+1) \times \mathbb{Z}/(p+1)$.

$\rightsquigarrow$ Easy method to control the group structure by choosing p!

$\rightsquigarrow$ Cryptography works well using supersingular curves.

In Sage:

```
sage: E = EllipticCurve(GF(62207), [1,0])
sage: E.is_supersingular()
True
sage: E.order()
62208
sage: E.order().factor()
2^8 * 3^5
```

In Sage:

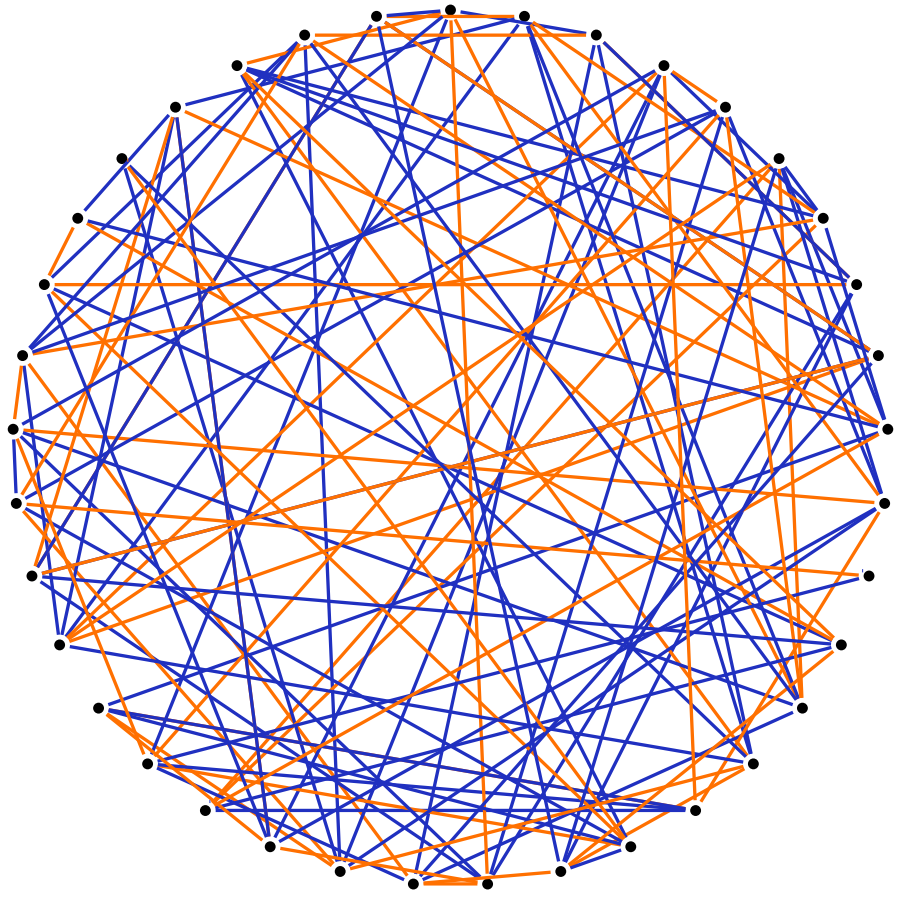
```
sage: E = EllipticCurve(GF(62207), [1,0])
sage: E.is_supersingular()
True
sage: E.order()
62208
sage: E.order().factor()
2^8 * 3^5
sage: EE = E.change_ring(GF(62207^2))
sage: EE.order().factor()
2^16 * 3^10
```

In Sage:

```
sage: E = EllipticCurve(GF(62207), [1,0])
sage: E.is_supersingular()
True
sage: E.order()
62208
sage: E.order().factor()
2^8 * 3^5
sage: EE = E.change_ring(GF(62207^2))
sage: EE.order().factor()
2^16 * 3^10
sage: EE.abelian_group()
Additive abelian group isomorphic to Z/62208 + Z/62208
        embedded in Abelian group of points
        on Elliptic Curve defined by y^2 = x^3 + x
        over Finite Field in z2 of size 62207^2
```

Plan for this lecture

- High-level overview for intuition. ✓
- Elliptic curves & isogenies, algorithms. ✓
- The SIDH/SIKE key-agreement protocol.
- The CSIDH non-interactive key-exchange.



Now: SIDH (Jao, De Feo; 2011)

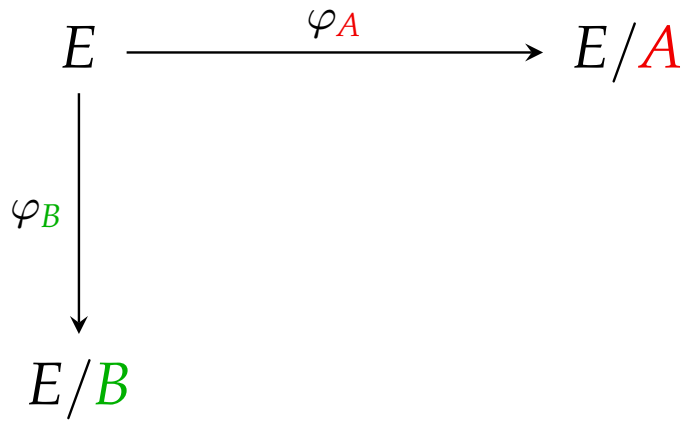
SIDH: High-level view

E

SIDH: High-level view

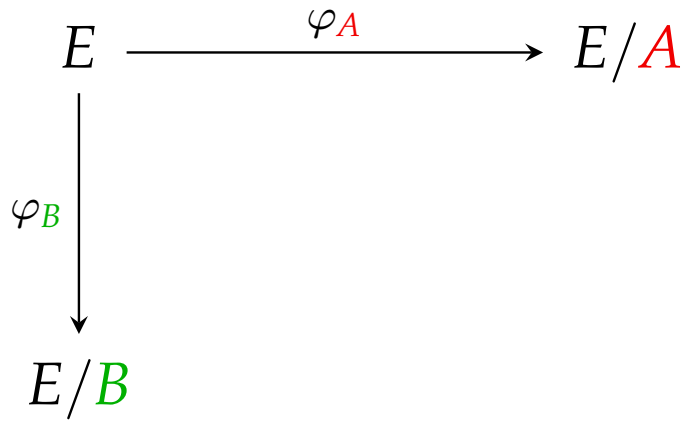
- Alice & Bob pick secret subgroups A and B of E.

SIDH: High-level view



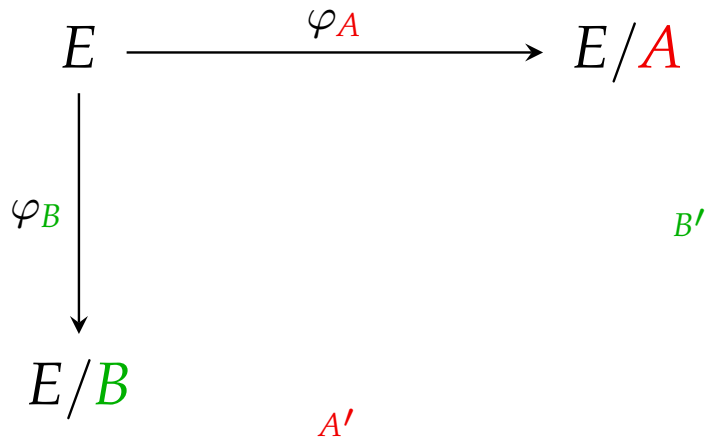
- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
  (These isogenies correspond to walking on the isogeny graph.)

SIDH: High-level view



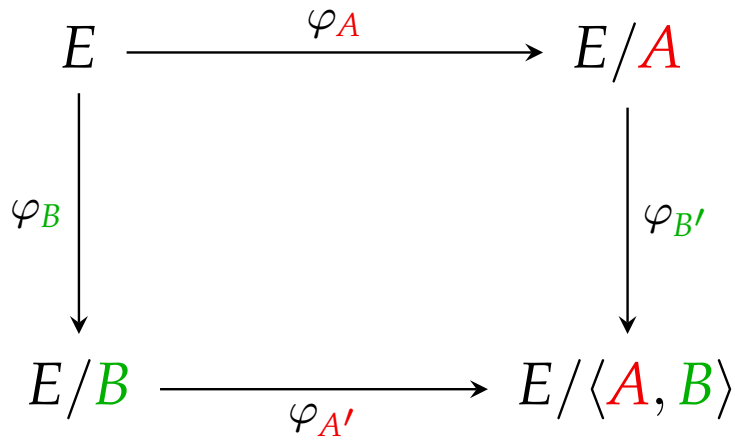
- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
  (These isogenies correspond to walking on the isogeny graph.)
- Alice and Bob transmit the values E/A and E/B.

SIDH: High-level view



- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
  (These isogenies correspond to walking on the isogeny graph.)
- Alice and Bob transmit the values E/A and E/B.
- Alice somehow obtains $A' := \varphi_B(A)$. (Similar for Bob.)

SIDH: High-level view



- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
  (These isogenies correspond to walking on the isogeny graph.)
- Alice and Bob transmit the values E/A and E/B.
- Alice somehow obtains $A' := \varphi_B(A)$. (Similar for Bob.)
- They both compute the shared secret
  $$(E/B)/A' \cong E/\langle A, B\rangle \cong (E/A)/B'.$$

SIDH's auxiliary points

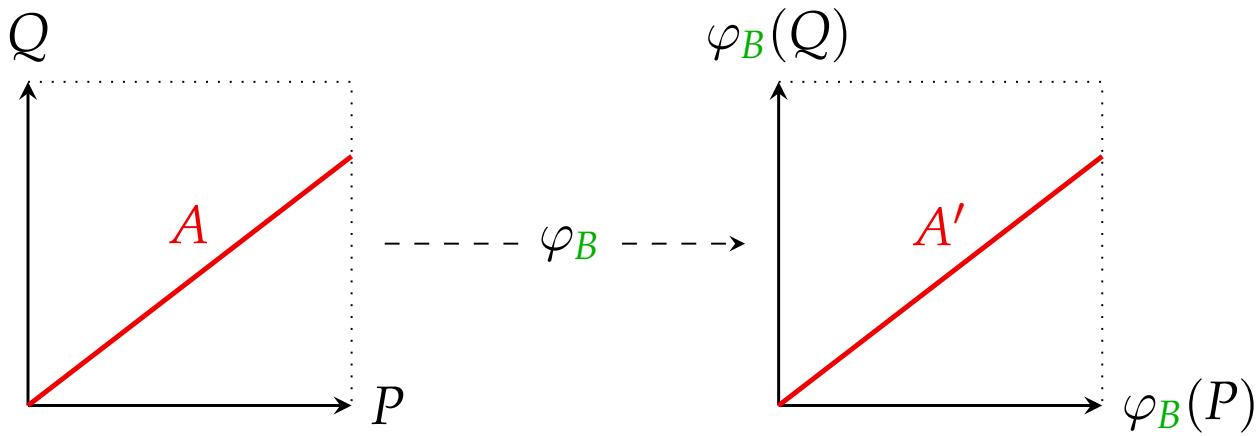
"Alice somehow obtains $A' := \varphi_B(A)$."

...but Alice knows only A, Bob knows only φ_B. Hmm.

SIDH's auxiliary points

"Alice somehow obtains $A' := \varphi_B(A)$."
...but Alice knows only A, Bob knows only φ_B. Hmm.

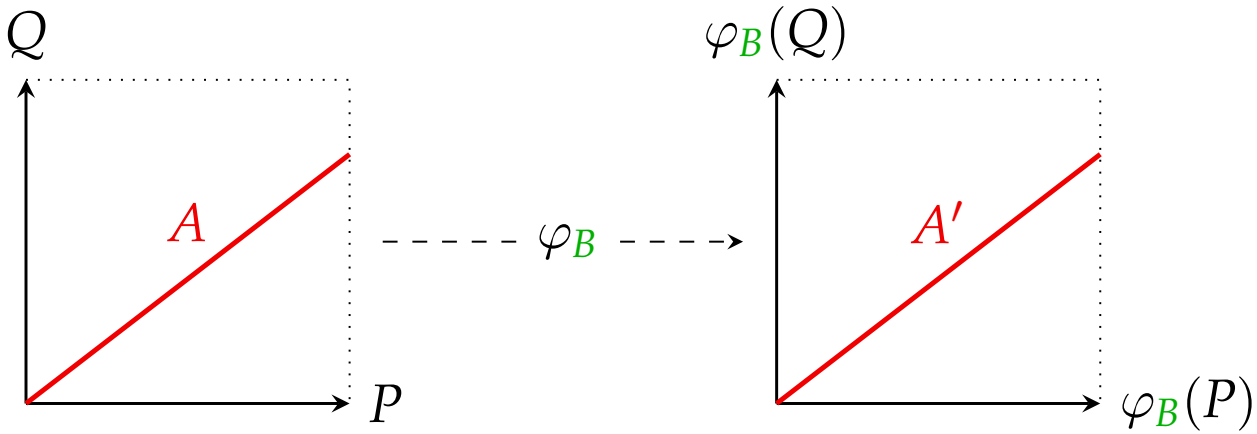
SIDH's solution: φ_B is a group homomorphism!



SIDH's auxiliary points

"Alice somehow obtains $A' := \varphi_B(A)$."
...but Alice knows only A, Bob knows only φ_B. Hmm.

SIDH's solution: φ_B is a group homomorphism! (and $A \cap B = \{\infty\}$)



- Alice picks A as $\langle P + [a]Q \rangle$ for fixed public $P, Q \in E$.
- Bob includes $\varphi_B(P)$ and $\varphi_B(Q)$ in his public key.

$\Longrightarrow$ Now Alice can compute A' as $\langle \varphi_B(P) + [a]\varphi_B(Q) \rangle$.

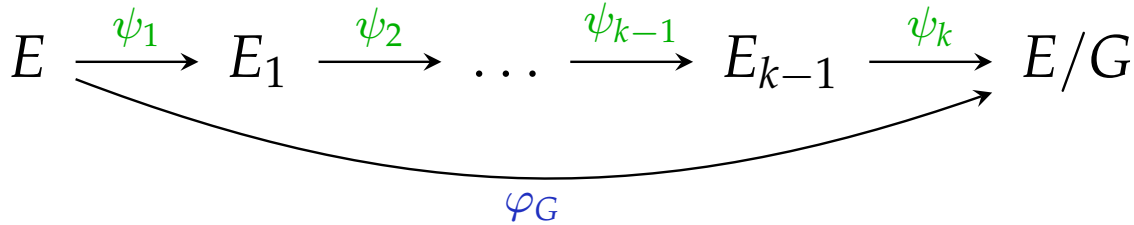
Decomposing smooth isogenies

- In SIDH, $\#A$ and $\#B$ are "crypto-sized".
  Vélu's formulas take $\Theta(\#G)$ to compute $\varphi_G \colon E \to E/G$.

Decomposing smooth isogenies

- In SIDH, $\#A = 2^n$ and $\#B = 3^m$ are "crypto-sized".
  Vélu's formulas take $\Theta(\#G)$ to compute $\varphi_G \colon E \to E/G$.

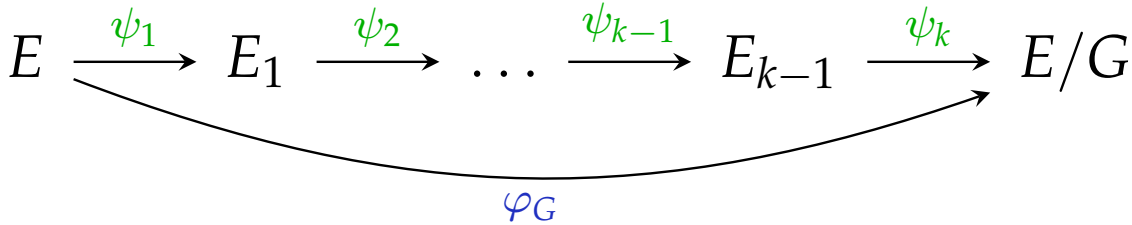
!! Evaluate φ_G as a chain of small-degree isogenies:
For $G \cong \mathbb{Z}/\ell^k$, set $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.



Decomposing smooth isogenies

- In SIDH, $\#A = 2^n$ and $\#B = 3^m$ are "crypto-sized".
  Vélu's formulas take $\Theta(\#G)$ to compute $\varphi_G \colon E \to E/G$.

!! Evaluate φ_G as a chain of small-degree isogenies:
For $G \cong \mathbb{Z}/\ell^k$, set $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.

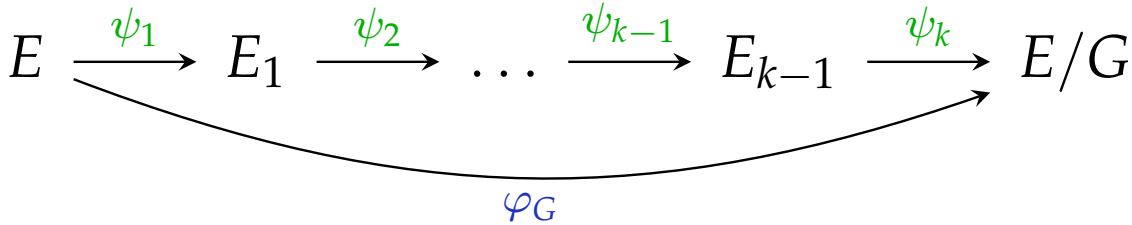


$\rightsquigarrow$ Complexity: $O(k^2 \cdot \ell)$.
Exponentially smaller than a ℓ^k-isogeny!

Decomposing smooth isogenies

- In SIDH, $\#A = 2^n$ and $\#B = 3^m$ are "crypto-sized".
  Vélu's formulas take $\Theta(\#G)$ to compute $\varphi_G \colon E \to E/G$.

!! Evaluate φ_G as a chain of small-degree isogenies:
For $G \cong \mathbb{Z}/\ell^k$, set $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.



$\rightsquigarrow$ Complexity: $O(k^2 \cdot \ell)$.
Exponentially smaller than a ℓ^k-isogeny!

- Graph view: Each ψ_i is a step in the ℓ-isogeny graph.

SIDH in one slide

Public parameters:

- a large prime $p = 2^n 3^m - 1$ and a supersingular $E/\mathbb{F}_p$
- bases (P, Q) of $E[2^n]$ and (R, S) of $E[3^m]$ (recall $E[k] \cong \mathbb{Z}/k \times \mathbb{Z}/k$)

Alice	public	Bob
$a \xleftarrow{\text{random}} \{0 \dots 2^n - 1\}$		$b \xleftarrow{\text{random}} \{0 \dots 3^m - 1\}$
$A := \langle P + [a]Q \rangle$		$B := \langle R + [b]S \rangle$
compute $\varphi_A : E \to E/A$		compute $\varphi_B : E \to E/B$
$E/A,\ \varphi_A(R),\ \varphi_A(S)$	$\longrightarrow$	
	$\longleftarrow$	$E/B,\ \varphi_B(P),\ \varphi_B(Q)$
$A' := \langle \varphi_B(P) + [a]\varphi_B(Q) \rangle$		$B' := \langle \varphi_A(R) + [b]\varphi_A(S) \rangle$
$s := j\big((E/B)/A'\big)$		$s := j\big((E/A)/B'\big)$

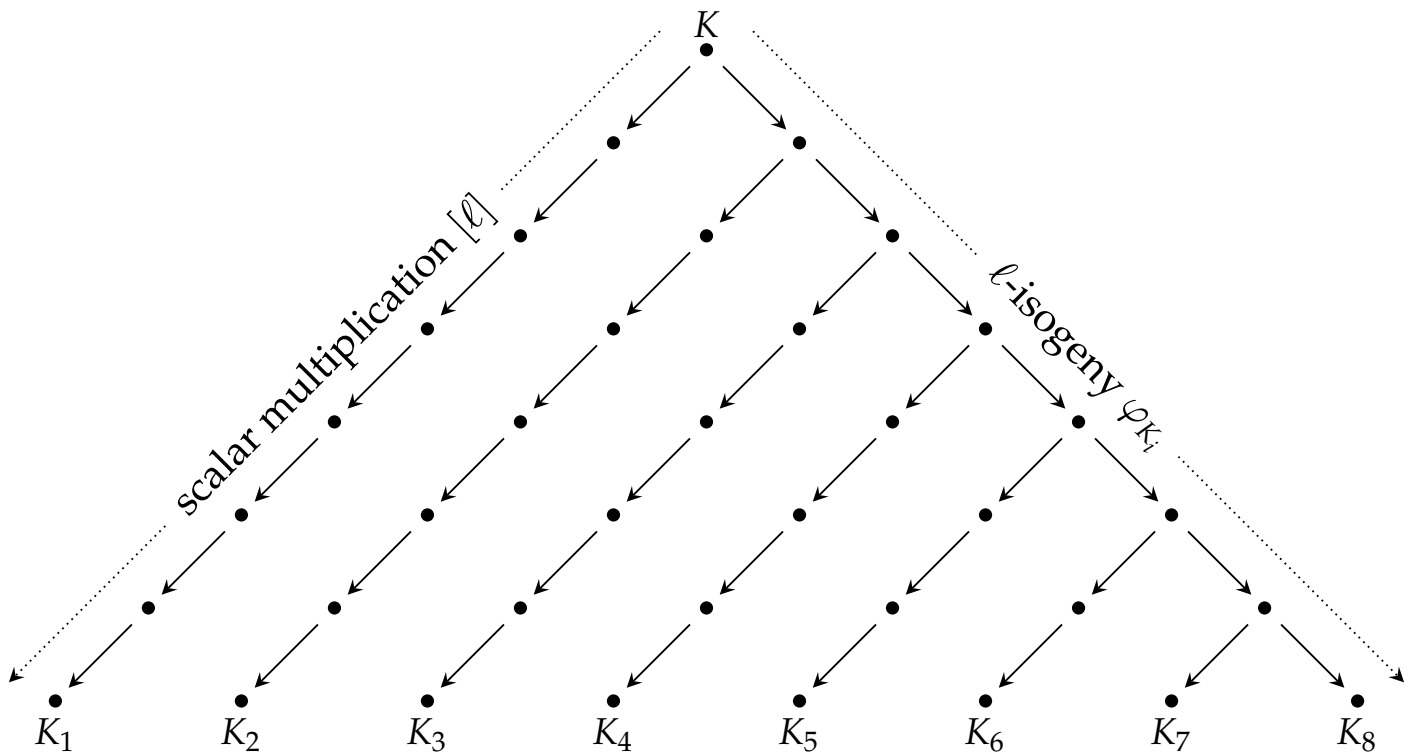
Strategies for SIDH

Recall: SIDH splits ℓ^k-isogenies into k individual ℓ-isogenies.
This requires computing $K_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(K)$ for all i.

Strategies for SIDH

Recall: SIDH splits ℓ^k-isogenies into k individual ℓ-isogenies.
This requires computing $K_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(K)$ for all i.

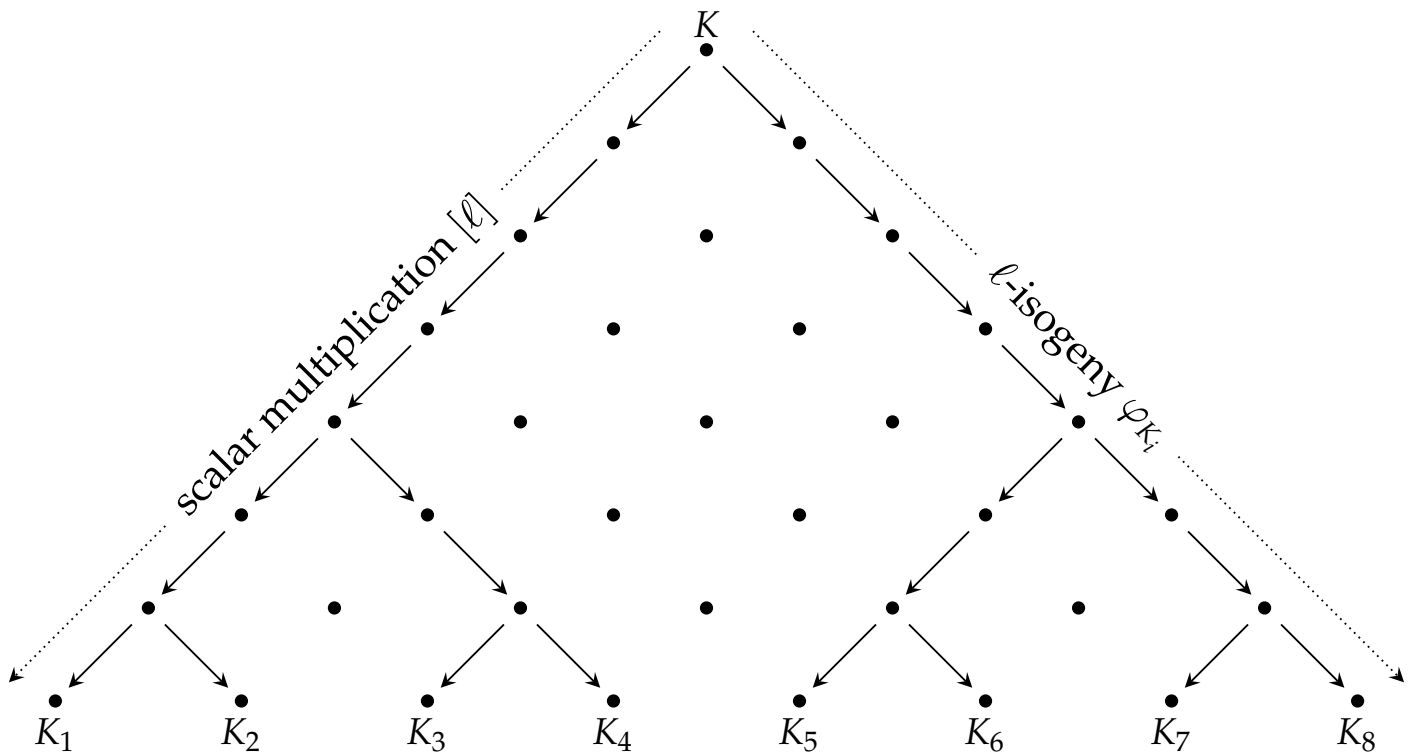
Naïve strategy:



Strategies for SIDH

Recall: SIDH splits ℓ^k-isogenies into k individual ℓ-isogenies.
This requires computing $K_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(K)$ for all i.

Sparse strategy:



Optimal strategies for SIDH

$\Longrightarrow$ Sparse strategy improves $O(k^2 \cdot \ell)$ to $O(k \log k \cdot \ell)$.

Optimal strategies for SIDH

$\Longrightarrow$ Sparse strategy improves $O(k^2 \cdot \ell)$ to $O(k \log k \cdot \ell)$.

When the costs of $[\ell]$ and φ_{K_i} are imbalanced, other trees can be even more efficient. They can be constructed easily.

$\rightsquigarrow$ *"optimal strategies"*

In Sage:

Factored isogenies were implemented recently (version ≥ 9.5):

```
sage: E = EllipticCurve(GF(2^127-1), [1,0])
sage: K = E(23, 40490046516039691075571867486180936666)
sage: K.order()
106338239662793269832304564822427566 08
sage: K.order().factor()
2^123
```

In Sage:

Factored isogenies were implemented recently (version ≥ 9.5):

```
sage: E = EllipticCurve(GF(2^127-1), [1,0])
sage: K = E(23, 40490046516039691075571867486180936666)
sage: K.order()
106338239662793269832304564822427566 08
sage: K.order().factor()
2^123
sage: phi = E.isogeny(K, algorithm="factored")
sage: phi
Composite morphism of degree 1063...6608 = 2^123:
  From: Elliptic Curve defined by y^2 = x^3 + x
                over Finite Field of size 1701...5727
  To:   Elliptic Curve defined by
                y^2 = x^3 + 1625...8575*x + 1200...7360
                over Finite Field of size 1701...5727
```

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Classical attacks:

- Meet-in-the-middle: $\tilde{\mathcal{O}}(p^{1/4})$ time *& space (!)*.

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Classical attacks:

- Meet-in-the-middle: $\tilde{\mathcal{O}}(p^{1/4})$ time *& space (!)*.
- Collision finding: $\tilde{\mathcal{O}}(p^{3/8}/\sqrt{memory}/cores)$.

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Classical attacks:

- Meet-in-the-middle: $\tilde{\mathcal{O}}(p^{1/4})$ time *& space (!)*.
- Collision finding: $\tilde{\mathcal{O}}(p^{3/8}/\sqrt{memory}/cores)$.

Quantum attacks:

- Claw finding: claimed $\tilde{\mathcal{O}}(p^{1/6})$.

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Classical attacks:

- Meet-in-the-middle: $\tilde{\mathcal{O}}(p^{1/4})$ time *& space (!)*.
- Collision finding: $\tilde{\mathcal{O}}(p^{3/8}/\sqrt{\mathit{memory}}/\mathit{cores})$.

Quantum attacks:

- Claw finding: claimed $\tilde{\mathcal{O}}(p^{1/6})$.

  [ePrint 2019/103]: *more expensive than classical attacks*.

  "An adversary with enough quantum memory to run [quantum claw finding] with the query-optimal parameters could break SIKE faster by using the classical control hardware to run [collision finding]."

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Alice & Bob can choose from about $\sqrt{p}$ secret keys each.

Classical attacks:

- Meet-in-the-middle: $\tilde{\mathcal{O}}(p^{1/4})$ time *& space (!)*.
- Collision finding: $\tilde{\mathcal{O}}(p^{3/8}/\sqrt{\mathit{memory}}/\mathit{cores})$.

Quantum attacks:

- Claw finding: claimed $\tilde{\mathcal{O}}(p^{1/6})$.

  [ePrint 2019/103]: *more expensive than classical attacks*.

  "An adversary with enough quantum memory to run [quantum claw finding] with the query-optimal parameters could break SIKE faster by using the classical control hardware to run [collision finding]."

Bottom line: Fully exponential. Complexity $\exp\left((\log p)^{1+o(1)}\right)$.

Security of SIDH: The $IKE challenges

In 2021, a well-known large technology company announced challenge instances of SIDH as a target for cryptanalysis:

Security of SIDH: The $IKE challenges

In 2021, a well-known large technology company announced challenge instances of SIDH as a target for cryptanalysis:

- **$IKEp182:** $p \approx 2^{181.3}$ / prize $\underline{5 \cdot 10^3 \text{ USD}}$
  Solved using meet-in-the-middle. [ePrint 2021/1421]

Security of SIDH: The $IKE challenges

In 2021, a well-known large technology company announced challenge instances of SIDH as a target for cryptanalysis:

- **$IKEp182:** $p \approx 2^{181.3}$ / prize $5 \cdot 10^3$ USD
  Solved using meet-in-the-middle. [ePrint 2021/1421]

- **$IKEp217:** $p \approx 2^{216.2}$ / prize $5 \cdot 10^4$ USD
  Still open. (Good luck!)

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice. She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice. She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.
- Bob cheats and sends $Q'' := Q' + [2^{n-1}]P'$ instead of Q'. Alice computes $A'' = \langle P' + [a]Q'' \rangle$.

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice. She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.

- Bob cheats and sends $Q'' := Q' + [2^{n-1}]P'$ instead of Q'. Alice computes $A'' = \langle P' + [a]Q'' \rangle$.

  If $a = 2u$: $[a]Q'' = [a]Q' + [u][2^n]P' = [a]Q'$.
  If $a = 2u+1$: $[a]Q'' = [a]Q' + [u][2^n]P' + [2^{n-1}]P' = [a]Q' + [2^{n-1}]P'$.

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice. She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.

- Bob cheats and sends $Q'' := Q' + [2^{n-1}]P'$ instead of Q'. Alice computes $A'' = \langle P' + [a]Q'' \rangle$.

  If $a = 2u$: $[a]Q'' = [a]Q' + [u][2^n]P' = [a]Q'$.
  If $a = 2u+1$: $[a]Q'' = [a]Q' + [u][2^n]P' + [2^{n-1}]P' = [a]Q' + [2^{n-1}]P'$.

  $\Longrightarrow$ Bob learns the LSB of a.

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice.
  She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.

- Bob cheats and sends $Q'' := Q' + [2^{n-1}]P'$ instead of Q'.
  Alice computes $A'' = \langle P' + [a]Q'' \rangle$.

  If $a = 2u$: $[a]Q'' = [a]Q' + [u][2^n]P' = [a]Q'$.
  If $a = 2u+1$: $[a]Q'' = [a]Q' + [u][2^n]P' + [2^{n-1}]P' = [a]Q' + [2^{n-1}]P'$.

  $\Longrightarrow$ Bob learns the LSB of a.
  Similarly, he can completely recover a in $O(n)$ queries.

An *active* attack on SIDH key reuse

- Recall: Bob sends $P' := \varphi_B(P)$ and $Q' := \varphi_B(Q)$ to Alice.
  She computes $A' = \langle P' + [a]Q' \rangle$ and, from that, obtains s.

- Bob cheats and sends $Q'' := Q' + [2^{n-1}]P'$ instead of Q'.
  Alice computes $A'' = \langle P' + [a]Q'' \rangle$.

  If $a = 2u$: $[a]Q'' = [a]Q' + [u][2^n]P' = [a]Q'$.
  If $a = 2u+1$: $[a]Q'' = [a]Q' + [u][2^n]P' + [2^{n-1}]P' = [a]Q' + [2^{n-1}]P'$.

  $\Longrightarrow$ Bob learns the LSB of a.
  Similarly, he can completely recover a in $O(n)$ queries.

Validating that Bob is honest is $\approx$ as hard as breaking SIDH.

$\Longrightarrow$ only usable with ephemeral keys or as a KEM *"SIKE"*.

SIKE

Fix for the active attack on SIDH [Fujisaki–Okamoto 1999]:

SIKE

Fix for the active attack on SIDH [Fujisaki–Okamoto 1999]:

- The first thing Bob sends encrypted is *his private key*.

SIKE

Fix for the active attack on SIDH [Fujisaki–Okamoto 1999]:

- The first thing Bob sends encrypted is *his private key*.
- Alice can then recompute Bob's public key.
  If it does not match what Bob sent, she aborts.

SIKE

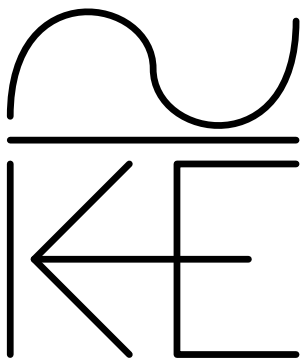
Fix for the active attack on SIDH [Fujisaki–Okamoto 1999]:

- The first thing Bob sends encrypted is *his private key*.
- Alice can then recompute Bob's public key.
  If it does not match what Bob sent, she aborts.
- Result: *One* party (Alice) can reuse her key. (Bob can't!)

SIKE

Fix for the active attack on SIDH [Fujisaki–Okamoto 1999]:

- The first thing Bob sends encrypted is *his private key*.
- Alice can then recompute Bob's public key.
  If it does not match what Bob sent, she aborts.
- Result: *One* party (Alice) can reuse her key. (Bob can't!)



This is (the essence of) **SIKE**, a KEM submitted to NIST's standardization project.

See `https://sike.org`.

Some numbers for SIKE

The NIST submission contains four parameter sets:
{SIKEp434, SIKEp503, SIKEp610, SIKEp751}

Some numbers for SIKE

The NIST submission contains four parameter sets:
{SIKEp434, SIKEp503, SIKEp610, SIKEp751}

Sizes:

- Public keys: 330–564 bytes.
- Ciphertexts: 346–596 bytes.
- Secret keys: 374–644 bytes.

Some numbers for SIKE

The NIST submission contains four parameter sets:
{SIKEp434, SIKEp503, SIKEp610, SIKEp751}

Sizes:

- Public keys: 330–564 bytes.
- Ciphertexts: 346–596 bytes.
- Secret keys: 374–644 bytes.

Speed (Skylake)**:**

- Key generation: 5–25 *million* cycles.
- Encapsulation: 10–40 *million* cycles.
- Decapsulation: 10–44 *million* cycles.

Plan for this lecture

- High-level overview for intuition. ✓
- Elliptic curves & isogenies, algorithms. ✓
- The SIDH/SIKE key-agreement protocol. ✓
- The CSIDH non-interactive key-exchange.


CSIDH ['si:ˌsaɪd]
[Castryck–Lange–Martindale–Panny–Renes 2018]

A different way to kill the active attack?

- Recall: SIDH cannot reuse keys because of the auxiliary points $\varphi_B(P), \varphi_B(Q)$ required to make the scheme work.

A different way to kill the active attack?

- Recall: SIDH cannot reuse keys because of the auxiliary points $\varphi_B(P), \varphi_B(Q)$ required to make the scheme work.
- These points are needed because Bob must help Alice "transport" her secret subgroup A to his public curve E_B.

A different way to kill the active attack?

- Recall: SIDH cannot reuse keys because of the auxiliary points $\varphi_B(P), \varphi_B(Q)$ required to make the scheme work.
- These points are needed because Bob must help Alice "transport" her secret subgroup A to his public curve E_B.

CSIDH's solution:

Use special subgroups A which can be transported to E_B independently of the secret isogeny φ_B.

How CSIDH avoids auxiliary points

Notation: For $E/\mathbb{F}_q$ and $\ell, \lambda \in \mathbb{Z}$, write $E_{\ell,\lambda} := E[\ell] \cap \ker(\pi - [\lambda])$.
(Recall $\pi : (x, y) \mapsto (x^q, y^q)$.)

How CSIDH avoids auxiliary points

Notation: For $E/\mathbb{F}_q$ and $\ell, \lambda \in \mathbb{Z}$, write $E_{\ell,\lambda} := E[\ell] \cap \ker(\pi - [\lambda])$.
(Recall $\pi : (x, y) \mapsto (x^q, y^q)$.)

Let $E/\mathbb{F}_q$ and $\ell, \lambda, m, \mu \in \mathbb{Z} \backslash \{0\}$.
Write $E' = E/E_{\ell,\lambda}$ and $E'' = E/E_{m,\mu}$.

$$\implies \quad E'/E'_{m,\mu} \cong E''/E''_{\ell,\lambda}\,.$$

How CSIDH avoids auxiliary points

Notation: For $E/\mathbb{F}_q$ and $\ell, \lambda \in \mathbb{Z}$, write $E_{\ell,\lambda} := E[\ell] \cap \ker(\pi - [\lambda])$.
(Recall $\pi : (x, y) \mapsto (x^q, y^q)$.)

Let $E/\mathbb{F}_q$ and $\ell, \lambda, m, \mu \in \mathbb{Z} \backslash \{0\}$.
Write $E' = E/E_{\ell,\lambda}$ and $E'' = E/E_{m,\mu}$.

$$\implies \quad E'/E'_{m,\mu} \cong E''/E''_{\ell,\lambda}\,.$$

Problem: The *typical* case is $E_{\ell,\lambda} = \{\infty\}$. Not useful.

How CSIDH avoids auxiliary points

Notation: For $E/\mathbb{F}_q$ and $\ell, \lambda \in \mathbb{Z}$, write $E_{\ell,\lambda} := E[\ell] \cap \ker(\pi - [\lambda])$.
(Recall $\pi : (x, y) \mapsto (x^q, y^q)$.)

Let $E/\mathbb{F}_q$ and $\ell, \lambda, m, \mu \in \mathbb{Z} \backslash \{0\}$.
Write $E' = E/E_{\ell,\lambda}$ and $E'' = E/E_{m,\mu}$.

$$\implies \quad E'/E'_{m,\mu} \cong E''/E''_{\ell,\lambda}\,.$$

Problem: The *typical* case is $E_{\ell,\lambda} = \{\infty\}$. Not useful.

But: *If* there are enough pairs ℓ, λ with $\varphi_{E_{\ell,\lambda}} \neq id$, we can use compositions of various $\varphi_{E_{\ell,\lambda}}$ as secret isogenies.

How CSIDH avoids auxiliary points

Notation: For $E/\mathbb{F}_q$ and $\ell, \lambda \in \mathbb{Z}$, write $E_{\ell,\lambda} := E[\ell] \cap \ker(\pi - [\lambda])$.
(Recall $\pi : (x, y) \mapsto (x^q, y^q)$.)

Let $E/\mathbb{F}_q$ and $\ell, \lambda, m, \mu \in \mathbb{Z}\backslash\{0\}$.
Write $E' = E/E_{\ell,\lambda}$ and $E'' = E/E_{m,\mu}$.

$$\implies \quad E'/E'_{m,\mu} \cong E''/E''_{\ell,\lambda}\,.$$

Problem: The *typical* case is $E_{\ell,\lambda} = \{\infty\}$. Not useful.

But: *If* there are enough pairs ℓ, λ with $\varphi_{E_{\ell,\lambda}} \neq id$, we can use compositions of various $\varphi_{E_{\ell,\lambda}}$ as secret isogenies.

CSIDH's solution:

- Use supersingular curves over $\mathbb{F}_p$ to easily control $\#E(\mathbb{F}_p)$.
- Pick smooth $p+1$, i.e., many small prime factors ℓ_i.
  $\rightsquigarrow$ All the groups $E_{\ell_i,\pm 1}$ are good for crypto!

CSIDH in one slide

CSIDH in one slide

- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.

CSIDH in one slide

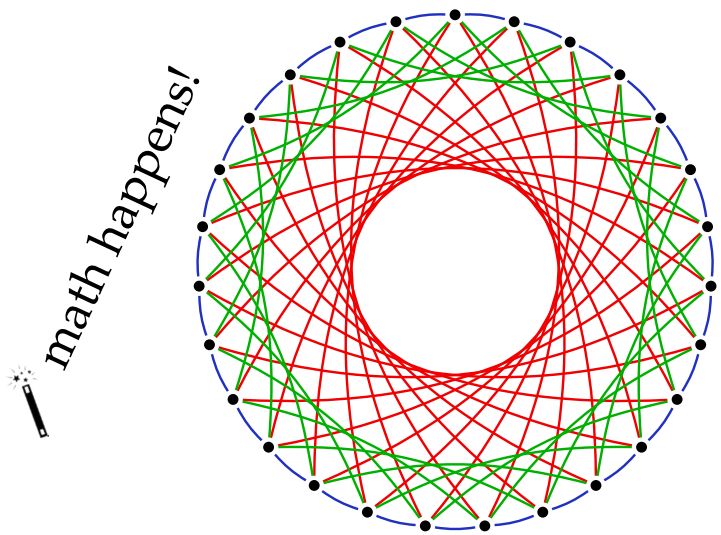
- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ supersingular with $A \in \mathbb{F}_p\}$.

CSIDH in one slide

- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ supersingular with $A \in \mathbb{F}_p\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.

CSIDH in one slide

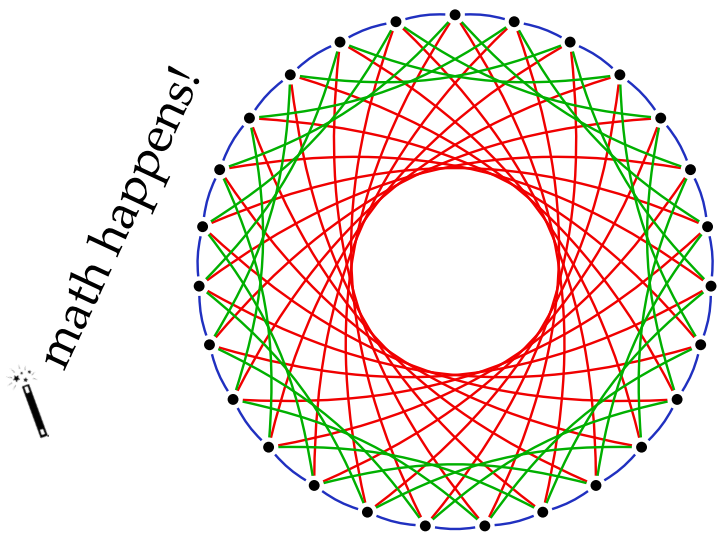
- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ supersingular with $A \in \mathbb{F}_p\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

CSIDH in one slide

- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ supersingular with $A \in \mathbb{F}_p\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



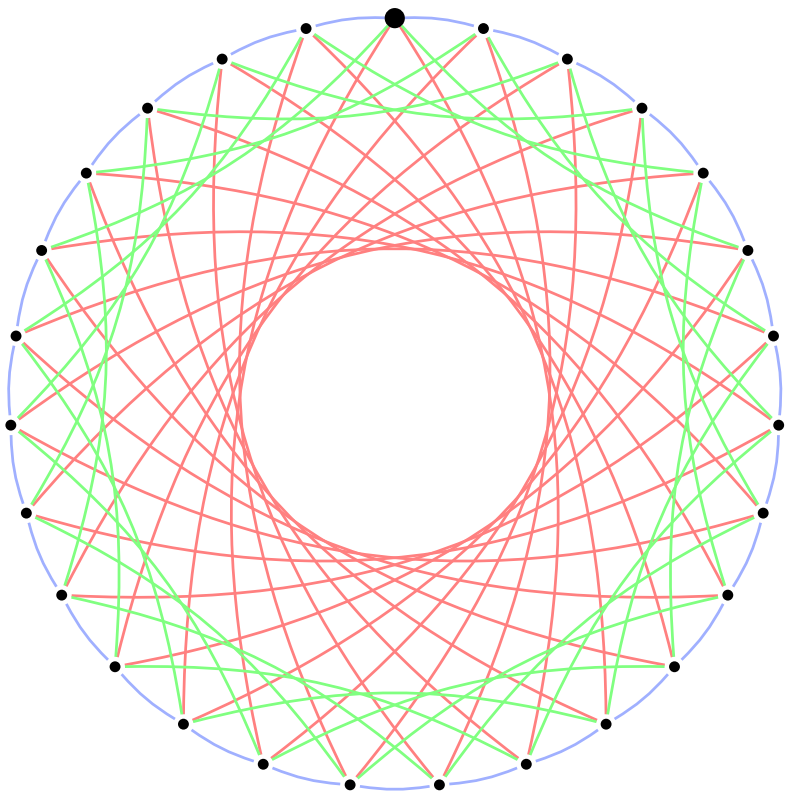
$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

- Walking "left" and "right" on any ℓ_i-subgraph is efficient.

CSIDH key exchange

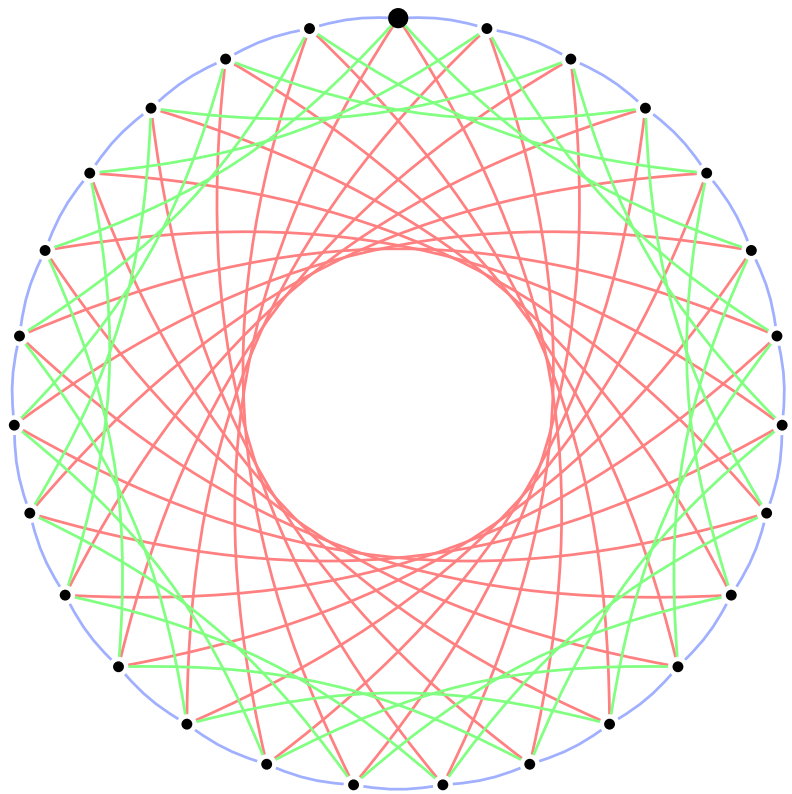
Alice

$[+, +, -, -]$



Bob

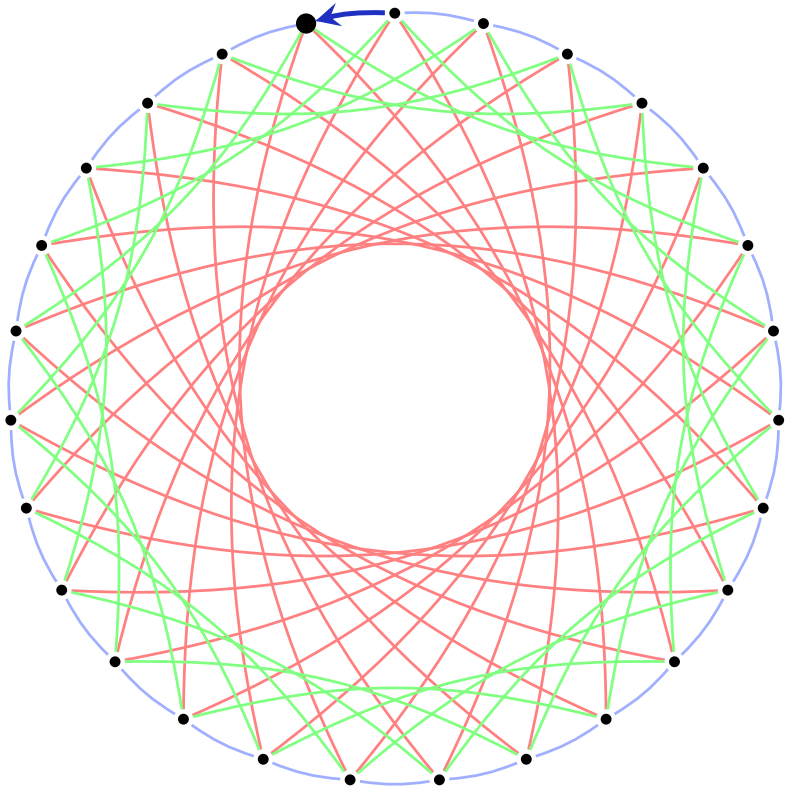
$[-, +, -, -]$



CSIDH key exchange

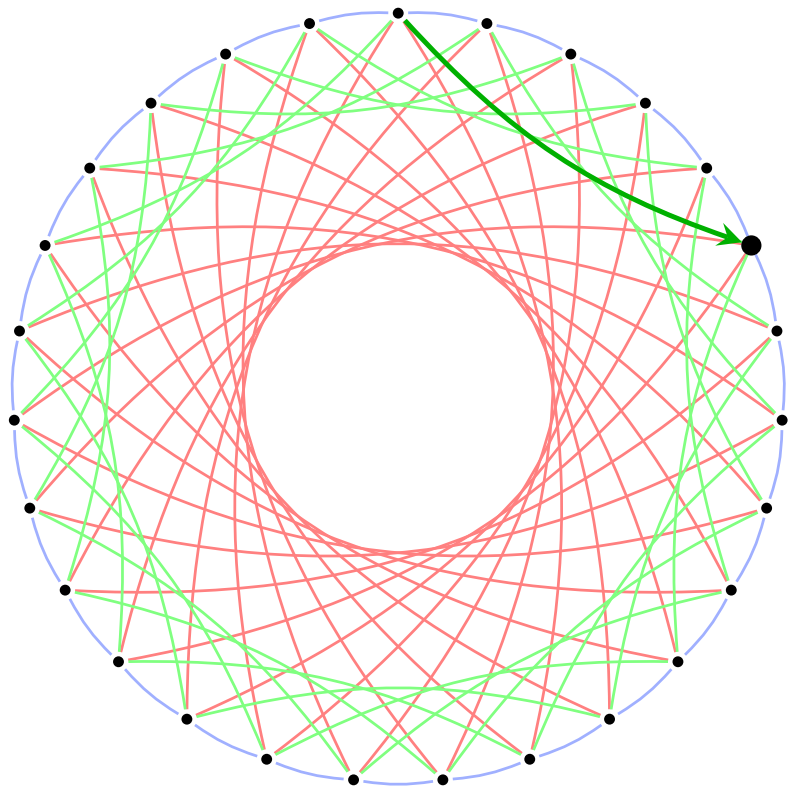
Alice

$[+, +, -, -]$



Bob

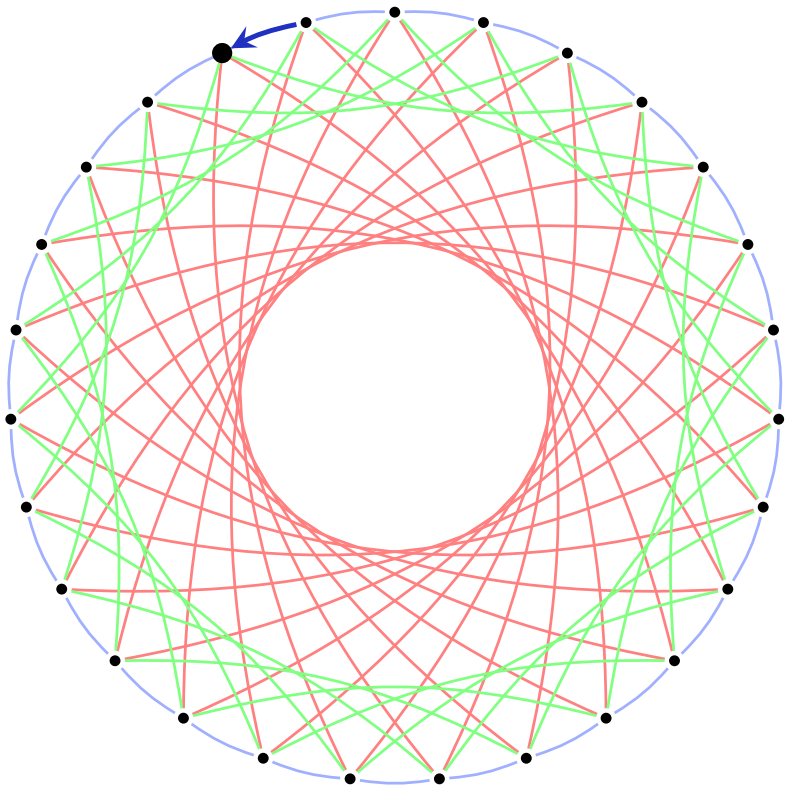
$[-, +, -, -]$



CSIDH key exchange

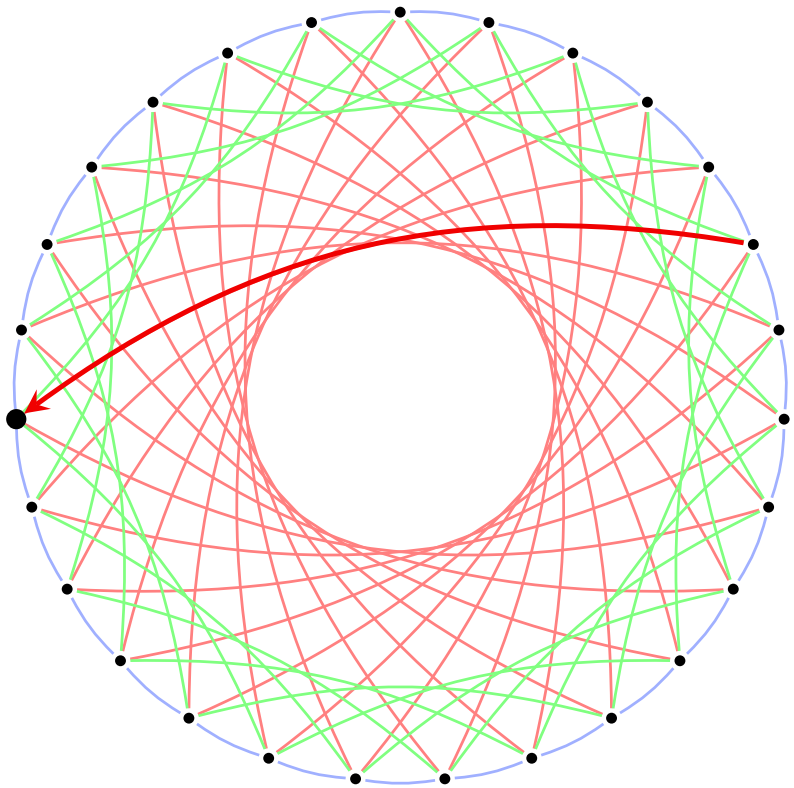
Alice

$[+, +, -, -]$



Bob

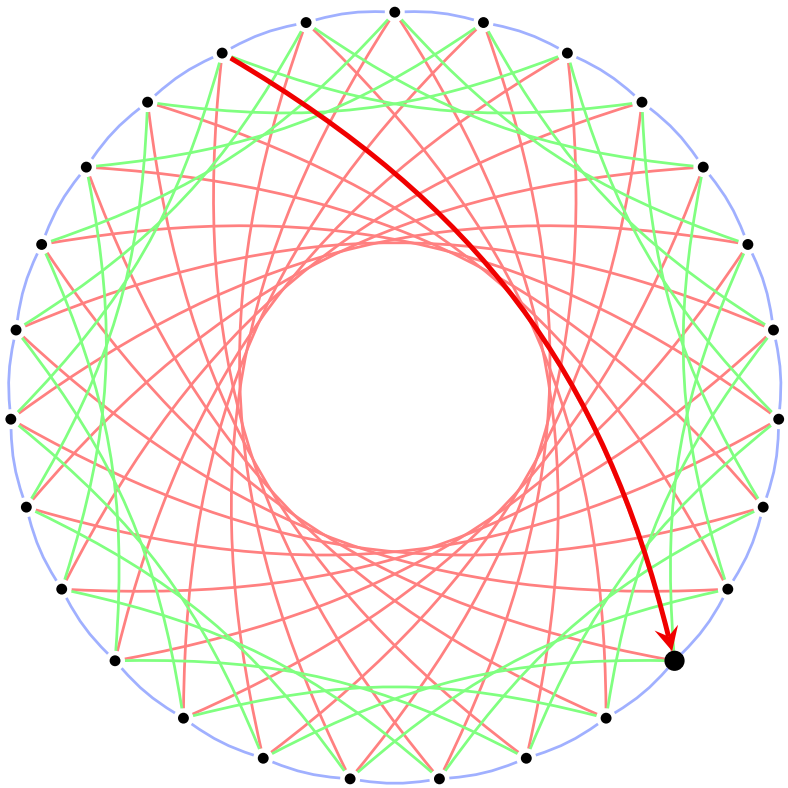
$[-, +, -, -]$



CSIDH key exchange

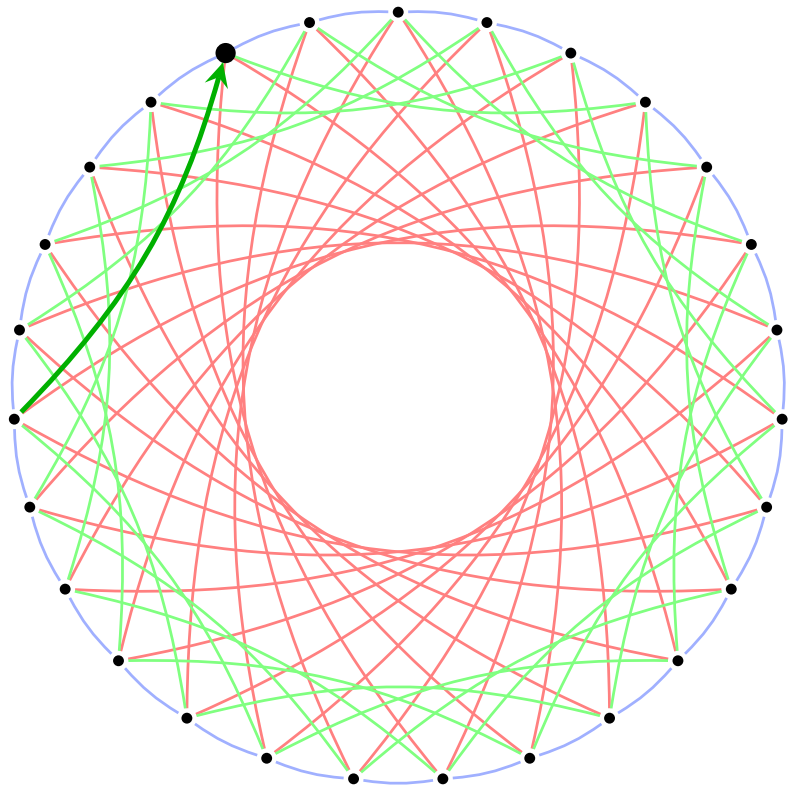
Alice

$[+, +, -, -]$



Bob

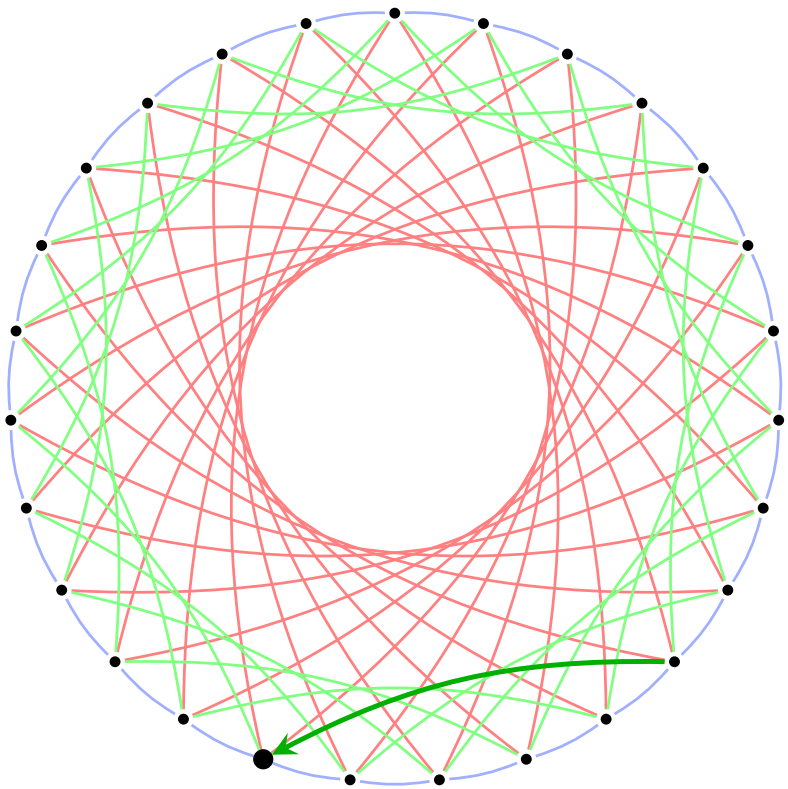
$[-, +, -, -]$



CSIDH key exchange

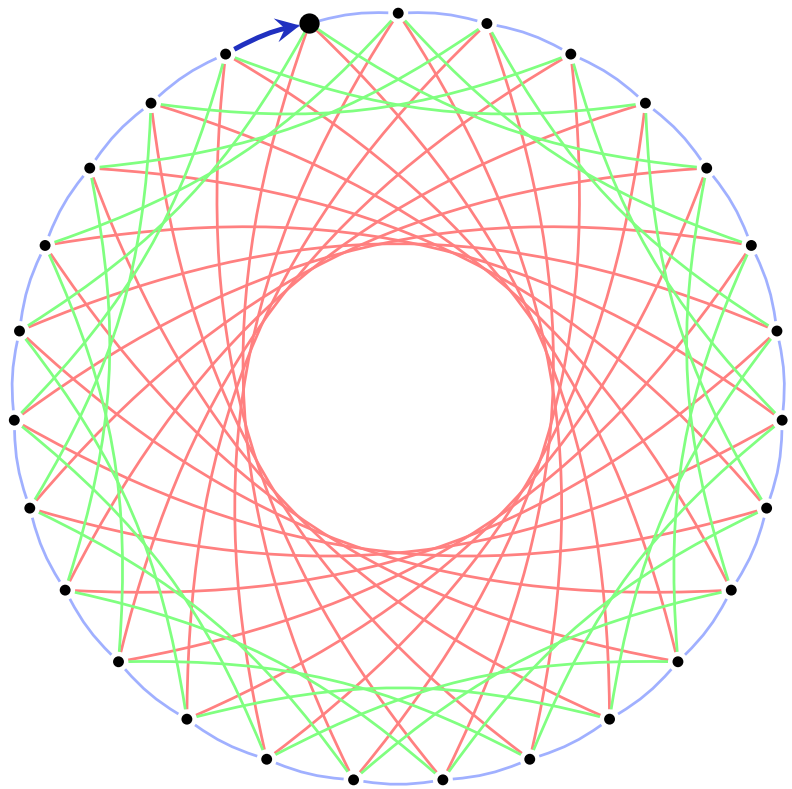
Alice

$[+, +, -, -]$



Bob

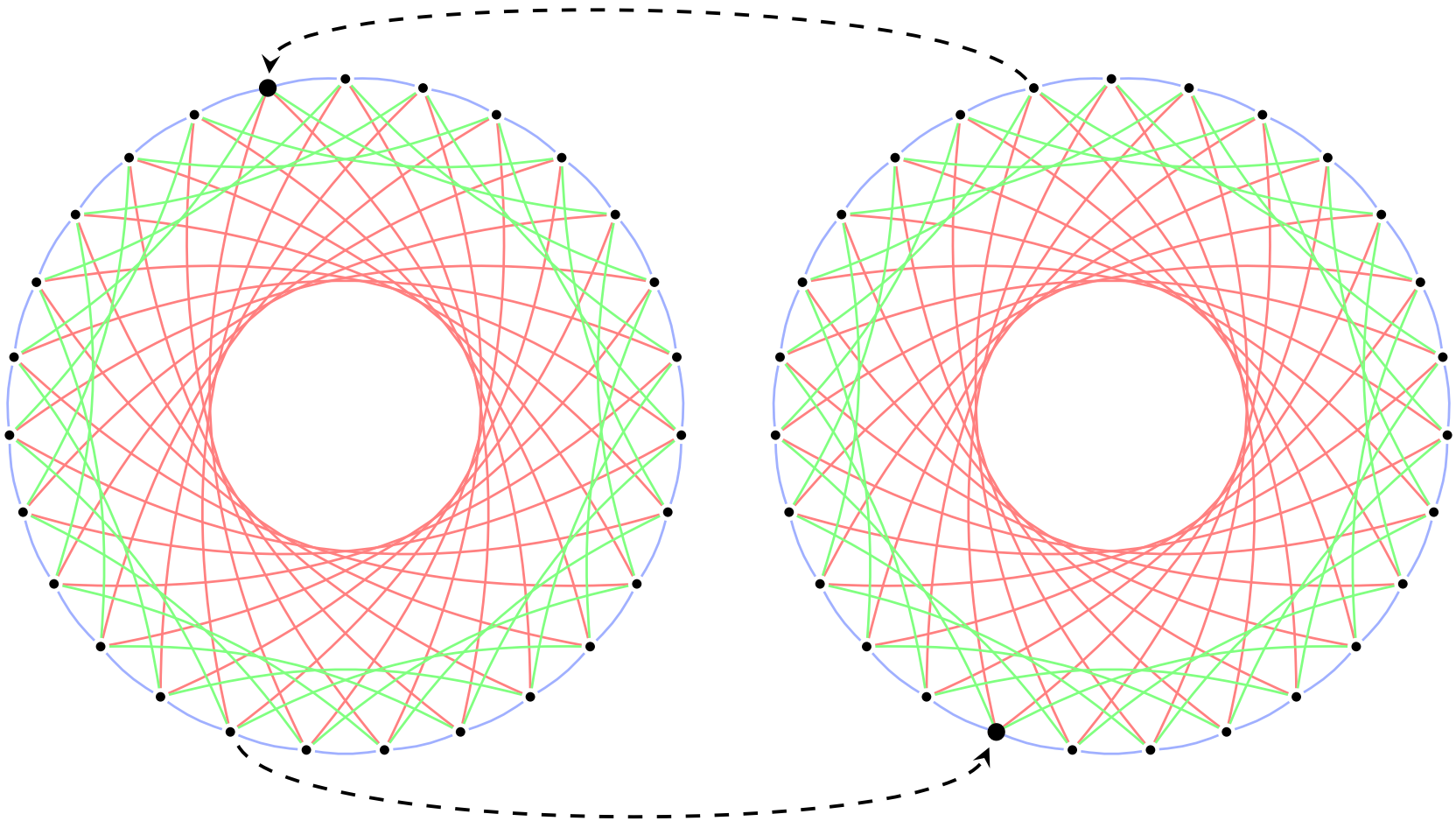
$[-, +, -, -]$



CSIDH key exchange

Alice
$[+, +, -, -]$

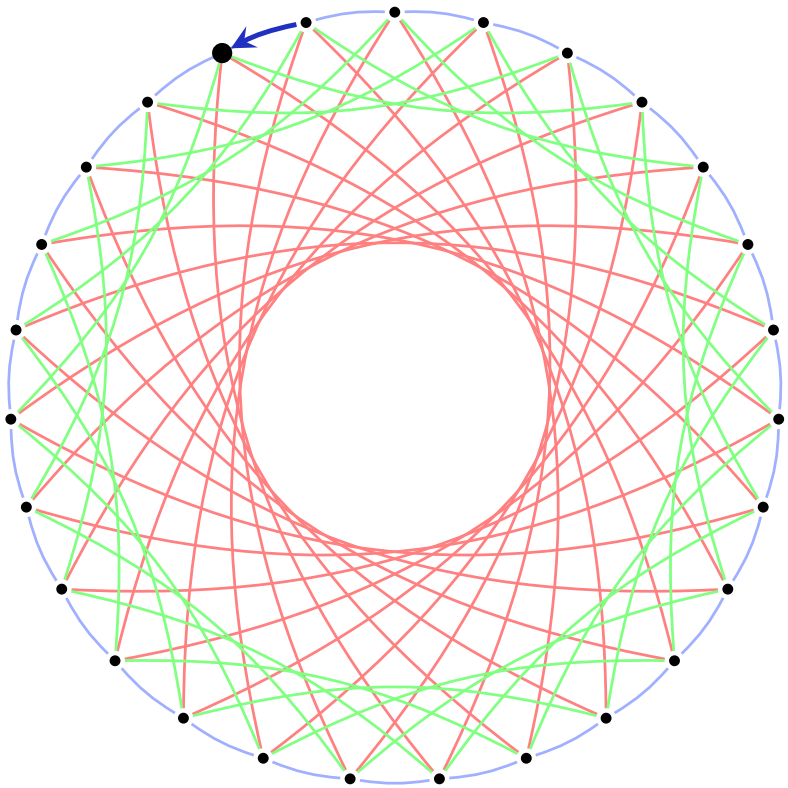
Bob
$[-, +, -, -]$



CSIDH key exchange

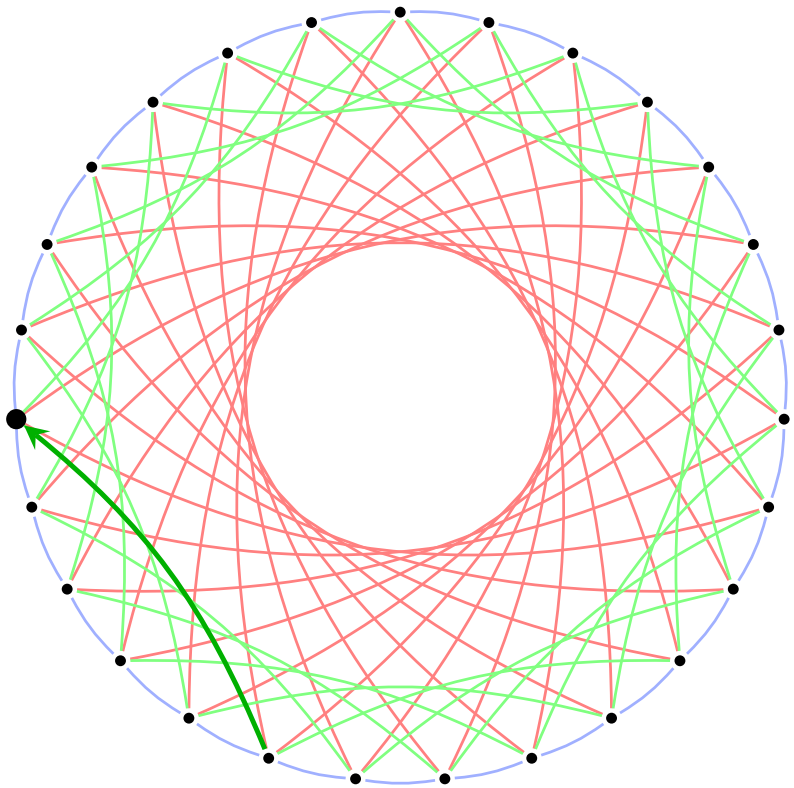
Alice

$[+, +, -, -]$



Bob

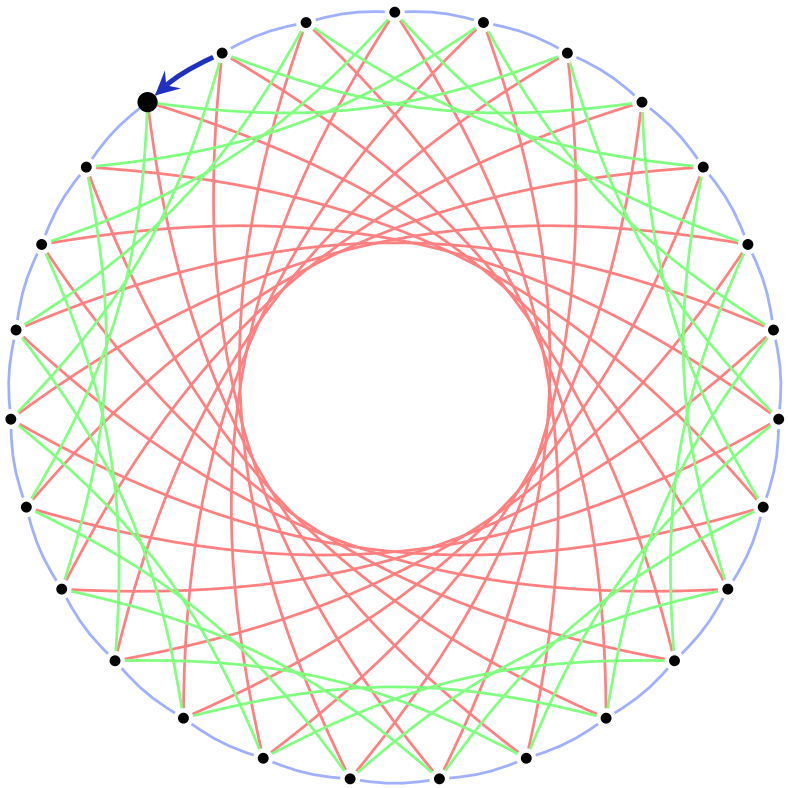
$[-, +, -, -]$



CSIDH key exchange

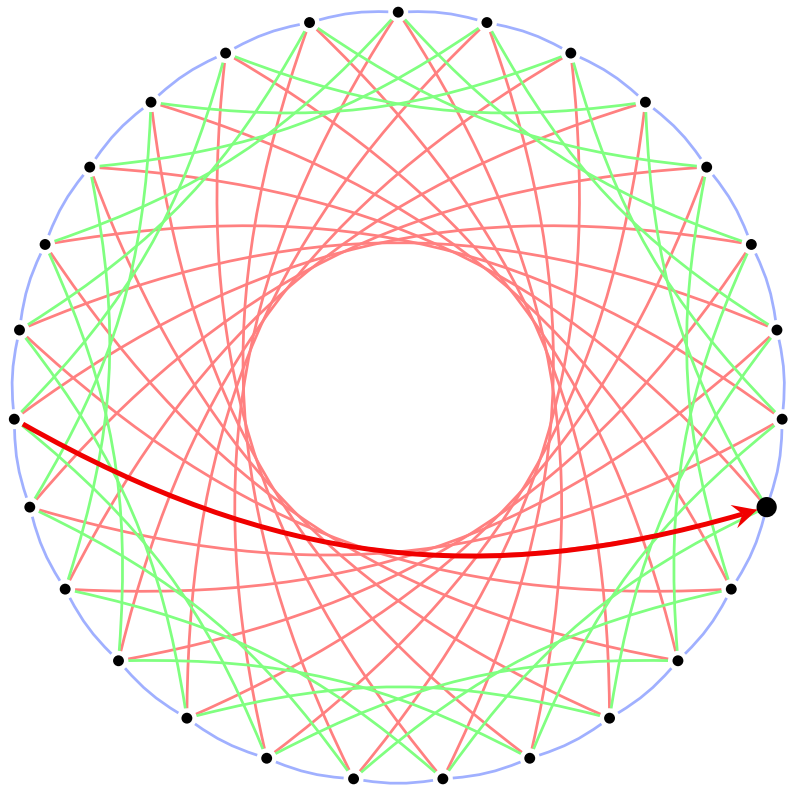
Alice

$[+, +, -, -]$



Bob

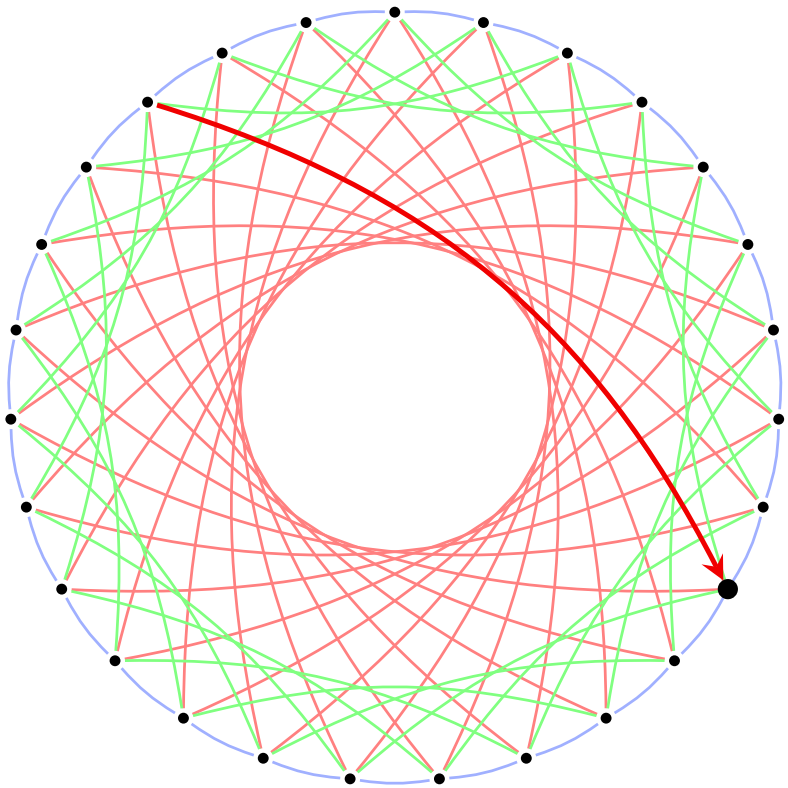
$[-, +, -, -]$



CSIDH key exchange

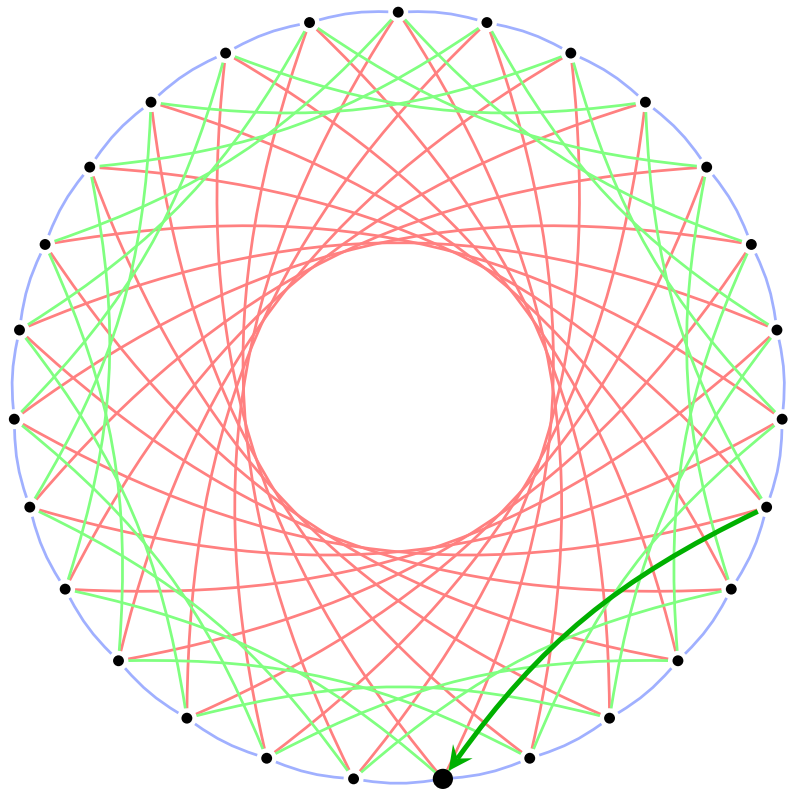
Alice

$[+, +, \underset{\uparrow}{-}, -]$



Bob

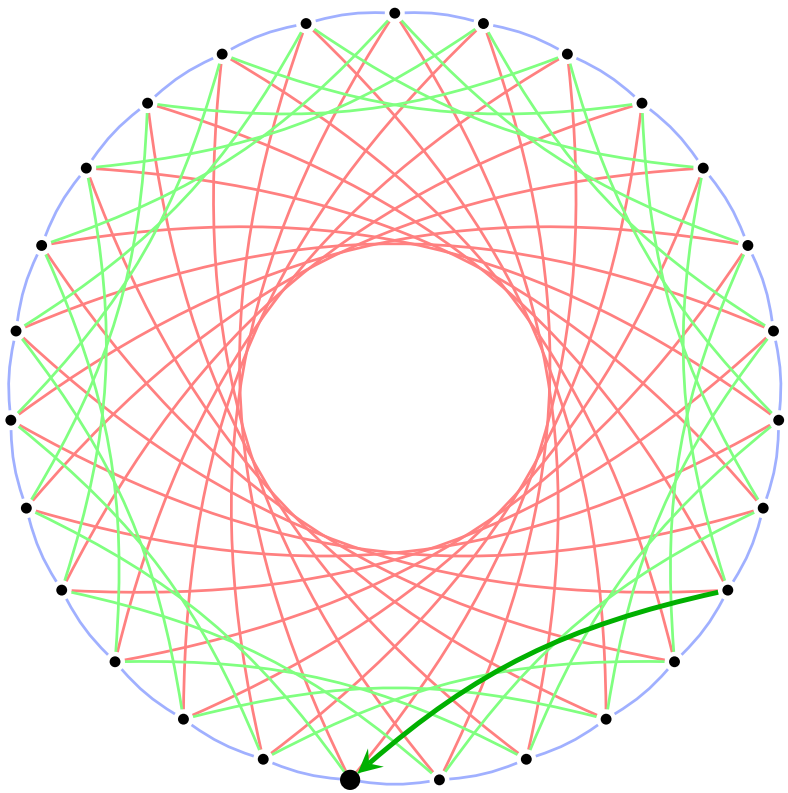
$[-, +, \underset{\uparrow}{-}, -]$



CSIDH key exchange

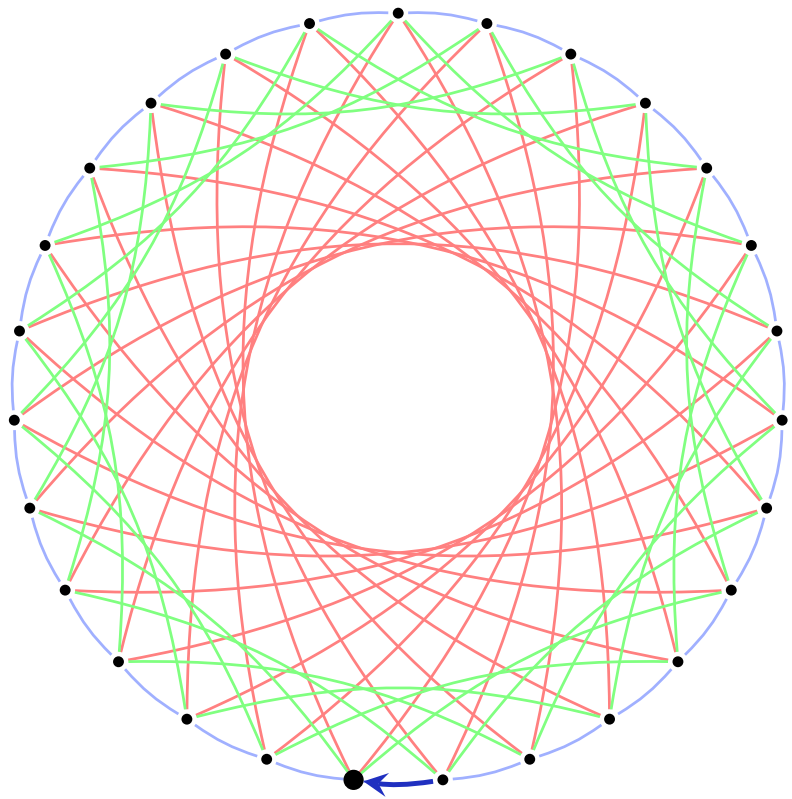
Alice

$[+, +, -, -]$



Bob

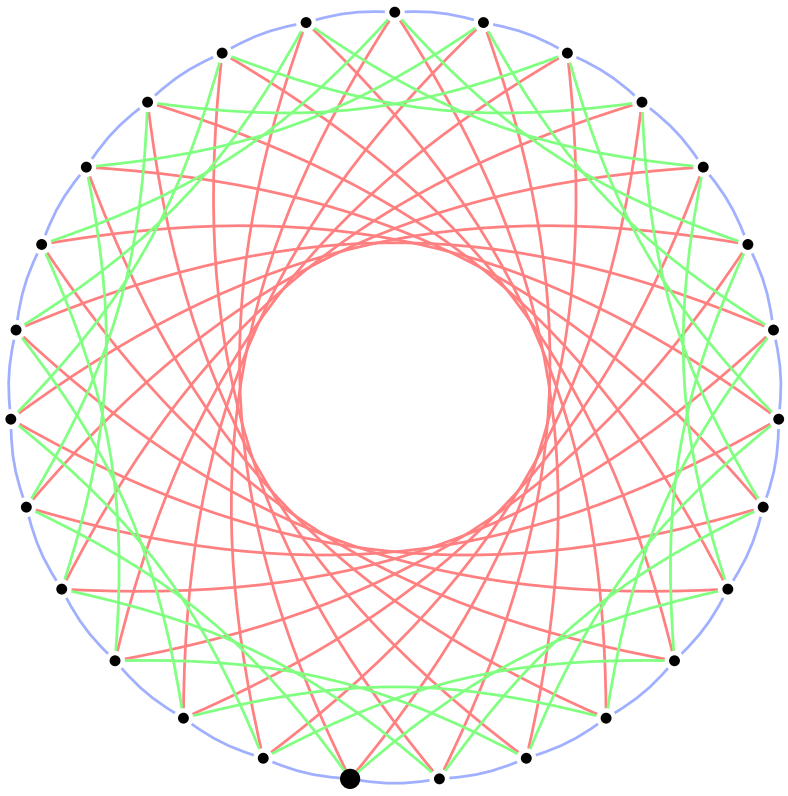
$[-, +, -, -]$



CSIDH key exchange

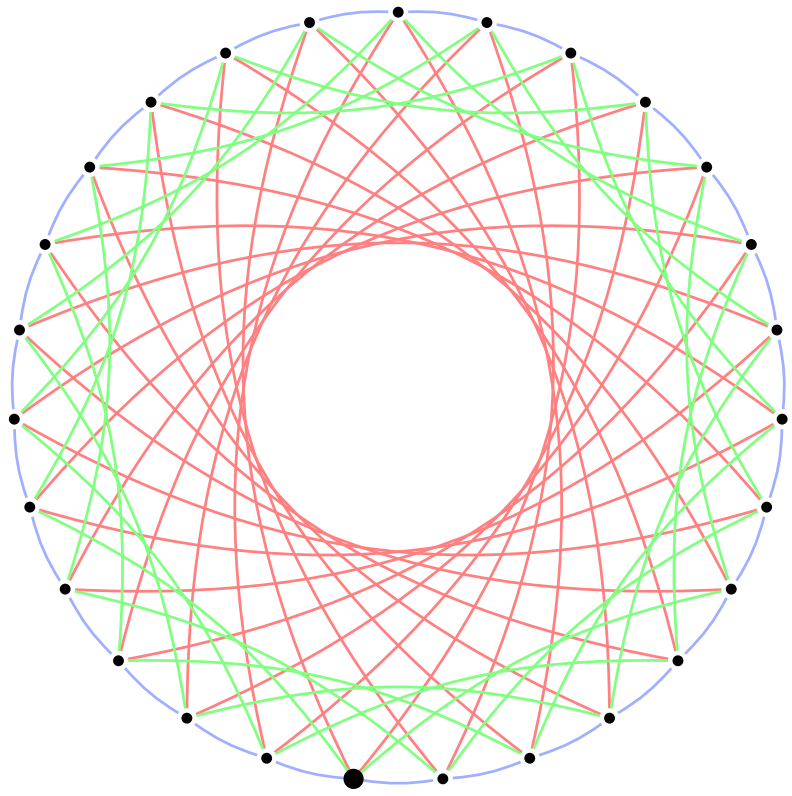
Alice

$[+, +, -, -]$



Bob

$[-, +, -, -]$



A group action

Cycles are compatible: [right then left] = [left then right]

A group action

Cycles are compatible: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of total step counts for each ℓ_i.

Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

A group action

Cycles are compatible: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of total step counts for each ℓ_i.

Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

There is a group action of $(\mathbb{Z}^n, +)$ on our set of curves X!

(An action of a group $(G, \cdot)$ on a set X is a map $* \colon G \times X \to X$ such that $id * x = x$ and $g * (h * x) = (g \cdot h) * x$ for all $g, h \in G$ and $x \in X$.)

A group action

Cycles are compatible: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of total step counts for each ℓ_i.

Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

There is a group action of $(\mathbb{Z}^n, +)$ on our set of curves X!

(An action of a group $(G, \cdot)$ on a set X is a map $* \colon G \times X \to X$ such that $id * x = x$ and $g * (h * x) = (g \cdot h) * x$ for all $g, h \in G$ and $x \in X$.)

We understand the structure:
By complex-multiplication theory, the action factors through the ideal-class group $\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$.

A brief history of CSIDH

Well-known classical theorem:

> *Sometimes*, there is a (free & transitive) group action of $\mathrm{cl}(\mathcal{O})$ on the set of curves with endomorphism ring $\mathcal{O}$.

A brief history of CSIDH

Well-known classical theorem:

Sometimes, there is a (free & transitive) group action of $\mathrm{cl}(\mathcal{O})$ on the set of curves with endomorphism ring $\mathcal{O}$.

[Couveignes 1997/2006], independently [Rostovtsev–Stolbunov 2006]:

Use this group action on ordinary curves for Diffie–Hellman.

A brief history of CSIDH

Well-known classical theorem:

> *Sometimes*, there is a (free & transitive) group action of $\mathrm{cl}(\mathcal{O})$ on the set of curves with endomorphism ring $\mathcal{O}$.

[Couveignes 1997/2006], independently [Rostovtsev–Stolbunov 2006]:

> Use this group action on ordinary curves for Diffie–Hellman.

[De Feo–Kieffer–Smith 2018]:

> Massive speedups, but still unbearably slow.

A brief history of CSIDH

Well-known classical theorem:

> *Sometimes*, there is a (free & transitive) group action of $\mathrm{cl}(\mathcal{O})$ on the set of curves with endomorphism ring $\mathcal{O}$.

[Couveignes 1997/2006], independently [Rostovtsev–Stolbunov 2006]:

> Use this group action on ordinary curves for Diffie–Hellman.

[De Feo–Kieffer–Smith 2018]:

> Massive speedups, but still unbearably slow.

[Castryck–Lange–Martindale–Panny–Renes 2018]:

> Switch to supersingular curves $\implies$ "practical" performance.

Walking in the CSIDH graph

- "Left" and "right" steps correspond to isogenies with special subgroups $E_{\ell_i,\pm 1}$ as kernels.

  (Recall that $E_{\ell,\lambda} = \{P \in E[\ell] \mid \pi(P) = [\lambda]P\}$.)

Walking in the CSIDH graph

- "Left" and "right" steps correspond to isogenies with special subgroups $E_{\ell_i,\pm 1}$ as kernels.

  (Recall that $E_{\ell,\lambda} = \{P \in E[\ell] \mid \pi(P) = [\lambda]P\}$.)

Computing a "left" step:

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$.

Walking in the CSIDH graph

- "Left" and "right" steps correspond to isogenies with special subgroups $E_{\ell_i,\pm 1}$ as kernels.

  (Recall that $E_{\ell,\lambda} = \{P \in E[\ell] \mid \pi(P) = [\lambda]P\}$.)

Computing a "left" step:

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$.

Computing a "right" step:

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$.

Walking in the CSIDH graph

- "Left" and "right" steps correspond to isogenies with special subgroups $E_{\ell_i,\pm 1}$ as kernels.

  (Recall that $E_{\ell,\lambda} = \{P \in E[\ell] \mid \pi(P) = [\lambda]P\}$.)

Computing a "left" step:

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$.

Computing a "right" step:

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$.

(Finding a point of order ℓ_i: Pick $x \in \mathbb{F}_p$ random. Find $y \in \mathbb{F}_{p^2}$ such that $P = (x, y) \in E$. Compute $Q = [\frac{p+1}{\ell_i}]P$. Hope that $Q \neq \infty$, else retry.)

Efficient x-only arithmetic

- For $n \in \mathbb{Z}$, we have $[n](-P) = -[n]P$. (This holds in any group.)

Efficient x-only arithmetic

- For $n \in \mathbb{Z}$, we have $[n](-P) = -[n]P$. (This holds in any group.)
- Recall that $P = (x, y)$ has inverse $-P = (x, -y)$.

Efficient x-only arithmetic

- For $n \in \mathbb{Z}$, we have $[n](-P) = -[n]P$. (This holds in any group.)
- Recall that $P = (x, y)$ has inverse $-P = (x, -y)$.

$\Longrightarrow$ We get an induced map $\mathtt{xMUL}_n$ on x-coordinates such that

$$\forall P \in E. \quad \mathtt{xMUL}_n(x(P)) = x([n]P)\,.$$

Efficient x-only arithmetic

- For $n \in \mathbb{Z}$, we have $[n](-P) = -[n]P$. (This holds in any group.)
- Recall that $P = (x, y)$ has inverse $-P = (x, -y)$.

$\Longrightarrow$ We get an induced map $\mathtt{xMUL}_n$ on x-coordinates such that

$$\forall P \in E. \quad \mathtt{xMUL}_n(x(P)) = x([n]P)\,.$$

The same reasoning works for isogeny formulas.

Net result: With x-only arithmetic everything happens over $\mathbb{F}_p$.
$\Longrightarrow$ Efficient CSIDH implementations!

Why no Shor?

Shor's quantum algorithm computes α from g^α in any group in polynomial time.

Why no Shor?

Shor's quantum algorithm computes α from g^α in any group in polynomial time.

Shor computes α from $h = g^\alpha$ by finding the kernel of the map

$$f\colon\ \mathbb{Z}^2 \to G,\ (x,y) \mapsto g^x \cdot h^y\ .$$

Why no Shor?

Shor's quantum algorithm computes α from g^α in any group in polynomial time.

Shor computes α from $h = g^\alpha$ by finding the kernel of the map

$$f\colon\ \mathbb{Z}^2 \to G,\ (x,y) \mapsto g^x \cdot h^y\,.$$

For group actions, we simply cannot compose $a * s$ and $b * s$!

Security of CSIDH

Core problem:
Given $E, E' \in X$, find a smooth-degree isogeny $E \to E'$.

Security of CSIDH

Core problem:
Given $E, E' \in X$, find a smooth-degree isogeny $E \to E'$.

The size of X is $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) = 3 \cdot h(-p) \approx \sqrt{p}$.

$\rightsquigarrow$ best known classical attack: meet-in-the-middle, $\tilde{\mathcal{O}}(p^{1/4})$.

Fully exponential: Complexity $\exp\big((\log p)^{1+o(1)}\big)$.

Security of CSIDH

Core problem:
Given $E, E' \in X$, find a smooth-degree isogeny $E \to E'$.

The size of X is $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) = 3 \cdot h(-p) \approx \sqrt{p}$.

$\rightsquigarrow$ best known classical attack: meet-in-the-middle, $\tilde{\mathcal{O}}(p^{1/4})$.

Fully exponential: Complexity $\exp\big((\log p)^{1+o(1)}\big)$.

Solving abelian hidden shift breaks CSIDH.

$\rightsquigarrow$ non-devastating quantum attack (Kuperberg's algorithm).

Subexponential: Complexity $\exp\big((\log p)^{1/2+o(1)}\big)$.

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. Evaluate the group action many times. ("oracle calls")
2. Combine the results in a certain way. ("sieving")

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. Evaluate the group action many times. ("oracle calls")
2. Combine the results in a certain way. ("sieving")

- The algorithm admits many different tradeoffs.
- Oracle calls are expensive.
- The sieving phase has classical *and* quantum operations.

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. Evaluate the group action many times. ("oracle calls")
2. Combine the results in a certain way. ("sieving")

- The algorithm admits many different tradeoffs.
- Oracle calls are expensive.
- The sieving phase has classical *and* quantum operations.

  How to compare costs?
  (Is one qubit operation $\approx$ one bit operation? a hundred? millions?)

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. Evaluate the group action many times. ("oracle calls")
2. Combine the results in a certain way. ("sieving")

- The algorithm admits many different tradeoffs.
- Oracle calls are expensive.
- The sieving phase has classical *and* quantum operations.

  How to compare costs?
  (Is one qubit operation $\approx$ one bit operation? a hundred? millions?)

$\Longrightarrow$ Security estimates for CSIDH vary wildly.

Plan for this lecture

- High-level overview for intuition. ✓
- Elliptic curves & isogenies, algorithms. ✓
- The SIDH/SIKE key-agreement protocol. ✓
- The CSIDH non-interactive key-exchange. ✓

That's nice and all, but... so what?

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

SIDH/SIKE...

- survived NIST's post-quantum not-a-competition (but wasn't selected as a winner in the first round).

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

SIDH/SIKE...

- survived NIST's post-quantum not-a-competition (but wasn't selected as a winner in the first round).
- has exponential attack cost as far as we know.

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

SIDH/SIKE...

- survived NIST's post-quantum not-a-competition (but wasn't selected as a winner in the first round).
- has exponential attack cost as far as we know.

Both...

- have tiny keys compared to other post-quantum schemes.

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

SIDH/SIKE...

- survived NIST's post-quantum not-a-competition (but wasn't selected as a winner in the first round).
- has exponential attack cost as far as we know.

Both...

- have tiny keys compared to other post-quantum schemes.
- are quite slow compared to other post-quantum schemes.

That's nice and all, but... so what?

CSIDH...

- is a drop-in post-quantum replacement for (EC)DH.
- is the only known somewhat efficient post-quantum non-interactive key exchange (full public-key validation).
- has a clean mathematical structure: a true group action.

SIDH/SIKE...

- survived NIST's post-quantum not-a-competition (but wasn't selected as a winner in the first round).
- has exponential attack cost as far as we know.

Both...

- have tiny keys compared to other post-quantum schemes.
- are quite slow compared to other post-quantum schemes.
- are really cool!

Brainteaser

Brainteaser

Recall **SIDH**:

- Setup: $E[2^n] = \langle P, Q \rangle$ and $E[3^m] = \langle R, S \rangle$
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(R), \varphi_A(S))$
- Bob: $B := \langle R + [b]S \rangle \leq E[3^m]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Brainteaser

Recall **SIDH**:

- Setup: $E[2^n] = \langle P, Q \rangle$ and $E[3^m] = \langle R, S \rangle$
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(R), \varphi_A(S))$
- Bob: $B := \langle R + [b]S \rangle \leq E[3^m]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Let's look at "**BAD SIDH**":

- Setup: $E[2^n] = \langle P, Q \rangle$ ~~and $E[3^m] = \langle R, S \rangle$~~
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(P), \varphi_A(Q))$
- Bob: $B := \langle P + [b]Q \rangle \leq E[2^n]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Brainteaser

Recall **SIDH**:

- Setup: $E[2^n] = \langle P, Q \rangle$ and $E[3^m] = \langle R, S \rangle$
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(R), \varphi_A(S))$
- Bob: $B := \langle R + [b]S \rangle \leq E[3^m]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Let's look at "**BAD SIDH**":

- Setup: $E[2^n] = \langle P, Q \rangle$ ~~and $E[3^m] = \langle R, S \rangle$~~
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(P), \varphi_A(Q))$
- Bob: $B := \langle P + [b]Q \rangle \leq E[2^n]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Advantage: Need only $2^n \mid (p+1)$ instead of $2^n 3^m \mid (p+1)$.

Brainteaser

Recall **SIDH**:

- Setup: $E[2^n] = \langle P, Q \rangle$ and $E[3^m] = \langle R, S \rangle$
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(R), \varphi_A(S))$
- Bob: $B := \langle R + [b]S \rangle \leq E[3^m]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Let's look at "**BAD SIDH**":

- Setup: $E[2^n] = \langle P, Q \rangle$ ~~and $E[3^m] = \langle R, S \rangle$~~
- Alice: $A := \langle P + [a]Q \rangle \leq E[2^n]$, public $(E_A, \varphi_A(P), \varphi_A(Q))$
- Bob: $B := \langle P + [b]Q \rangle \leq E[2^n]$, public $(E_B, \varphi_B(P), \varphi_B(Q))$

Advantage: Need only $2^n \mid (p+1)$ instead of $2^n 3^m \mid (p+1)$.

Disadvantage: Insecure. *Why?*