Isogeny-Based Cryptography

Tanja Lange
(with lots of slides by Lorenz Panny)

Eindhoven University of Technology

20 & 21 July 2020

Diffie–Hellman key exchange '76

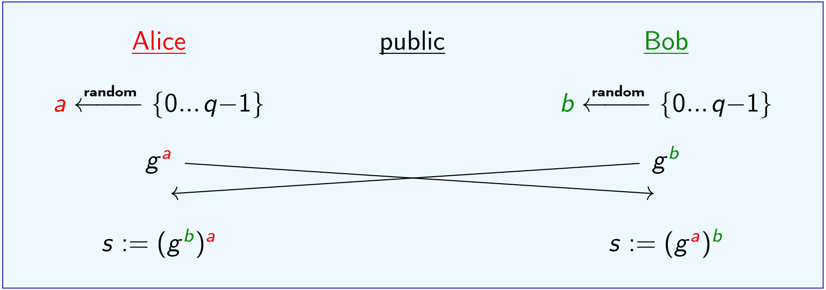
Public parameters:

- 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



Alice | public | Bob

$a \xleftarrow{\text{random}} \{0 \dots q-1\}$ | $b \xleftarrow{\text{random}} \{0 \dots q-1\}$

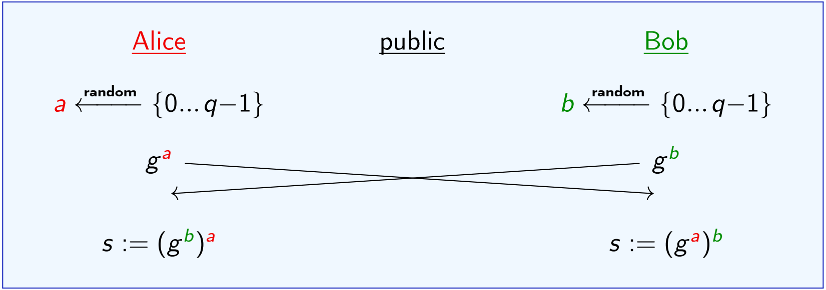
g^a | g^b

$s := (g^b)^a$ | $s := (g^a)^b$

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 $\cdot^b$ commute!

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

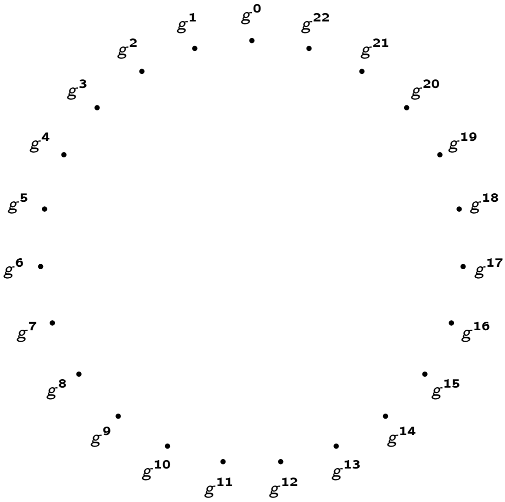
Bob	**Attacker** Eve
1. Set $t \leftarrow g$.	1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$.	2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$.	3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$.	4.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

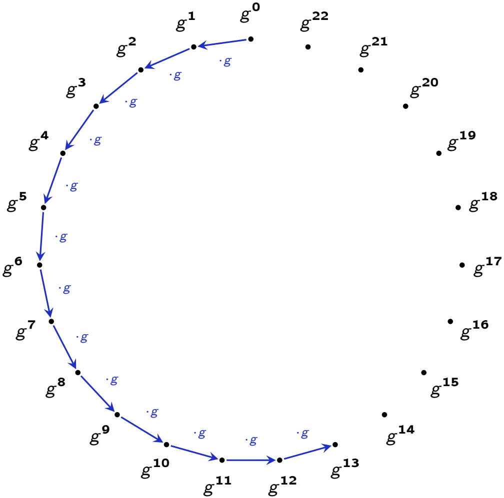
Effort for both: $O(\#G)$. Bob needs to be smarter.

(There also exist better attacks)



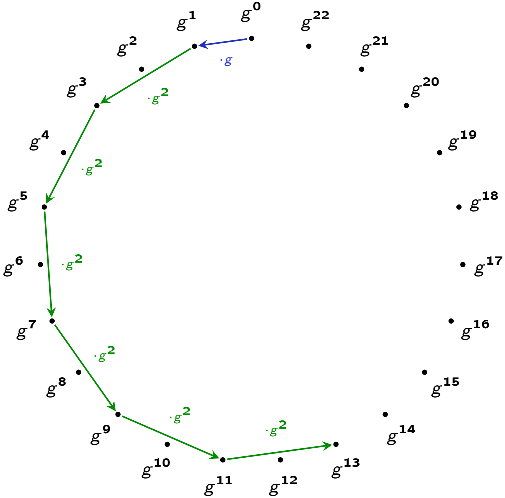
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

multiply



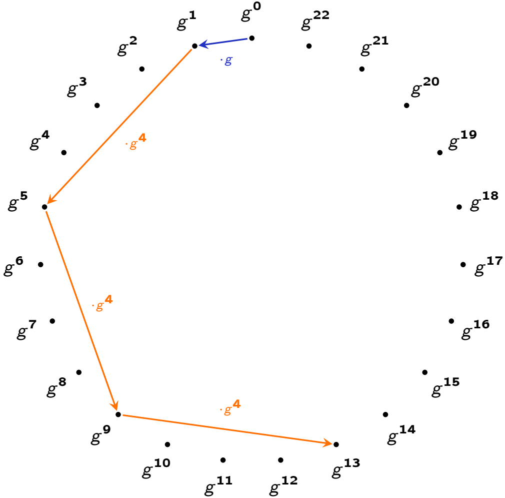
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply



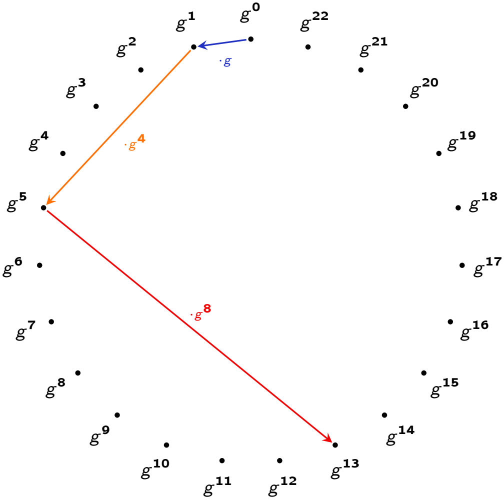
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

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



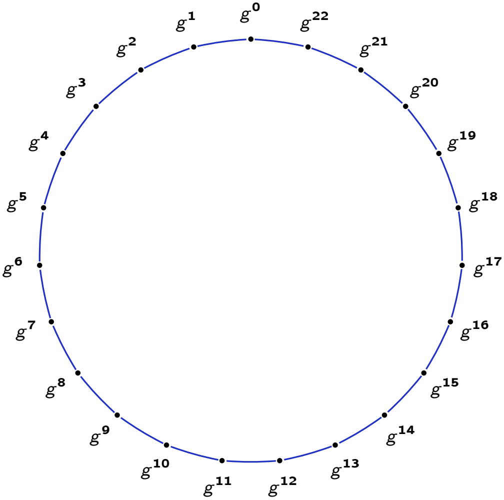
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

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



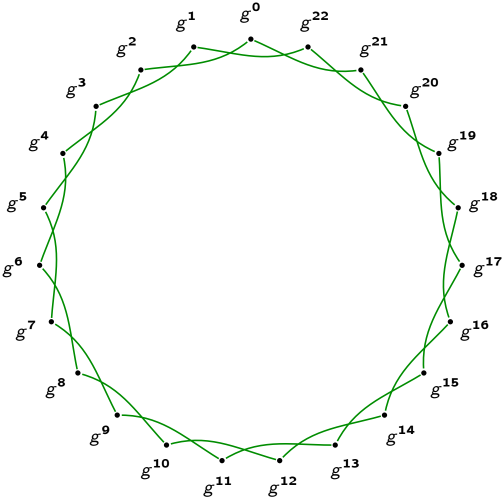
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



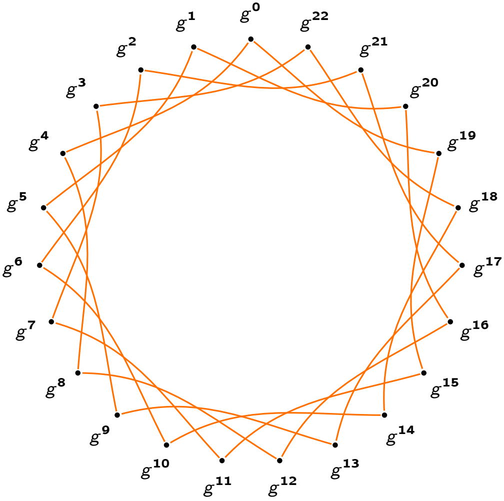
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



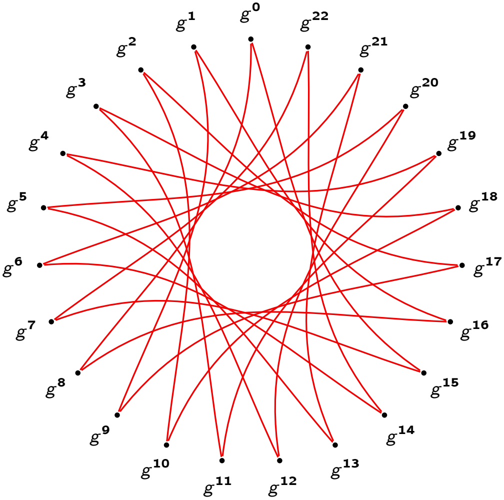
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



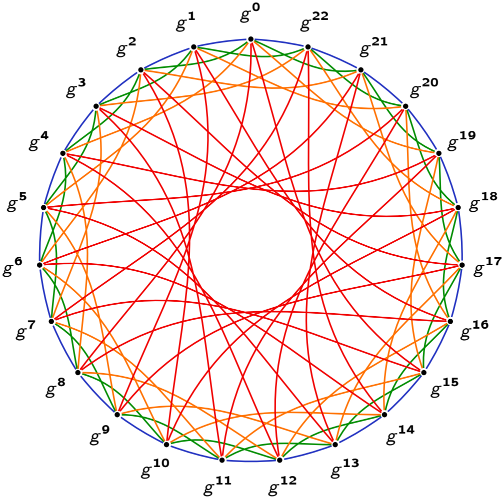
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



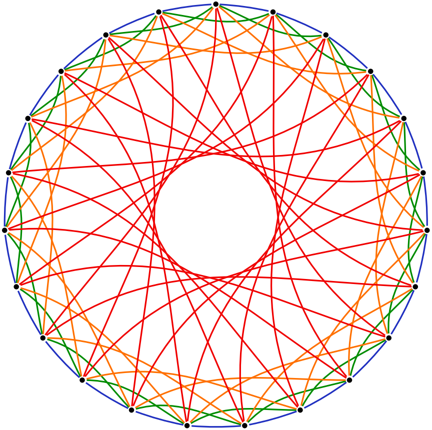
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



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

Exponential separation

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

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

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

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

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b in any group in polynomial time.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

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

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b in any group in polynomial time.
Isogeny graphs to the rescue!

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!

Topic of this lecture

- Isogenies are well-behaved maps between elliptic curves.

Topic of this lecture

- Isogenies are well-behaved maps between elliptic curves.

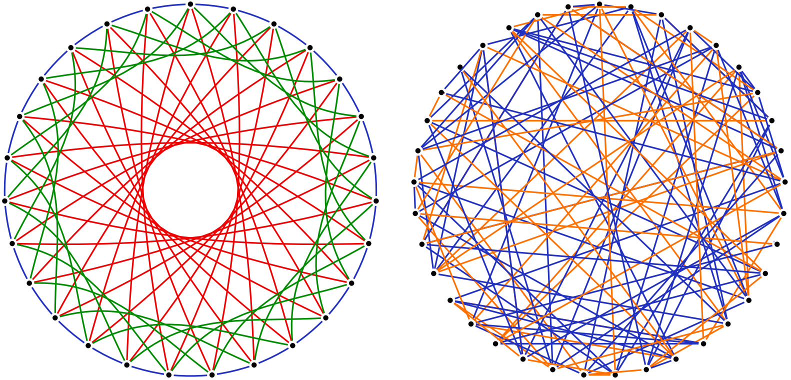
⇝ Isogeny graph: Nodes are curves, edges are isogenies.

(We usually care about subgraphs with certain properties.)

- Isogenies give rise to post-quantum Diffie–Hellman
(and more!)

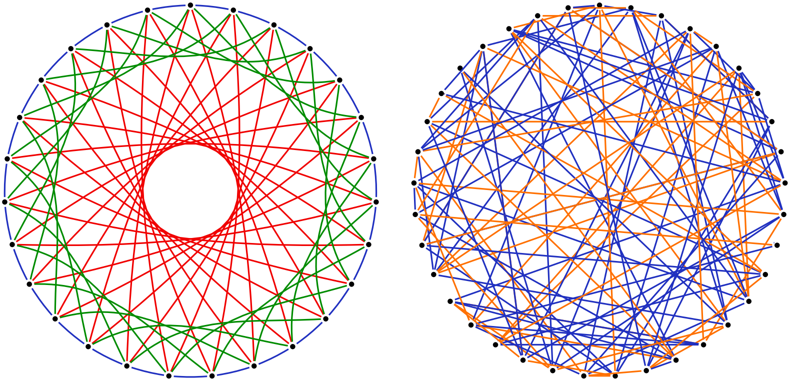
The beauty and the beast

Components of well-chosen isogeny graphs look like this:



The beauty and the beast

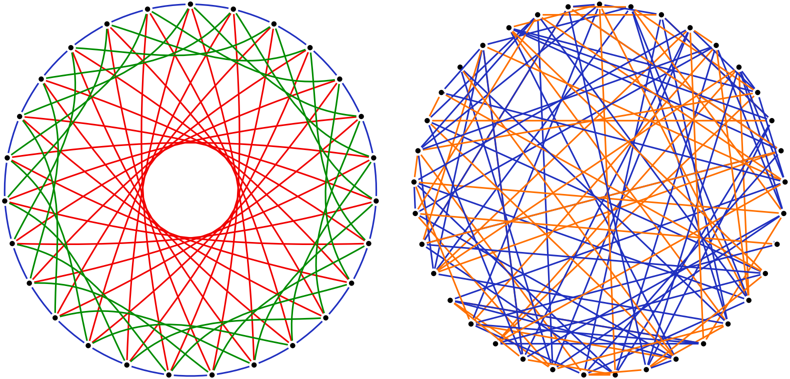
Components of well-chosen isogeny graphs look like this:



Which of these is good for crypto?

The beauty and the beast

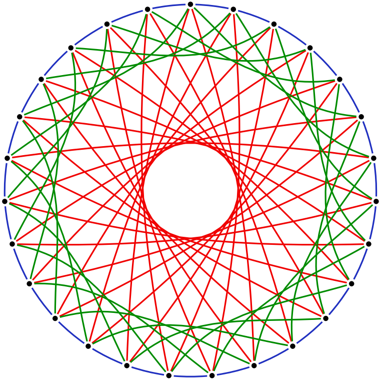
Components of well-chosen isogeny graphs look like this:



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

The beauty and the beast

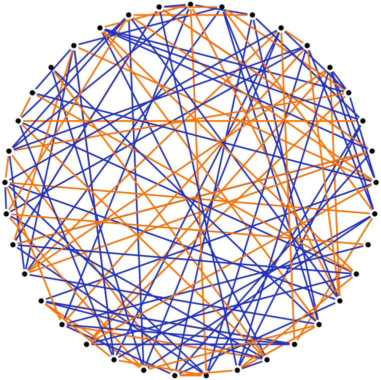
At this time, there are two distinct families of systems:



$q = p$

CSIDH [ˈsiːˌsaɪd]

https://csidh.isogeny.org



$q = p^2$

SIDH

https://sike.org

CSIDH [ˈsiːˌsaɪd]

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

Why CSIDH?

- Closest thing we have in PQC to normal DH key exchange: Keys can be reused, blinded; no difference between initiator &responder.
- Public keys are represented by some $A \in \mathbb{F}_p$; p fixed prime.
- Alice computes and distributes her public key A.
  Bob computes and distributes his public key B.
- Alice and Bob do computations on each other's public keys to obtain shared secret.
- Fancy math: computations start on some elliptic curve $E_A : y^2 = x^3 + Ax^2 + x$, use isogenies to move to a different curve.
- Computations need arithmetic (add, mult, div) modulo p and elliptic-curve computations.

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

An elliptic curve over $\mathbb{F}_p$ is given by an equation

$$E: \ y^2 = x^3 + ax + b, \text{ with } 4a^3 - 27b^2 \neq 0.$$

A point $P = (x, y)$ on E is a solution to this equation
or the point ∞ at infinity.

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

An elliptic curve over $\mathbb{F}_p$ is given by an equation

$$E: \; y^2 = x^3 + ax + b, \text{ with } 4a^3 - 27b^2 \neq 0.$$

A point $P = (x, y)$ on E is a solution to this equation
or the point ∞ at infinity.

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 $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ is $P_3 = (x_3, y_3) =$
  $$(\lambda^2 - x_1 - x_2, \; \lambda(x_1 - x_3) - y_1)$$
  where $\lambda = (y_2 - y_1)/(x_2 - x_1)$ if $x_1 \neq x_2$
  and $\lambda = (3x_1^2 + a)/(2y_1)$ if $P_1 = P_2 \neq -P_1$.

Takeaway: Computations in $\mathbb{F}_p$, some formulas.
Other curve shapes, such as Montgomery curves $y^2 = x^3 + Ax^2 + x$ are faster.

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

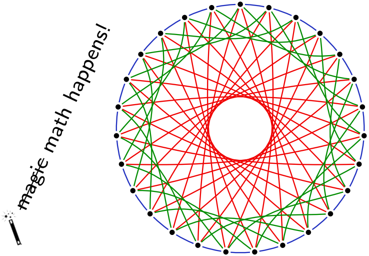
- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.

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

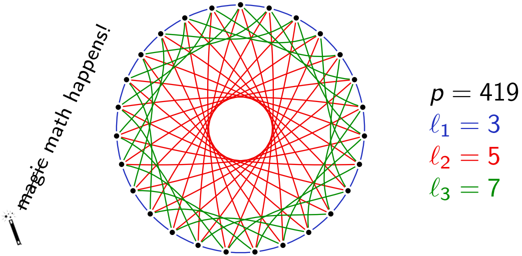
- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



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

CSIDH in one slide

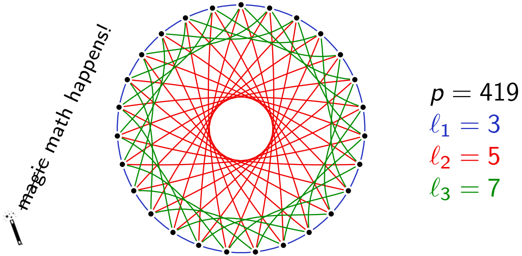
- Choose some small odd primes $\ell_1, \ldots, \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



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

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.



- 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 $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
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 $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p+1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
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).

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 $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
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 $\mathbb{F}_p$.

$\Longrightarrow$ 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. ($\approx$ 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!

Not all k-rational points of E/G are in the image of k-rational points on E; but $\#E(k)\ \#E/G(k)$.

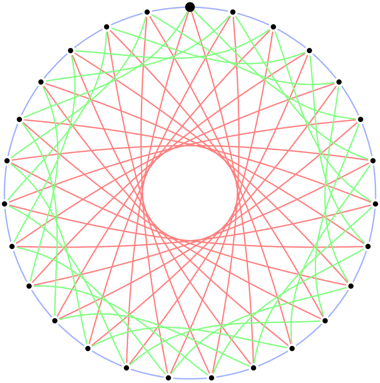
---

[1] (up to isomorphism of E')

CSIDH key exchange

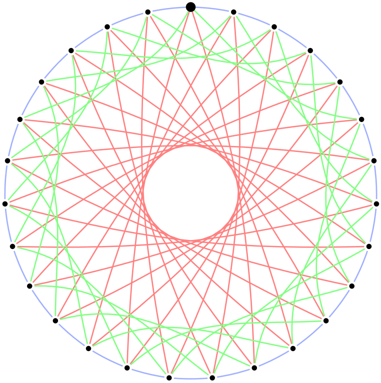
Alice

[+, +, −, −]



Bob

[−, +, −, −]



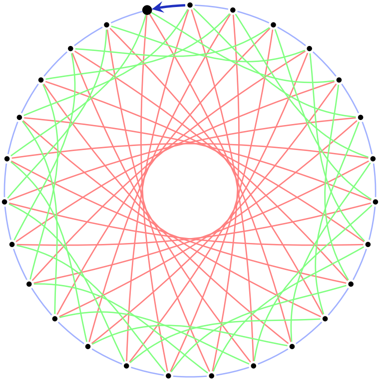
CSIDH key exchange

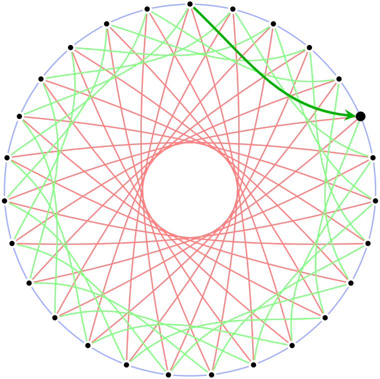
Alice

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

Bob

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

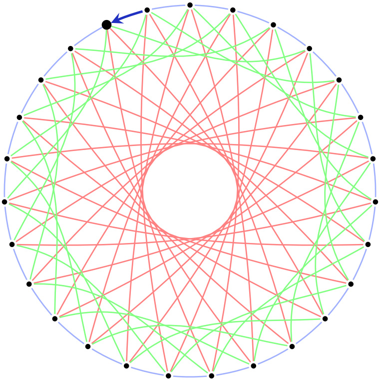




CSIDH key exchange

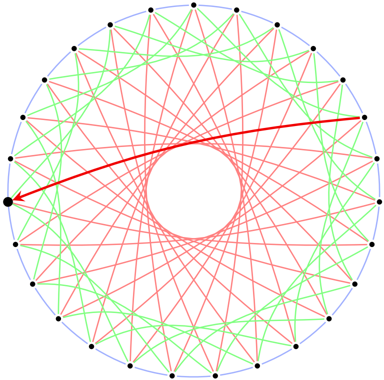
Alice

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



Bob

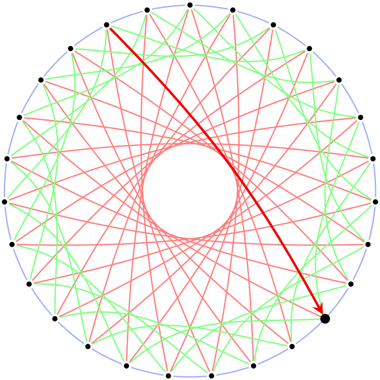
$[-, \underset{\uparrow}{+}, -, -]$



CSIDH key exchange

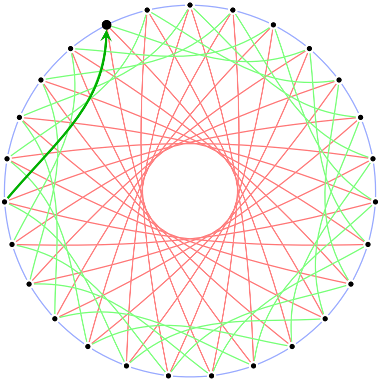
Alice

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



Bob

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



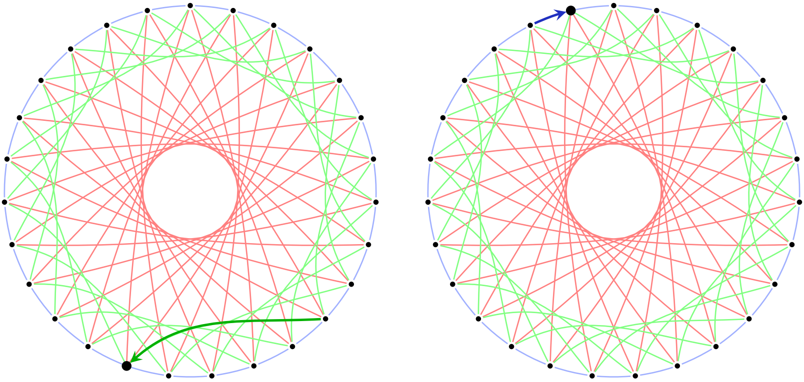
CSIDH key exchange

Alice

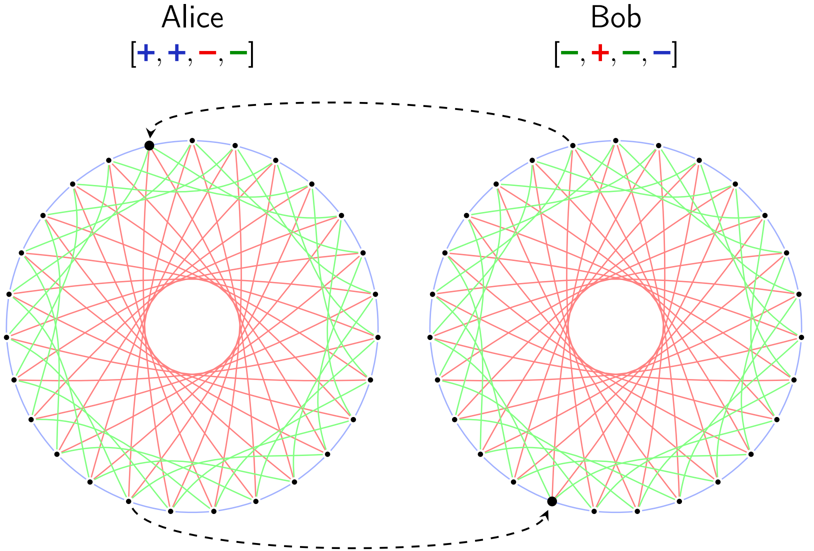
$[+, +, -, \underset{\uparrow}{-}]$

Bob

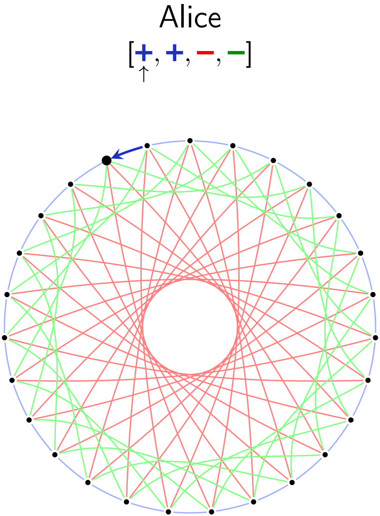
$[-, +, -, \underset{\uparrow}{-}]$

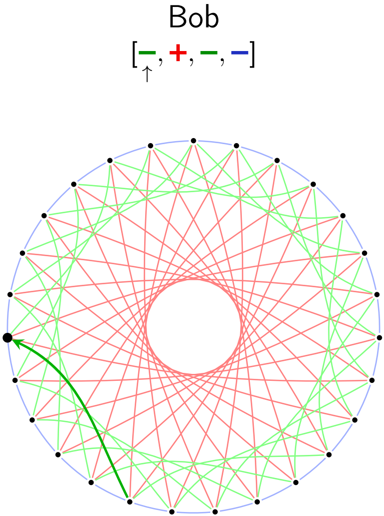


CSIDH key exchange



CSIDH key exchange

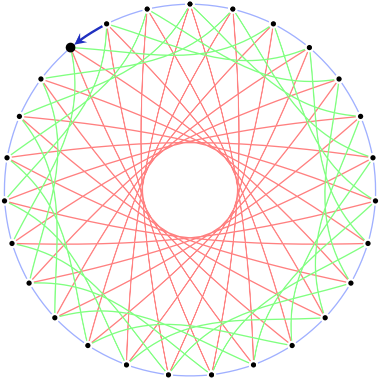




CSIDH key exchange

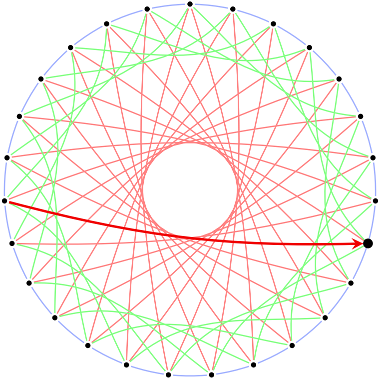
Alice

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



Bob

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

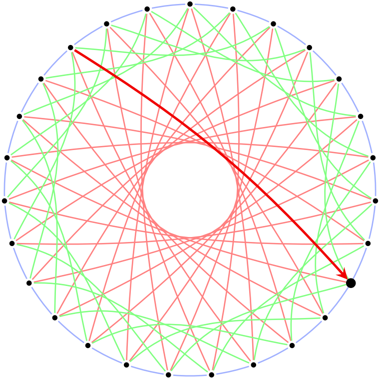


CSIDH key exchange

Alice

[+, +, −, −]

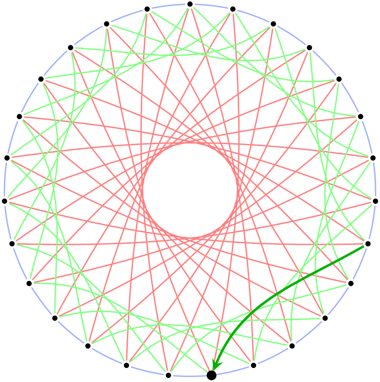
↑



Bob

[−, +, −, −]

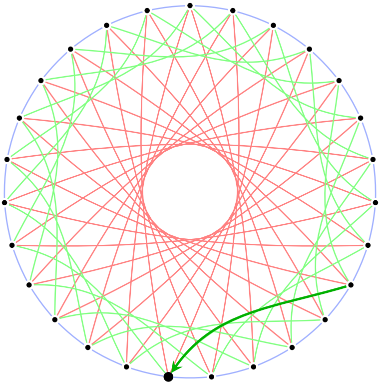
↑



CSIDH key exchange

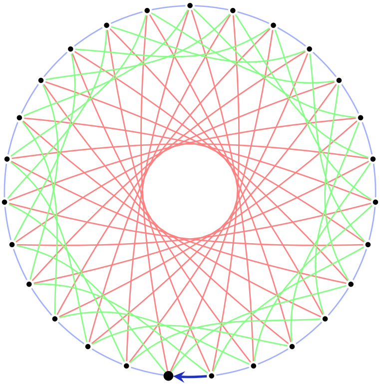
Alice

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



Bob

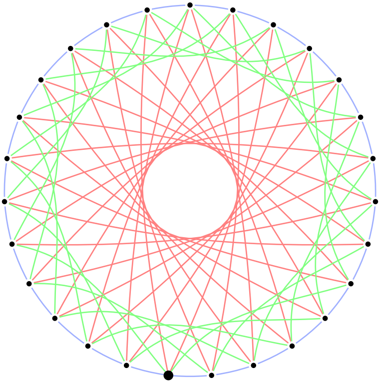
$[-, +, -, \underset{\uparrow}{-}]$



CSIDH key exchange

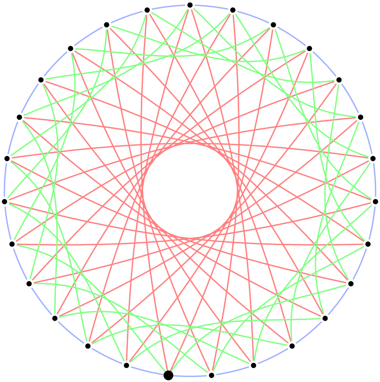
Alice

$[+, +, -, -]$



Bob

$[-, +, -, -]$



Abstract from Diffie-Hellman dataflow

"CSIDH: an efficient post-quantum
commutative group action"

Abstract from Diffie-Hellman dataflow

"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 $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

Abstract from Diffie-Hellman dataflow

"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: $[\mathbf{+},\mathbf{+},\mathbf{-},\mathbf{-},\mathbf{-},\mathbf{+},\mathbf{-},\mathbf{-}]$ 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!

Abstract from Diffie-Hellman dataflow

"CSIDH: an efficient post-quantum
commutative group action"

Cycles are compatible: [right then left] = [left then right]
⇝ 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 $\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$.

Math slide #4: Quadratic twists Not my fault …

E'/k is a twist elliptic curve E''/k if E is isomorphic to E' over $\bar{k}$.

For $E: y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$
$E': -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a quadratic twist.

Math slide #4: Quadratic twists

Not my fault . . .

E'/k is a twist elliptic curve E''/k if E is isomorphic to E' over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$
$E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a quadratic twist.

Picking (x, y) on E with $x \in \mathbb{F}_p, y \neq \mathbb{F}_p$ implicitly picks point in $E'(\mathbb{F}_p)$.

Math slide #4: Quadratic twists Not my fault . . .

E'/k is a twist elliptic curve $E"/k$ if E is isomorphic to E' over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$
$E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

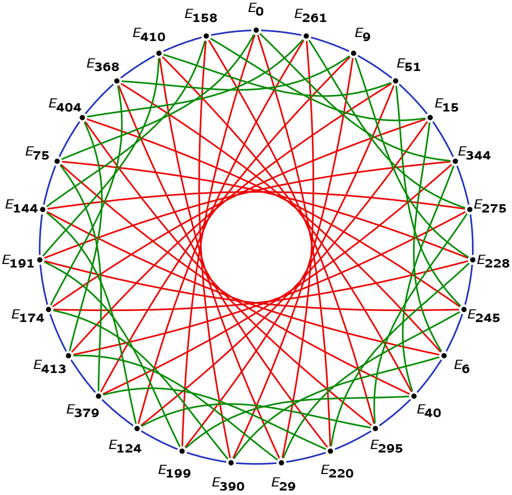
This map is defined over $\mathbb{F}_{p^2}$, so this is a quadratic twist.

Picking (x, y) on E with $x \in \mathbb{F}_p, y \neq \mathbb{F}_p$ implicitly picks point in $E'(\mathbb{F}_p)$.

E' is not in the isogeny graph, does not have the right shape.

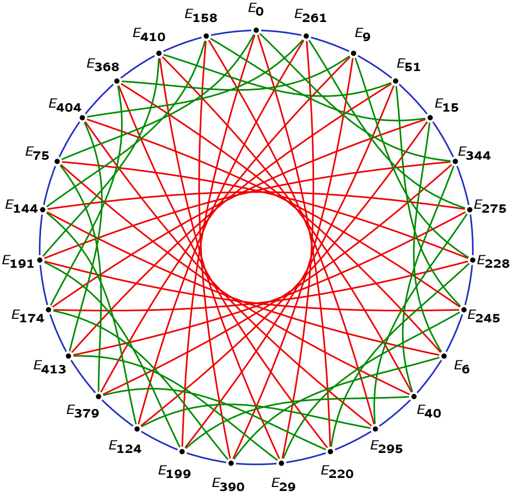
E' is isomorphic to $E'' : y^2 = x^3 - Ax^2 + x$ via $(x, y) \mapsto (-x, y)$ over $\mathbb{F}_p$.

Graphs of elliptic curves



Nodes: Supersingular elliptic curves $E_A\colon y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.

Graphs of elliptic curves



Nodes: Supersingular elliptic curves E_A: $y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
Each E_A on the left has E_{-A} on the right.
Negative direction means: flip to twist, go positive direction, flip back.

Math slide #5: Vélu's formulas

Let P have prime order ℓ on E_A.
For $1 \leq k < \ell$ let x_k be the x-coordinate of $[k]P$.
Let

$$\tau = \prod_{i=1}^{\ell-1} x_i, \quad \sigma = \sum_{i=1}^{\ell-1} \left(x_i - \frac{1}{x_i} \right)$$

Then the ℓ isogeny from E_A maps to E_B with $B = \tau(A - 3\sigma)$.

Main operation is to compute the x_k, just some elliptic-curve additions.

Note that $[\ell - k]P = -[k]P$ and both have the same x-coordinate.

Implementations often use projective formulas to avoid (or delay) inverstions.

Math slide #6: Class groups

Reminder: $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\Longleftrightarrow$ action of $\mathfrak{l}_i$ on X.

Math slide #6: Class groups

Reminder: $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\iff$ action of $\mathfrak{l}_i$ on X.

More precisely:
Subgroup corresponding to $\mathfrak{l}_i$ is $E[\mathfrak{l}_i] = E(\mathbb{F}_p)[\ell_i]$.
(Note that $\ker(\pi - 1)$ is just the $\mathbb{F}_p$-rational points!)

Subgroup corresponding to $\overline{\mathfrak{l}_i}$ is

$$E[\overline{\mathfrak{l}_i}] = \{P \in E[\ell_i] \mid \pi(P) = -P\}.$$

Math slide #6: Class groups

Reminder: $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\Longleftrightarrow$ action of $\mathfrak{l}_i$ on X.

More precisely:
Subgroup corresponding to $\mathfrak{l}_i$ is $E[\mathfrak{l}_i] = E(\mathbb{F}_p)[\ell_i]$.
(Note that $\ker(\pi - 1)$ is just the $\mathbb{F}_p$-rational points!)

Subgroup corresponding to $\overline{\mathfrak{l}_i}$ is

$$E[\overline{\mathfrak{l}_i}] = \{P \in E[\ell_i] \mid \pi(P) = -P\}.$$

For Montgomery curves,

$$E[\overline{\mathfrak{l}_i}] = \{(x, y) \in E[\ell_i] \mid x \in \mathbb{F}_p;\ y \notin \mathbb{F}_p\} \cup \{\infty\}.$$

Math slide #7: Commutative group action

$\mathrm{cl}(\mathcal{O})$ acts on X. For most ideal classes the kernel is big and formulas are expensive to compute.

$$I = \mathfrak{l}_1^{10}\mathfrak{l}_2^{-7}\mathfrak{l}_3^{27}$$

is a “big” ideal, but we can compute the action iteratively.

$\mathrm{cl}(\mathcal{O})$ is commutative[2] so we get a commutative group action..

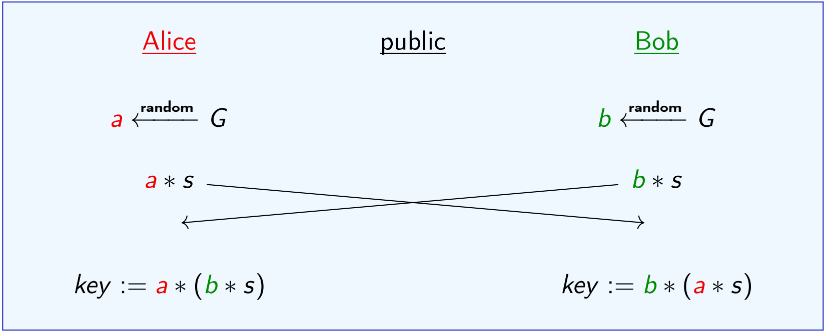
The choice for CSIDH:
Let $K = \{[\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_1}] \mid (e_1, ..., e_n) \text{ is 'short'}\} \subseteq \mathrm{cl}(\mathcal{O})$.
The action of K on X is very efficient!
Pick K as the keyspace

---

[2] Important to use the $\mathbb{F}_p$-endomorphism ring.

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

$a \xleftarrow{\text{random}} G$

$b \xleftarrow{\text{random}} G$

$a * s$

$b * s$

$key := a * (b * s)$

$key := b * (a * s)$

Why no Shor?

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

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

For general group actions, we cannot compose $x * s$ and $y * (b * s)$.

For CSIDH this would require composing two elliptic curves in some form compatible with the action of G.

CSIDH security

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

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
  (More precisely $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ keys.)

Without quantum computer:

- Meet-in-the-middle variants: Time $O(\sqrt[4]{p})$.
  (2016 Delfs–Galbraith)

CSIDH security

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

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
  (More precisely $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ keys.)

Without quantum computer:

- Meet-in-the-middle variants: Time $\mathrm{O}(\sqrt[4]{p})$.
  (2016 Delfs–Galbraith)

With quantum computer:

- Abellian hidden-shift algorithms apply
  (2014 Childs–Jao–Soukharev)
  - Kuperberg's algoirhtm has subexponential complexity.

CSIDH security:

- Public-key validation:
  Quickly check that $E_A : y^2 = x^3 + Ax^2 + x$ has $p + 1$ points.

CSIDH-512 `https://csidh.isogeny.org/`

Definition:

- $p = \prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \dots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes (just one $\mathbb{F}_p$ element).

Performance on typical Intel Skylake laptop core:

- Clock cycles: about $12 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.

CSIDH-512 `https://csidh.isogeny.org/`

Definition:

- $p = \prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \ldots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes (just one $\mathbb{F}_p$ element).

Performance on typical Intel Skylake laptop core:

- Clock cycles: about $12 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.
- Post-quantum: complicated.
  Recent work analyzing cost: see `https://quantum.isogeny.org`.
  Several papers analyzing Kuperberg. (2018 Biasse–Iezzi-Jacobson, 2018-2020 Bonnetain–Schrottenloher, 2020 Peikert)
  `https://csidh.isogeny.org/analysis.html`

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?

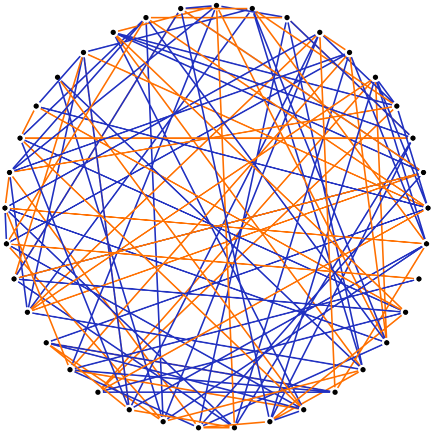
  (Is one qubit operation $\approx$ one bit operation? a hundred? millions?)

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

...but all known attacks cost $\exp\big((\log p)^{1/2+o(1)}\big)$!
Recent improvements to sieving target the $o(1)$.

Kuperberg applies to all commutative group actions.

SIDH – avoid commutativity



The supersingular isogeny graph over $\mathbb{F}_{p^2}$ looks differently.

Nodes are isomorphism classes of elliptic curves taken any extension field.
(All isooprhism classes of supersingular elliptic curves defined over $\mathbb{F}_{p^2}$).

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

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

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

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

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

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

SIDH: High-level view (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A\colon E \to E/A$; Bob computes $\varphi_B\colon E \to E/B$.
  (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 (2011 Jao–De Feo)

Promblem: quadratic twists are identified, $\ell + 1$ isogenies of degree ℓ from any curve, no more sense of direction.

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

- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A \colon E \to E/A$; Bob computes $\varphi_B \colon E \to E/B$.
  (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
  $$(E/B)/A' \cong E/\langle A, B\rangle \cong (E/A)/B'.$$
- Key is an isomorphism class; make this useable taking j-invariant.

SIDH's auxiliary points

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

Alice knows only A, Bob knows only φ_B.

SIDH's auxiliary points

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

Alice knows only A, Bob knows only φ_B.

- 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

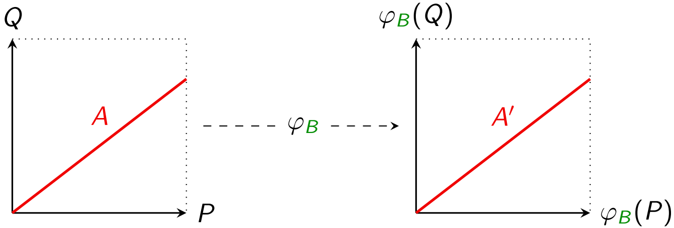
Previous slide: "Alice somehow obtains $A' := \varphi_B(A)$."

Alice knows only A, Bob knows only φ_B.

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

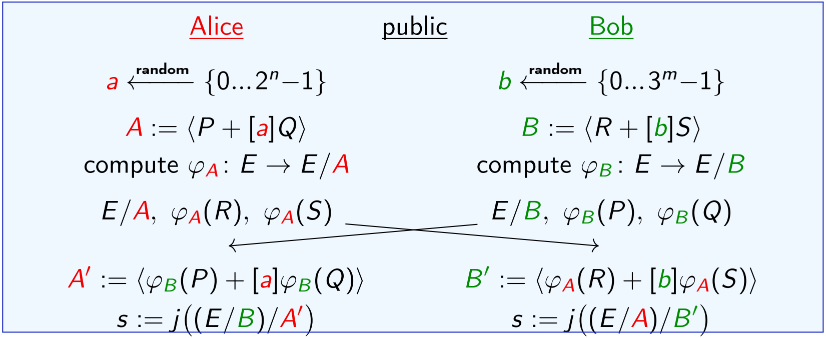


Using images of P and Q also lets Alice keep direction in iterative computation of φ_A.

SIDH in one slide

Public parameters:

- large prime $p = 2^n 3^m - 1$, supersingular $E/\mathbb{F}_{p^2}$ with $(p+1)^2$ points.
- bases (P, Q) and (R, S) of $E[2^n]$ and $E[3^m]$.
  Want these points defined over $\mathbb{F}_{p^2}$ for efficiency.
  Parameter chioce ensures this. 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 $\varphi_A \colon E \to E/A$ | compute $\varphi_B \colon E \to E/B$

$E/A,\ \varphi_A(R),\ \varphi_A(S)$ | $E/B,\ \varphi_B(P),\ \varphi_B(Q)$

$A' := \langle \varphi_B(P) + [a]\varphi_B(Q) \rangle$ | $B' := \langle \varphi_A(R) + [b]\varphi_A(S) \rangle$

$s := j\big((E/B)/A'\big)$ | $s := j\big((E/A)/B'\big)$

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

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

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

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

$\rightsquigarrow$ 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,
so their keys are in small corners of the key space.

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,
so their keys are in small corners of the key space.

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,
so their keys are in small corners of the key space.

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 $\tilde{\mathcal{O}}(p^{1/6})$. 2019 Jaques–Schank: $\tilde{\mathcal{O}}(p^{1/4})$:
  "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 $a = 2u$: $[a]Q'' = [a]Q' + [u][2^n]P' \qquad = [a]Q'$.
  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 ≈ as hard as breaking SIDH.

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

Comparison & open problems

Key bits where all known attacks take 2^λ operations (naive serial attack metric, ignoring memory cost):

	pre-quantum	post-quantum
SIDH, SIKE	$(24 + o(1))\lambda$	$(36 + o(1))\lambda$
compressed	$(14 + o(1))\lambda$	$(21 + o(1))\lambda$
CRS, CSIDH	$(4 + o(1))\lambda$	superlinear
ECDH	$(2 + o(1))\lambda$	exponential

- What CSIDH key sizes are needed for post-quantum security level 2^{64}? 2^{96}? 2^{128}?
- How is attack affected by occasional errors and non-uniform distributions over the group?

Comparison & open problems

Key bits where all known attacks take 2^λ operations (naive serial attack metric, ignoring memory cost):

	pre-quantum	post-quantum
SIDH, SIKE	$(24 + o(1))\lambda$	$(36 + o(1))\lambda$
compressed	$(14 + o(1))\lambda$	$(21 + o(1))\lambda$
CRS, CSIDH	$(4 + o(1))\lambda$	superlinear
ECDH	$(2 + o(1))\lambda$	exponential

- What CSIDH key sizes are needed for post-quantum security level 2^{64}? 2^{96}? 2^{128}?
- How is attack affected by occasional errors and non-uniform distributions over the group?
- How expensive is each CSIDH query? See our 2019 Eurocrypt paper—full 56-page version at `https://quantum.isogeny.org/` with detailed analysis and many optimizations.

Comparison & open problems

Key bits where all known attacks take 2^λ operations
(naive serial attack metric, ignoring memory cost):

	pre-quantum	post-quantum
SIDH, SIKE	$(24 + o(1))\lambda$	$(36 + o(1))\lambda$
compressed	$(14 + o(1))\lambda$	$(21 + o(1))\lambda$
CRS, CSIDH	$(4 + o(1))\lambda$	superlinear
ECDH	$(2 + o(1))\lambda$	exponential

- What CSIDH key sizes are needed for post-quantum security level 2^{64}? 2^{96}? 2^{128}?
- How is attack affected by occasional errors and non-uniform distributions over the group?
- How expensive is each CSIDH query? See our 2019 Eurocrypt paper—full 56-page version at `https://quantum.isogeny.org/` with detailed analysis and many optimizations.
- What about memory, using parallel AT metric?
- Find more attacks on SIDH. See "How to not break SIDH" `https://eprint.iacr.org/2019/558`.