Isogenies I & II

Lorenz Panny

Technische Universiteit Eindhoven

Executive School on Post-Quantum Cryptography,
Eindhoven, 2 July 2019

Please ask me anything!

Diffie–Hellman key exchange '76

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 '76

Public parameters:

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

<u>Alice</u>	<u>public</u>	<u>Bob</u>
$a \xleftarrow{\text{random}} \{0 \ldots q-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots q-1\}$
g^a		g^b
$s := (g^b)^a$		$s := (g^a)^b$

Diffie–Hellman key exchange ’76

Public parameters:

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

Alice	public	Bob
$a \xleftarrow{\text{random}} \{0 \ldots q-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots q-1\}$
g^a		g^b
$s := (g^b)^a$		$s := (g^a)^b$

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

Diffie–Hellman: Bob vs. Eve

<u>Bob</u>

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

<u>Bob</u>

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

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 $B \leftarrow t \cdot g$.

Attacker Eve

1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.

...

$b{-}2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b{-}2$.

$b{-}1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b{-}1$.

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 $b + 2$.

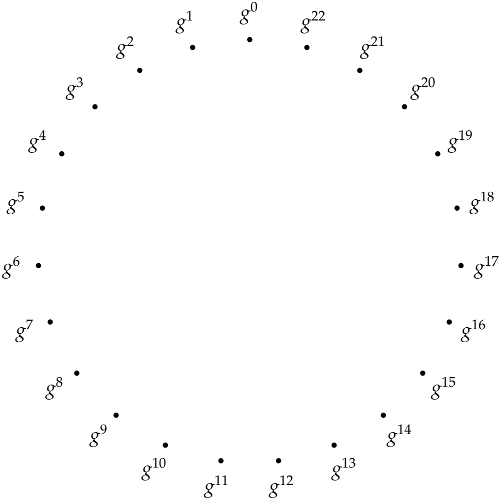
...

Diffie–Hellman: Bob vs. Eve

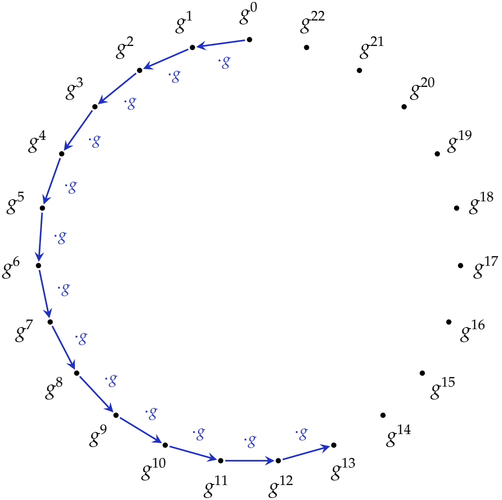
Bob	Attacker Eve
1. Set $t \leftarrow g$.	1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$.	2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$.	3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$.	4. Set $t \leftarrow t \cdot g$. 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.)

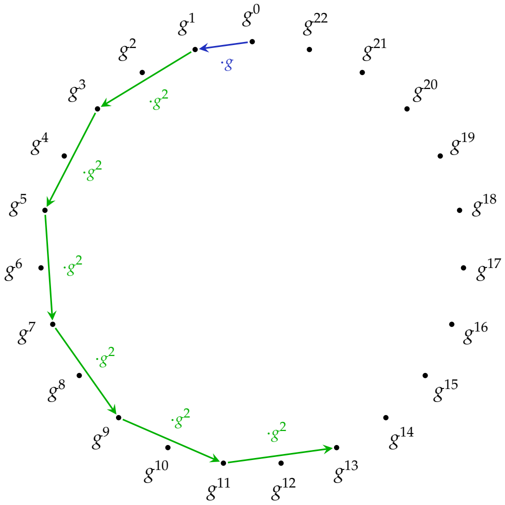

g^0
g^1
g^2
g^3
g^4
g^5
g^6
g^7
g^8
g^9
g^{10}
g^{11}
g^{12}
g^{13}
g^{14}
g^{15}
g^{16}
g^{17}
g^{18}
g^{19}
g^{20}
g^{21}
g^{22}

multiply

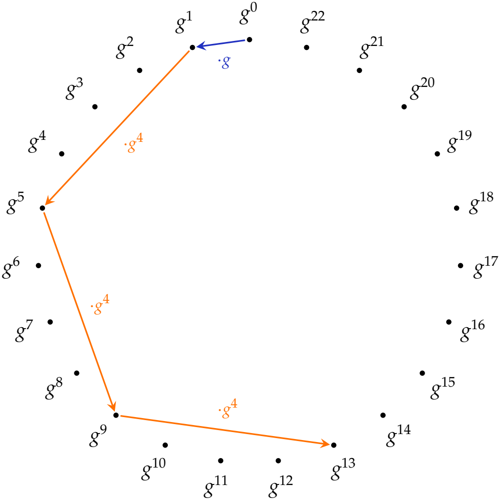


g^0
g^1
g^2
g^3
g^4
g^5
g^6
g^7
g^8
g^9
g^{10}
g^{11}
g^{12}
g^{13}
g^{14}
g^{15}
g^{16}
g^{17}
g^{18}
g^{19}
g^{20}
g^{21}
g^{22}
$\cdot g$

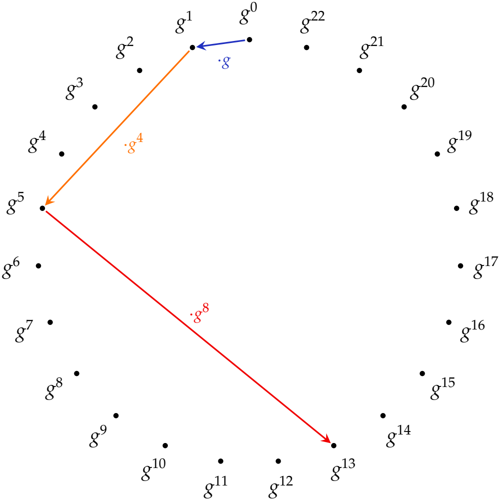
Square-and-multiply



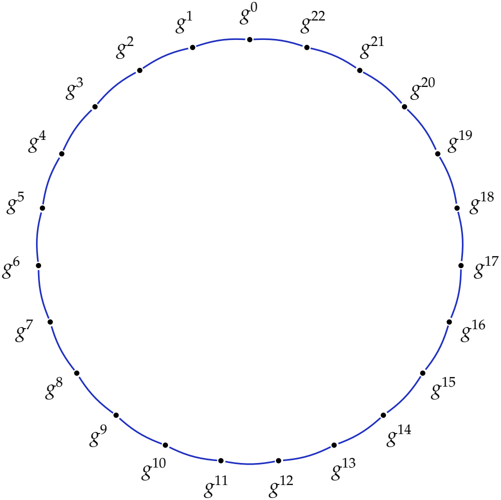
Square-and-multiply-and-square-and-multiply



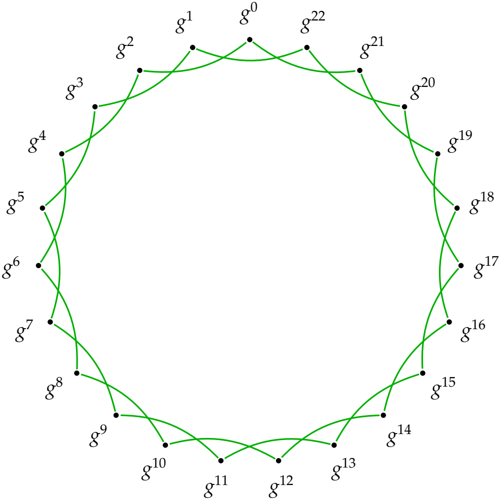
Square-and-multiply-and-square-and-multiply-and-squ



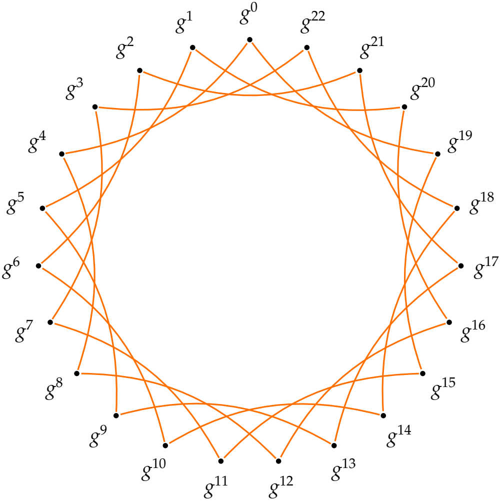
Square-and-multiply as graphs



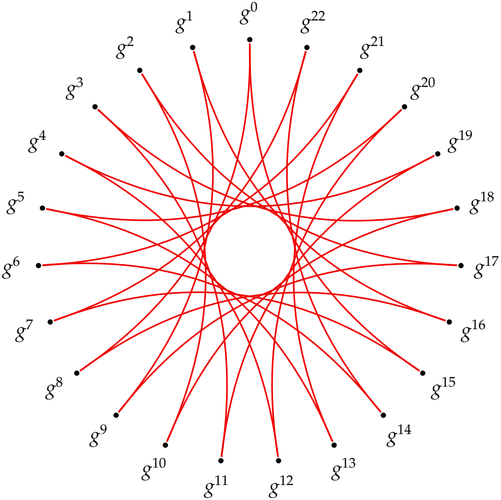
Square-and-multiply as graphs



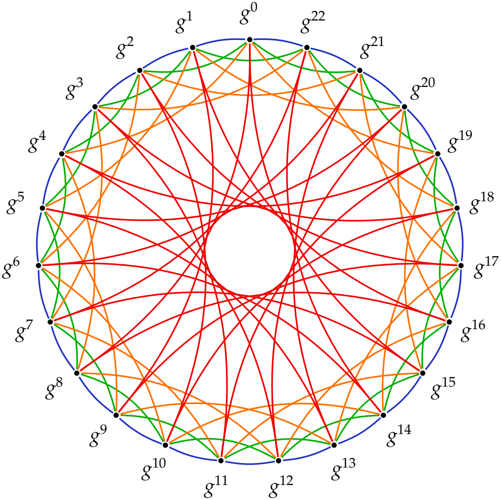
Square-and-multiply as graphs



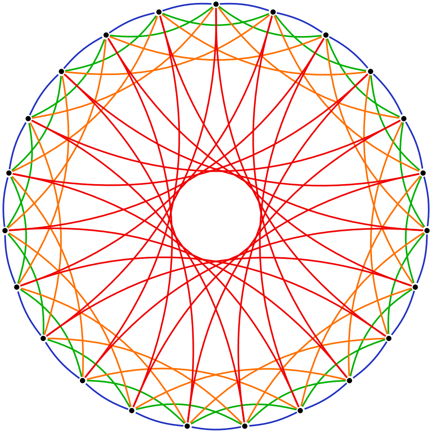
Square-and-multiply as graphs



Square-and-multiply as a graph



Square-and-multiply as a graph



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

With square-and-multiply, applying b takes $\Theta(\log \#G)$.

For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

$\rightsquigarrow$ Exponential separation!

With square-and-multiply, applying b takes $\Theta(\log \#G)$.

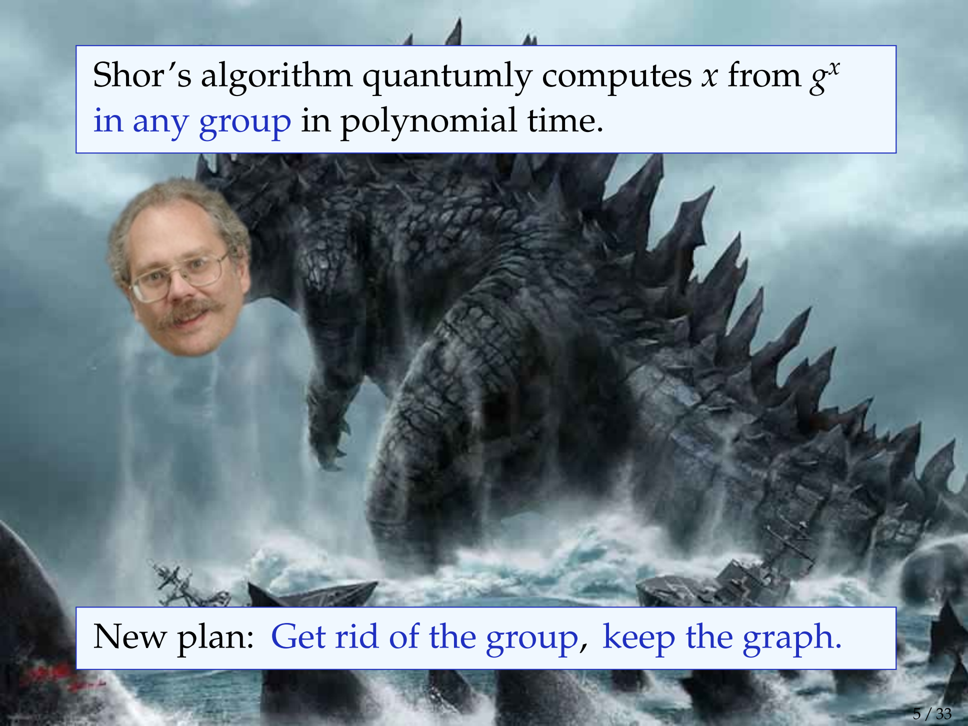
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

$\rightsquigarrow$ Exponential separation!

...and they lived happily ever after?

Shor's algorithm quantumly computes x from g^x in any group in polynomial time.

Shor's algorithm quantumly computes x from g^x in any group in polynomial time.



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

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. More later.

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. More later.

It is easy to construct graphs that satisfy *almost* all of these —
not enough for crypto!

Upshot

Isogenies give rise to

‘post-quantum Diffie–Hellman’.

(and more!)

Slightly smaller picture

- Isogenies are well-behaved maps between elliptic curves.

Slightly smaller picture

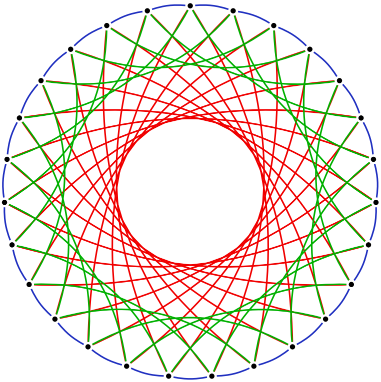
- Isogenies are well-behaved maps between elliptic curves.

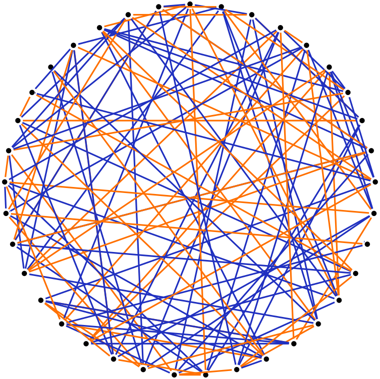
$\rightsquigarrow$ Isogeny graph: Nodes are curves, edges are isogenies.

(We usually care about subgraphs with certain properties.)

The beauty and the beast

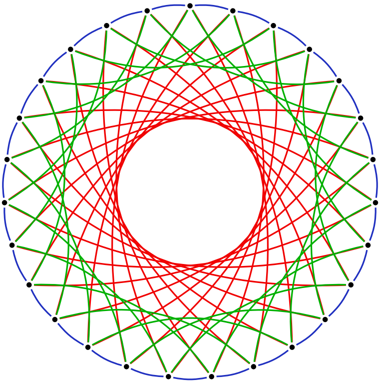
Components of well-chosen isogeny graphs look like this:

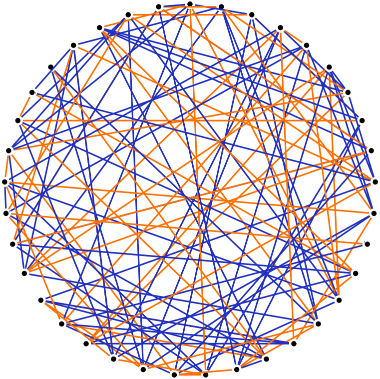




The beauty and the beast

Components of well-chosen isogeny graphs look like this:

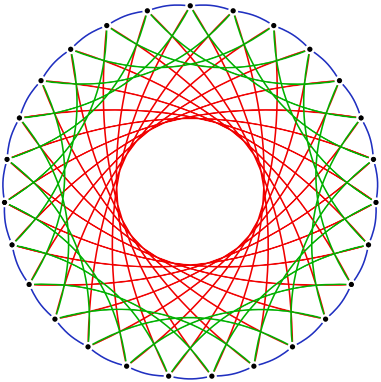


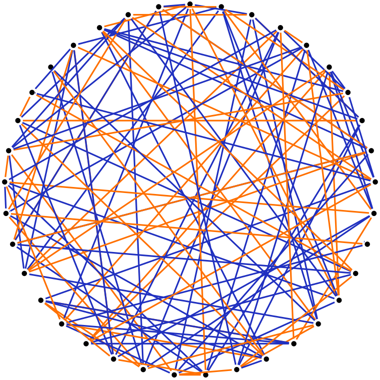


Which of these is good for crypto?

The beauty and the beast

Components of well-chosen isogeny graphs look like this:

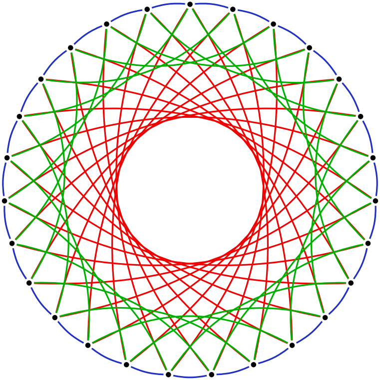




Which of these is good for crypto? **Both.**

The beauty and the beast

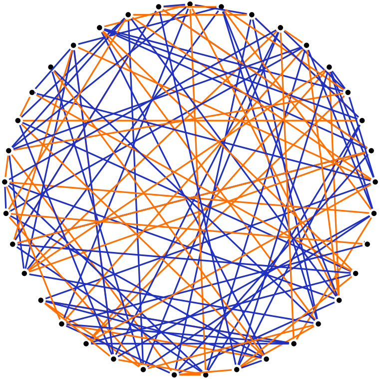
At this time, there are two distinct families of systems:



$q = p$

CSIDH [ˈsiːˌsaɪd]

`https://csidh.isogeny.org`



$q = p^2$

SIDH

`https://sike.org`

CSIDH [ˈsiːˌsaɪd]

(Castryck, Lange, Martindale, Panny, Renes; 2018)

CSIDH [ˈsiːˌsaɪd]

(Castryck, Lange, Martindale, Panny, Renes; 2018)

Why CSIDH?

- Drop-in post-quantum replacement for (EC)DH.

Why CSIDH?

- Drop-in post-quantum replacement for (EC)DH.
- Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)

Why CSIDH?

- Drop-in post-quantum replacement for (EC)DH.
- Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)
- Small keys: starts at 64 bytes.*

* Security evaluation is complicated, might get bigger & slower.

Why CSIDH?

- Drop-in post-quantum replacement for (EC)DH.
- Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)
- Small keys: starts at 64 bytes.*
- Competitive speed: $\approx 55\,\mathrm{ms}$ / full key exchange.* (Skylake)

* Security evaluation is complicated, might get bigger & slower.

Why CSIDH?

- Drop-in post-quantum replacement for (EC)DH.
- Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)
- Small keys: starts at 64 bytes.*
- Competitive speed: $\approx$ 55 ms / full key exchange.* (Skylake)
- Flexible: compatible with 0-RTT protocols such as QUIC; yields signatures, (pre-quantum) VDFs, etc.

---

* Security evaluation is complicated, might get bigger & slower.

Stand back!

We’re going to do math.

Math slide #1: Elliptic curves *(nodes)*

An elliptic curve (modulo details) is given by an equation

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

A point on E is a solution to this equation *or* the 'fake' point ∞.

Math slide #1: Elliptic curves *(nodes)*

An elliptic curve (modulo details) is given by an equation

$$E\colon\ y^2 = x^3 + ax + b.$$

A point on E is a solution to this equation *or* the 'fake' point ∞.

E is an abelian group: we can 'add' and 'subtract' 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

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

do **not** remember these formulas!

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.

Math slide #2: Isogenies *(edges)*

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

- given by rational functions
- that is a group homomorphism.

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

Math slide #2: Isogenies *(edges)*

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

- given by rational functions
- that is a group homomorphism.

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

Example #1: For each $m \neq 0$, the multiplication-by-m map

$$[m]\colon E \to E$$

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

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

Math slide #2: Isogenies *(edges)*

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

- given by rational functions
- that is a group homomorphism.

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

Example #2: For any a and b, the map $\iota\colon (x, y) \mapsto (-x, \sqrt{-1} \cdot y)$ 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 $\{\infty\}$.

Math slide #2: Isogenies *(edges)*

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

- given by rational functions
- that is a group homomorphism.

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

Example #3: $(x, y) \mapsto \left(\frac{x^3-4x^2+30x-12}{(x-2)^2}, \frac{x^3-6x^2-14x+35}{(x-2)^3} \cdot y \right)$
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 $\{(2, 9), (2, -9), \infty\}$.

CSIDH in one slide

CSIDH in one slide

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

CSIDH in one slide

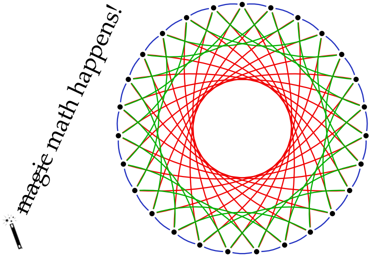
- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.

CSIDH in one slide

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

CSIDH in one slide

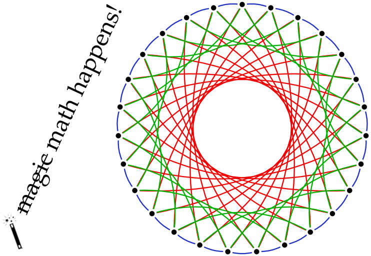
- Choose some small odd primes $\ell_1, \dots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



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

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 $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.

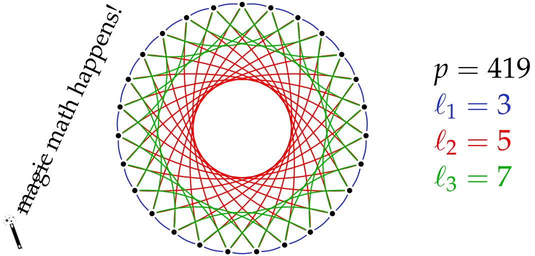


$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

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

CSIDH 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 $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



- Walking 'left' and 'right' on any ℓ_i-subgraph is efficient.
- We can represent $E \in X$ as a single coefficient $A \in \mathbb{F}_p$.

Walking in the CSIDH graph

Taking a 'positive' step on the ℓ_i-subgraph.

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   This uses scalar multiplication by $(p+1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Walking in the CSIDH graph

Taking a 'positive' step on the ℓ_i-subgraph.

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   This uses scalar multiplication by $(p+1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Taking a 'negative' step on the ℓ_i-subgraph.

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ <u>but $y \notin \mathbb{F}_p$</u>.
   This uses scalar multiplication by $(p+1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Walking in the CSIDH graph

Taking a 'positive' step on the ℓ_i-subgraph.

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   This uses scalar multiplication by $(p+1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Taking a 'negative' step on the ℓ_i-subgraph.

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ $\underline{\text{but } y \notin \mathbb{F}_p}$.
   This uses scalar multiplication by $(p+1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Upshot: With 'x-only arithmetic' everything happens over $\mathbb{F}_p$.
$\implies$ Efficient to implement!

Math slide #3: Isogenies and kernels

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

The curve E' is called E/G. ($\approx$ 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')

Math slide #3: Isogenies and kernels

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

The curve E' is called E/G. (≈ quotient groups)

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

Vélu '71:
Formulas for computing E/G and evaluating φ_G at a point.

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

[1](up to isomorphism of E')

Math slide #3: Isogenies and kernels

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

The curve E' is called E/G. (≈ quotient groups)

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

Vélu '71:
Formulas for computing E/G and evaluating φ_G at a point.

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

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

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

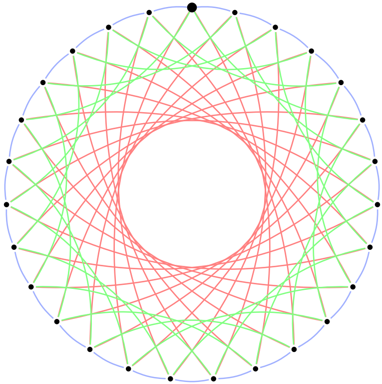
$\rightsquigarrow$ this is why we use special p and curves with $p+1$ points!

[1](up to isomorphism of E')

CSIDH key exchange

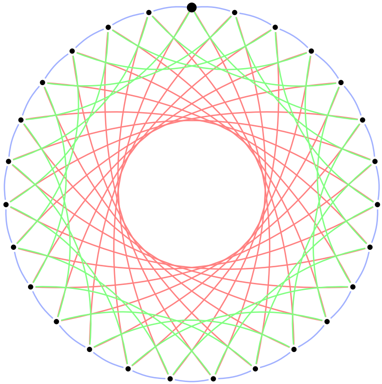
Alice

[+, +, −, −]



Bob

[−, +, −, −]



CSIDH key exchange

Alice

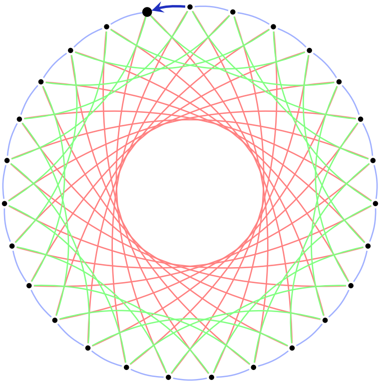
[+, +, −, −]

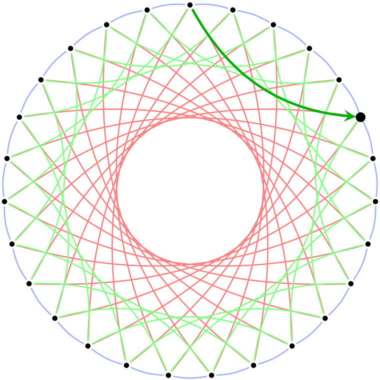
↑

Bob

[−, +, −, −]

↑

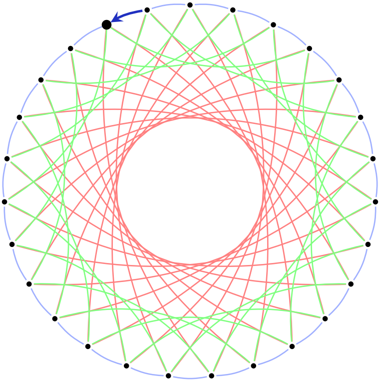




CSIDH key exchange

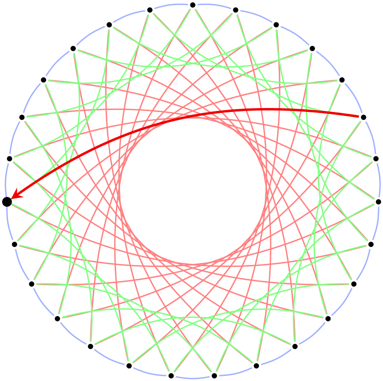
Alice

$[\textcolor{blue}{+}, \underset{\uparrow}{\textcolor{blue}{+}}, \textcolor{red}{-}, \textcolor{green}{-}]$



Bob

$[\textcolor{green}{-}, \underset{\uparrow}{\textcolor{red}{+}}, \textcolor{green}{-}, \textcolor{blue}{-}]$

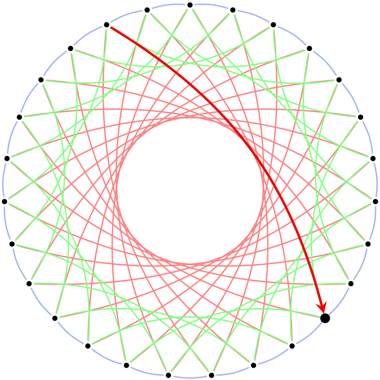


CSIDH key exchange

Alice

[+, +, −, −]

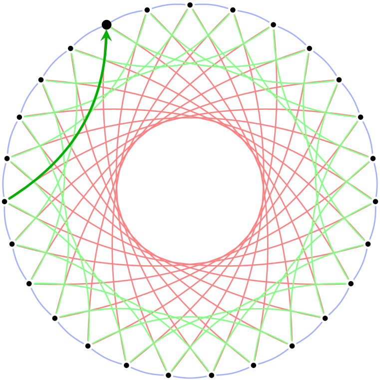
↑



Bob

[−, +, −, −]

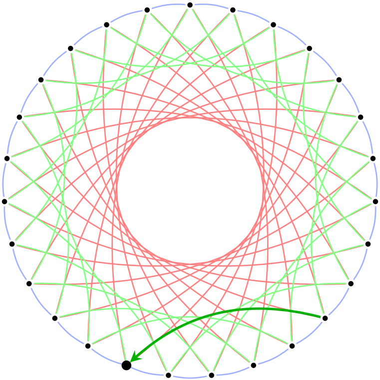
↑



CSIDH key exchange

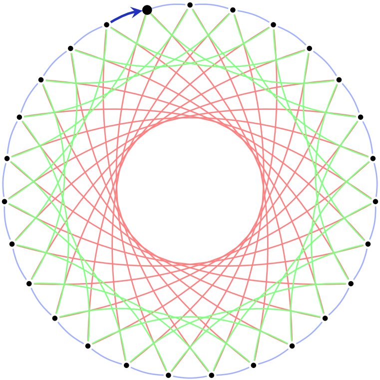
Alice

[+, +, −, −]

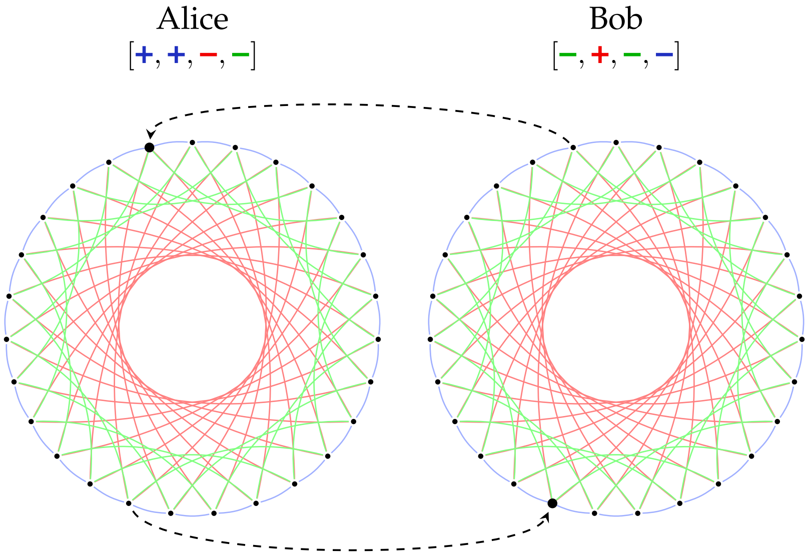


Bob

[−, +, −, −]



CSIDH key exchange



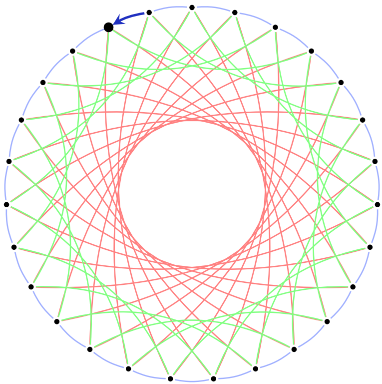
CSIDH key exchange

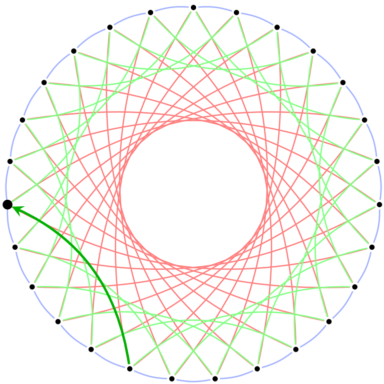
Alice

[+, +, −, −]

Bob

[−, +, −, −]





CSIDH key exchange

Alice

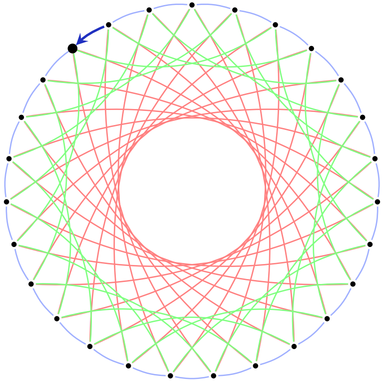
[+, +, −, −]

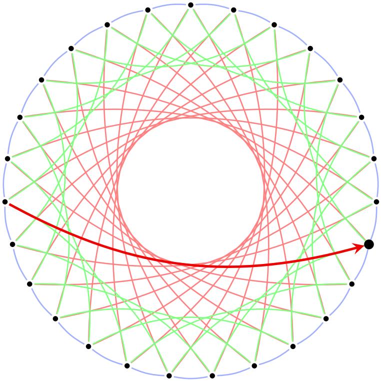
↑

Bob

[−, +, −, −]

↑



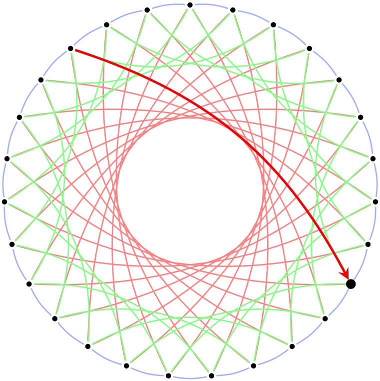


CSIDH key exchange

Alice

[+, +, −, −]

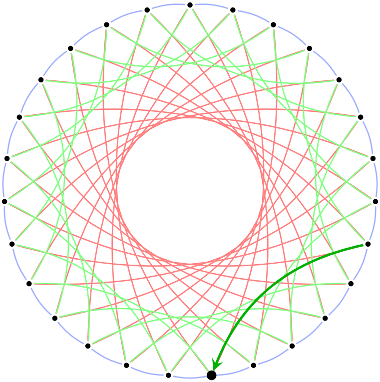
↑



Bob

[−, +, −, −]

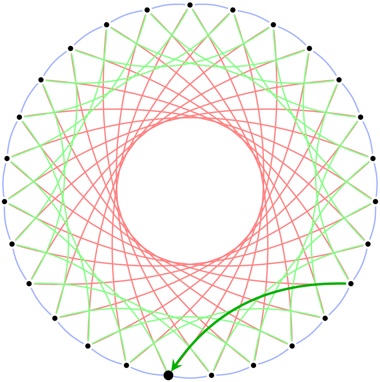
↑



CSIDH key exchange

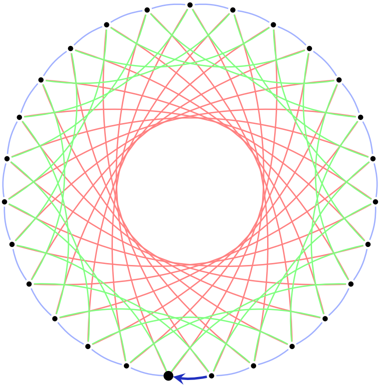
Alice

$[+, +, -, -]$



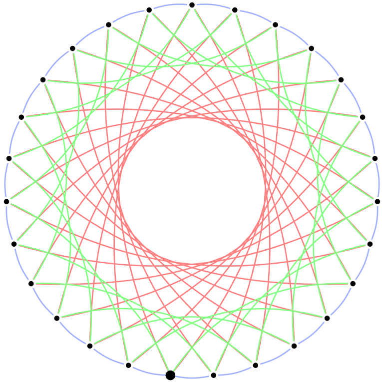
Bob

$[-, +, -, -]$

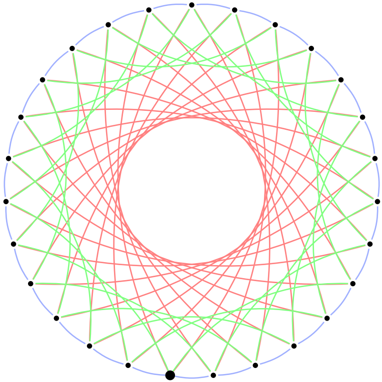


CSIDH key exchange

Alice
[+, +, −, −]



Bob
[−, +, −, −]



Has anyone seen my group action?

"CSIDH: an efficient post-quantum
<u>commutative group action</u>"

Has anyone seen my group action?

"CSIDH: an efficient post-quantum commutative 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 $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

Has anyone seen my group action?

"CSIDH: an efficient post-quantum commutative 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!

Has anyone seen my group action?

"CSIDH: an efficient post-quantum commutative 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!

Many paths are 'useless'. *Fun fact:* Quotienting out trivial actions yields the ideal-class group $\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$.

Cryptographic group actions

Like in the CSIDH example, we *generally* get a DH-like key exchange from a commutative group action $G \times S \to S$:

Alice	public	Bob
$a \xleftarrow{\text{random}} G$		$b \xleftarrow{\text{random}} G$
$a * s$		$b * s$
$\mathit{key} := a * (b * s)$		$\mathit{key} := b * (a * s)$

Why no Shor?

Recall from Dan's talk:

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 general group actions, we 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 $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) \approx \sqrt{p}$.

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

Security of CSIDH

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

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

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

Solving abelian hidden shift breaks CSIDH.

$\rightsquigarrow$ quantum subexponential attack (Kuperberg's algorithm).

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?)

$\Longrightarrow$ It is still rather unclear how to choose CSIDH parameters.

...but all known attacks cost $\exp\big((\log p)^{1/2+o(1)}\big)$!

Can we avoid Kuperberg's algorithm?

With great commutative group action
comes great subexponential attack.

Can we avoid Kuperberg's algorithm?

With great commutative group action
comes great subexponential attack.

The supersingular isogeny graph over $\mathbb{F}_{p^2}$ has less structure.

- **SIDH** uses the full $\mathbb{F}_{p^2}$-isogeny graph. No group action!

Can we avoid Kuperberg's algorithm?

With great commutative group action
comes great subexponential attack.

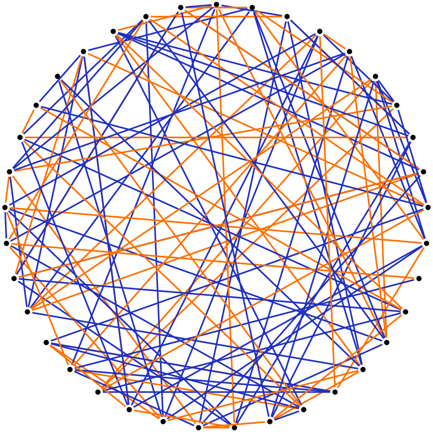
The supersingular isogeny graph over $\mathbb{F}_{p^2}$ has less structure.

- **SIDH** uses the full $\mathbb{F}_{p^2}$-isogeny graph. No group action!

- Problem: also no more intrinsic sense of direction.

"It all bloody looks the same!" — a famous isogeny cryptographer

$\rightsquigarrow$ need extra information to let Alice & Bob's walks commute.



Now: SIDH (Jao, De Feo; 2011)

(...whose name doesn't allow for nice pictures of beaches...)

Wikipedia about SIDH...

"While several steps of SIDH involve complex isogeny calculations, the overall flow of SIDH for parties A and B is straightforward for those familiar with a Diffie–Hellman key exchange or its elliptic curve variant. [...]

Wikipedia about SIDH...

"While several steps of SIDH involve complex isogeny calculations, the overall flow of SIDH for parties A and B is straightforward for those familiar with a Diffie–Hellman key exchange or its elliptic curve variant. [...]

Setup.

1. A prime of the form $p = w_A^{e_A} \cdot w_B^{e_B} \cdot f \pm 1$.
2. A supersingular elliptic curve E over $\mathbb{F}_{p^2}$.
3. Fixed elliptic points P_A, Q_A, P_B, Q_B on E.
4. The order of P_A and Q_A is $(w_A)^{e_A}$.
5. The order of P_B and Q_B is $(w_B)^{e_B}$.

Key exchange. [...]

1A. A generates two random integers $m_A, n_A < (w_A)^{e_A}$.
2A. A generates $R_A := m_A \cdot (P_A) + n_A \cdot (Q_A)$.
3A. A uses the point R_A to create an isogeny mapping $\phi_A : E \to E_A$ and curve E_A isogenous to E.
4A. A applies ϕ_A to P_B and Q_B to form two points on E_A : $\phi_A(P_B)$ and $\phi_A(Q_B)$.
5A. A sends to B E_A, $\phi_A(P_B)$, and $\phi_A(Q_B)$.
1B–4B. Same as A1 through A4, but with A and B subscripts swapped.
5B. B sends to A E_B, $\phi_B(P_A)$, and $\phi_B(Q_A)$.
6A. A has $m_A, n_A, \phi_B(P_A)$, and $\phi_B(Q_A)$ and forms $S_{BA} := m_A(\phi_B(P_A)) + n_A(\phi_B(Q_A))$.
7A. A uses S_{BA} to create an isogeny mapping ψ_{BA}.
8A. A uses ψ_{BA} to create an elliptic curve E_{BA} which is isogenous to E.
9A. A computes $K :=$ j-invariant (j_{BA}) of the curve E_{BA}.
6B. Similarly, B has $m_B, n_B, \phi_A(P_B)$, and $\phi_A(Q_B)$ and forms $S_{AB} = m_B(\phi_A(P_B)) + n_B(\phi_A(Q_B))$.
7B. B uses S_{AB} to create an isogeny mapping ψ_{AB}.
8B. B uses ψ_{AB} to create an elliptic curve E_{AB} which is isogenous to Ek
9B. B computes $K :=$ j-invariant (j_{AB}) of the curve E_{AB}.

The curves E_{AB} and E_{BA} are guaranteed to have the same j-invariant."

SIDH: High-level view

$$\begin{tikzcd}
E \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E/A \arrow[d, "\varphi_{B'}"] \\
E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle
\end{tikzcd}$$

SIDH: High-level view

$$\begin{tikzcd} E \arrow[r, "\varphi_{A}"] \arrow[d, "\varphi_{B}"'] & E/A \arrow[d, "\varphi_{B'}"] \\ E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle \end{tikzcd}$$

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

SIDH: High-level view

$$\begin{tikzcd} E \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E/A \arrow[d, "\varphi_{B'}"] \\ E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle \end{tikzcd}$$

- 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

$$\begin{tikzcd} E \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E/A \arrow[d, "\varphi_{B'}"] \\ E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle \end{tikzcd}$$

- 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

$$\begin{tikzcd}
E \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E/A \arrow[d, "\varphi_{B'}"] \\
E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle
\end{tikzcd}$$

- 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 <u>somehow</u> obtains $A' := \varphi_B(A)$. (Similar for Bob.)

SIDH: High-level view

$$\begin{tikzcd} E \arrow[r, "\varphi_A"] \arrow[d, "\varphi_B"'] & E/A \arrow[d, "\varphi_{B'}"] \\ E/B \arrow[r, "\varphi_{A'}"'] & E/\langle A, B\rangle \end{tikzcd}$$

- 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 <u>somehow</u> 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

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

Alice knows only A, Bob knows only φ_B. Hm.

SIDH's auxiliary points

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

Alice knows only A, Bob knows only φ_B. Hm.

- 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.

SIDH's auxiliary points

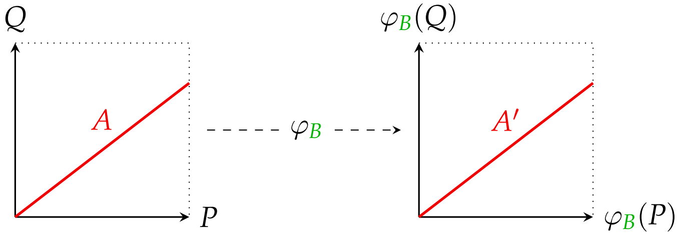
Previous slide: "Alice somehow obtains $A' := \varphi_B(A)$."

Alice knows only A, Bob knows only φ_B. Hm.

Solution: φ_B is a group homomorphism!

- 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 $\langle \varphi_B(P) + [a]\varphi_B(Q) \rangle$!



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) and (R, S) of $E[2^n]$ and $E[3^m]$ (recall $E[k] \cong \mathbb{Z}/k \times \mathbb{Z}/k$)

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

Decomposing smooth isogenies

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

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 $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.

$$\begin{tikzcd} E \arrow[r, "\psi_1"] \arrow[rrrr, bend right, "\varphi_G"'] & E_1 \arrow[r, "\psi_2"] & \ldots \arrow[r, "\psi_{k-1}"] & E_{k-1} \arrow[r, "\psi_k"] & E/G \end{tikzcd}$$

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 $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.

$$\begin{tikzcd} E \arrow[r, "\psi_1"] \arrow[rrrr, bend right, "\varphi_G"'] & E_1 \arrow[r, "\psi_2"] & \dots \arrow[r, "\psi_{k-1}"] & E_{k-1} \arrow[r, "\psi_k"] & E/G \end{tikzcd}$$

$\rightsquigarrow$ Complexity: $O(k^2 \cdot \ell)$. Exponentially smaller than ℓ^k!
'Optimal strategy' improves this to $O(k \log k \cdot \ell)$.

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 $\ker \psi_i := [\ell^{k-i}](\psi_{i-1} \circ \cdots \circ \psi_1)(G)$.

$$\begin{tikzcd} E \arrow[r, "\psi_1"] \arrow[rrrr, bend right, "\varphi_G"'] & E_1 \arrow[r, "\psi_2"] & \dots \arrow[r, "\psi_{k-1}"] & E_{k-1} \arrow[r, "\psi_k"] & E/G \end{tikzcd}$$

$\rightsquigarrow$ Complexity: $O(k^2 \cdot \ell)$. Exponentially smaller than ℓ^k!
'Optimal strategy' improves this to $O(k \log k \cdot \ell)$.

- BTW: The choice of p makes sure everything stays over $\mathbb{F}_{p^2}$.

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Each secret isogeny φ_A, φ_B is a walk of about $\log p/2$ steps.
(Alice & Bob can choose from about $\sqrt{p}$ secret keys each.)

Security of SIDH

The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$.
Each secret isogeny φ_A, φ_B is a walk of about $\log p/2$ steps.
(Alice & Bob can choose from about $\sqrt{p}$ secret keys each.)

Classical attacks:

- Cannot reuse keys without extra caution. (next slide)
- 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 $\lfloor p/12 \rfloor + \varepsilon$.
Each secret isogeny φ_A, φ_B is a walk of about $\log p/2$ steps.
(Alice & Bob can choose from about $\sqrt{p}$ secret keys each.)

Classical attacks:

- Cannot reuse keys without extra caution. (next slide)
- 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 $\tilde{\mathcal{O}}(p^{1/6})$. Newer paper says $\tilde{\mathcal{O}}(p^{1/4})$:

  "An adversary with enough quantum memory to run Tani's algorithm with the query-optimal parameters could break SIKE faster by using the classical control hardware to run van Oorschot–Wiener."

Thou shalt not reuse SIDH keys

- 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.

Thou shalt not reuse SIDH keys

- 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 $A'' = \langle P' + [a]Q'' \rangle$.

Thou shalt not reuse SIDH keys

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

Thou shalt not reuse SIDH keys

- 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 parity of a.

Thou shalt not reuse SIDH keys

- 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 parity of a.
  Similarly, he can completely recover a in $O(n)$ queries.

Thou shalt not reuse SIDH keys

- 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 parity 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*'.


Questions?