SLMath Summer School
Isogeny-based cryptography
Day 2

Chloe Martindale

University of Bristol

Recall: Diffie–Hellman key exchange '76

Public parameters:

- a prime p (experts: uses $\mathbb{F}_p^*$, today also elliptic curves)
- a number $g \pmod{p}$ (nonexperts: think of an integer less than p)

Alice	Eve	Bob
$\mathrm{sk}_A \xleftarrow{\text{random}} \{0 \ldots p-1\}$		$\mathrm{sk}_B \xleftarrow{\text{random}} \{0 \ldots p-1\}$
$\mathrm{pk}_A = g^{\mathrm{sk}_A}$		$\mathrm{pk}_B = g^{\mathrm{sk}_B}$
$ss := (g^{\mathrm{sk}_B})^{\mathrm{sk}_A}$		$ss := (g^{\mathrm{sk}_A})^{\mathrm{sk}_B}$

- Alice and Bob agree on a shared secret key ss, then they can use that to encrypt their messages.
- Eve sees $\mathrm{pk}_A = g^{\mathrm{sk}_A}$, $\mathrm{pk}_B = g^{\mathrm{sk}_B}$; can't find sk_A, sk_B, ss.

Recall: Diffie–Hellman key exchange ’76

Public parameters:

- a prime p (experts: uses $\mathbb{F}_p^*$, today also elliptic curves)
- a number $g \pmod{p}$ (nonexperts: think of an integer less than p)

Alice | Eve

$\mathrm{sk}_A \xleftarrow{\text{random}} \{0 \ldots p-1\}$ | [illegible] $-1\}$

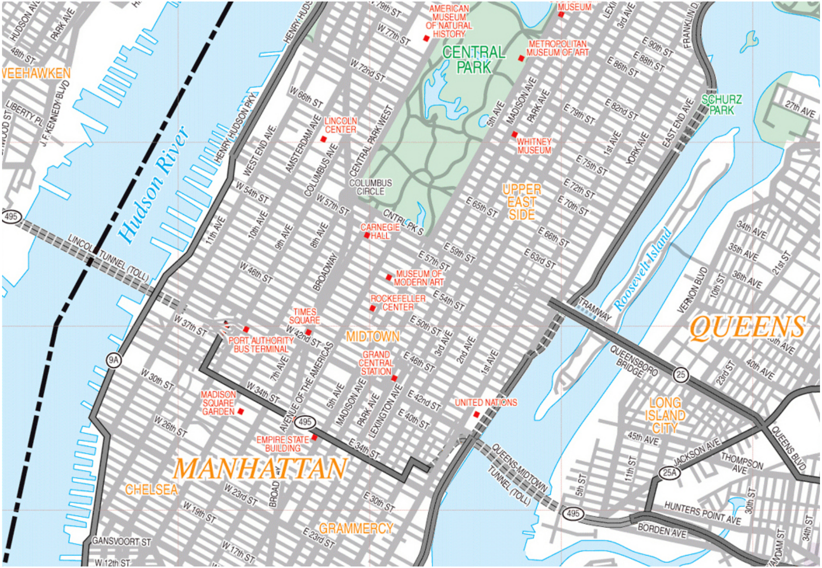
$\mathrm{pk}_A = g^{\mathrm{sk}_A}$ | [illegible] $_B = g^{\mathrm{sk}_B}$

[illegible] | $ss := (g^{\mathrm{sk}_A})^{\mathrm{sk}_B}$

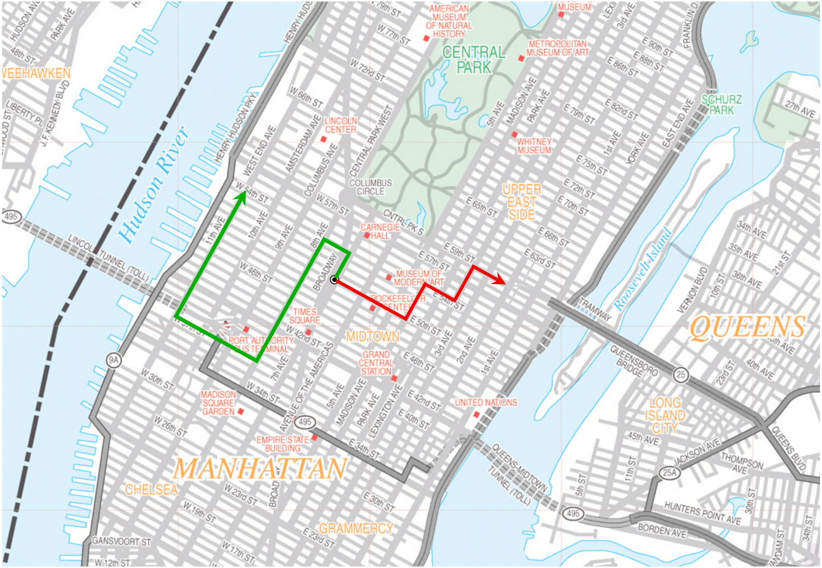
Broken by Shor!

- A[illegible]ob agree on a shared secret key ss, then they can [illegible]se that to encrypt their messages.
- Eve sees $\mathrm{pk}_A = g^{\mathrm{sk}_A}$, $\mathrm{pk}_B = g^{\mathrm{sk}_B}$; can’t find sk_A, sk_B, ss.

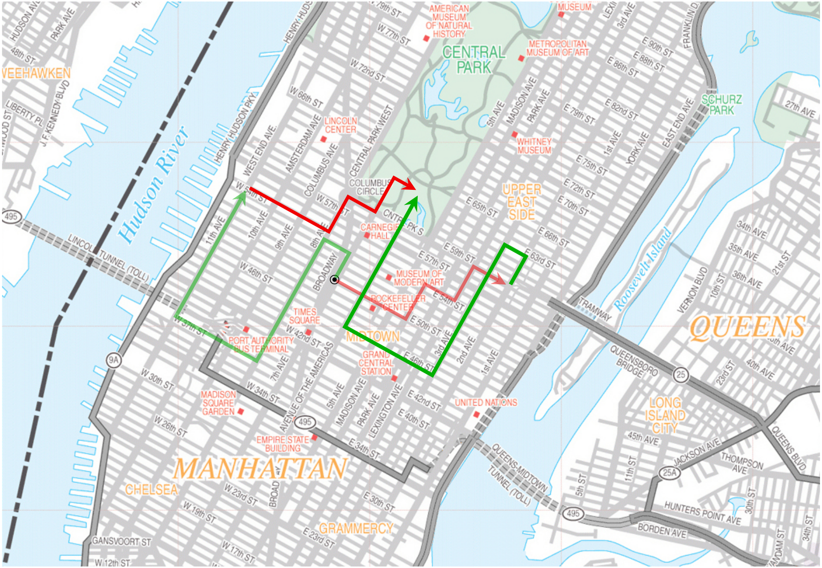
Graph walking Diffie–Hellman?



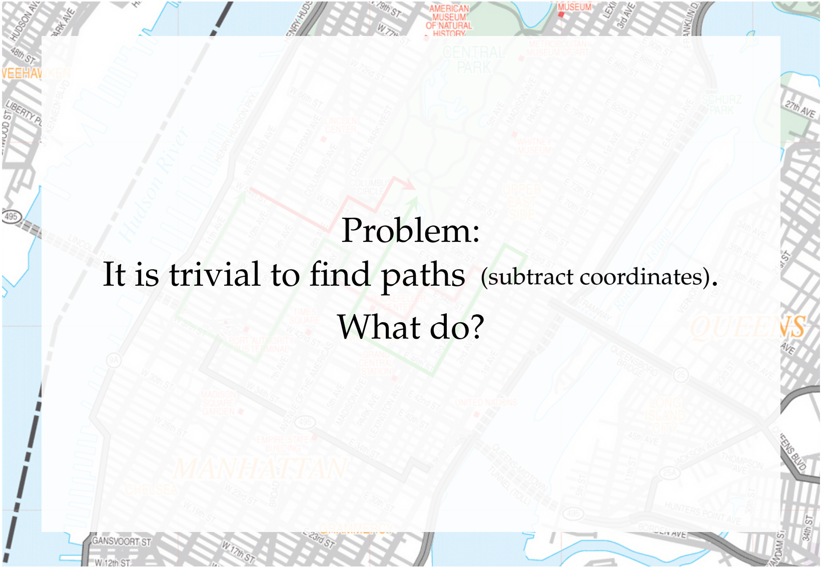
Graph walking Diffie–Hellman?



Graph walking Diffie–Hellman?



Graph walking Diffie–Hellman?



Problem:

It is trivial to find paths (subtract coordinates).

What do?

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 known efficient algorithms to recover paths from endpoints.

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 known efficient algorithms to recover paths from endpoints.
- 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 known efficient algorithms to recover paths from endpoints.
- 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!

Stand back!

We're going to do maths.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

An endomorphism of E is an isogeny $E \to E$, or the zero map.

The ring of endomorphisms of E is denoted by $\mathrm{End}(E)$.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

An endomorphism of E is an isogeny $E \to E$, or the zero map.
The ring of endomorphisms of E is denoted by $\mathrm{End}(E)$.

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

Maths background #2/3: 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 denoted by E/G. (cf. quotient groups)

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

[1] (up to isomorphism of E')

Maths background #2/3: 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 denoted by E/G. (cf. 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')

Maths background #2/3: 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 denoted by E/G. (cf. 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 supersingular curves!

[1] (up to isomorphism of E')

Math slide #3/3: Supersingular isogeny graphs

Let p be a prime, q a power of p, and ℓ a positive integer $\notin p\mathbb{Z}$.

An elliptic curve $E/\mathbb{F}_q$ is *supersingular* if $p \mid (q + 1 - \#E(\mathbb{F}_q))$.

We care about the cases $\#E(\mathbb{F}_p) = p + 1$ and $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$.

$\rightsquigarrow$ easy way to control the group structure by choosing p!

Math slide #3/3: Supersingular isogeny graphs

Let p be a prime, q a power of p, and ℓ a positive integer $\notin p\mathbb{Z}$.

An elliptic curve $E/\mathbb{F}_q$ is *supersingular* if $p \mid (q + 1 - \#E(\mathbb{F}_q))$.

We care about the cases $\#E(\mathbb{F}_p) = p + 1$ and $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$.

$\rightsquigarrow$ easy way to control the group structure by choosing p!

Let $S \not\ni p$ denote a set of prime numbers.

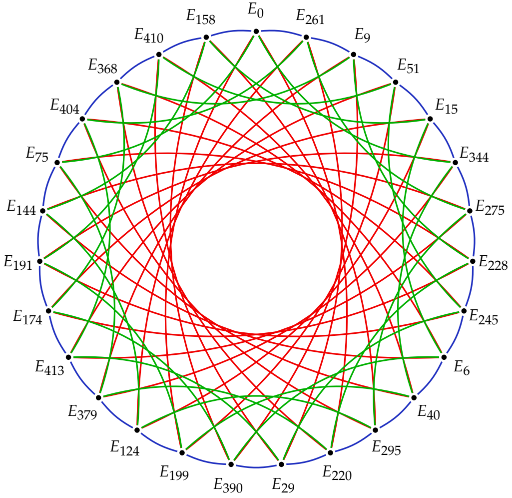
The supersingular S-isogeny graph over $\mathbb{F}_q$ consists of:

- vertices given by isomorphism classes of supersingular elliptic curves,
- edges given by equivalence classes[1] of ℓ-isogenies ($\ell \in S$),

both defined over $\mathbb{F}_q$.

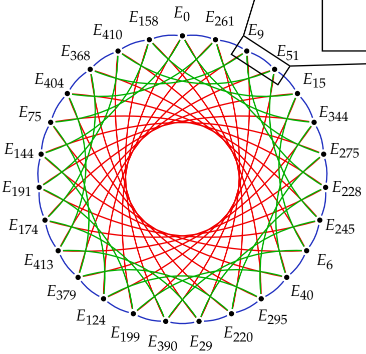
[1] Two isogenies $\varphi\colon E \to E'$ and $\psi\colon E \to E''$ are identified if $\psi = \iota \circ \varphi$ for some isomorphism $\iota\colon E' \to E''$.

Graphs of elliptic curves

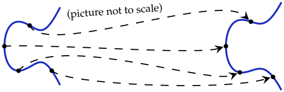


Nodes: Supersingular curves $E_A: y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
Edges: 3-, 5-, and 7-isogenies

Graphs of elliptic curves



A 3-isogeny



$$E_{51}\colon y^2=x^3+51x^2+x \longrightarrow E_9\colon y^2=x^3+9x^2+x$$

$$(x,y) \longmapsto \left(\frac{97x^3-183x^2+x}{x^2-183x+97},\; y\cdot\frac{133x^3+154x^2-5x+97}{-x^3+65x^2+128x-133}\right)$$

[ˈsiːˌsaɪd]

CRS or CSIDH

Traditionally, Diffie-Hellman works in a group G via the map

$$\begin{array}{rcl} \mathbb{Z} \times G & \rightarrow & G \\ (x, g) & \mapsto & g^x. \end{array}$$

CRS or CSIDH

Traditionally, Diffie-Hellman works in a group G via the map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x. \end{array}$$

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

CRS or CSIDH

Traditionally, Diffie-Hellman works in a group G via the map

$$\begin{array}{ccc} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x. \end{array}$$

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

$\rightsquigarrow$ Idea:

Replace exponentiation on the group G by a group action of a group H on a set S:

$$H \times S \to S.$$

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

by a group action on a set.

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.
- Replace $\mathbb{Z}$ by the commutative group $\mathrm{cl}(\mathbb{Z}\sqrt{-p}])$.

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.
- Replace $\mathbb{Z}$ by the commutative group $\mathrm{cl}(\mathbb{Z}\sqrt{-p}])$.
- An ideal in $\mathrm{cl}(\mathrm{End}_{\mathbb{F}_p}(E_A))$ is the kernel of an isogeny from E_A.

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{ccc} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.
- Replace $\mathbb{Z}$ by the commutative group $\mathrm{cl}(\mathbb{Z}\sqrt{-p}])$.
- An ideal in $\mathrm{cl}(\mathrm{End}_{\mathbb{F}_p}(E_A))$ is the kernel of an isogeny from E_A.
- The action of a well-chosen $\mathfrak{l} \in \mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ on S moves the elliptic curves one step around one of the cycles.

$$\begin{array}{ccc} \mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) \times S & \to & S \\ (\mathfrak{l}_3, E) & \mapsto & \mathfrak{l}_3 * E. \end{array}$$

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.
- Replace $\mathbb{Z}$ by the commutative group $\mathrm{cl}(\mathbb{Z}\sqrt{-p}])$.
- An ideal in $\mathrm{cl}(\mathrm{End}_{\mathbb{F}_p}(E_A))$ is the kernel of an isogeny from E_A.
- The action of a well-chosen $\mathfrak{l} \in \mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ on S moves the elliptic curves one step around one of the cycles.

$$\begin{array}{rcl} \mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) \times S & \to & S \\ (\mathfrak{l}_5, E) & \mapsto & \mathfrak{l}_5 * E. \end{array}$$

Quantumifying Exponentiation

- We want to replace the exponentiation map

$$\begin{array}{rcl} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

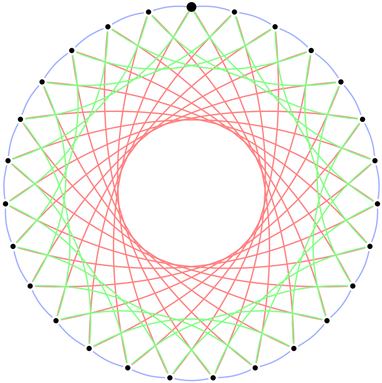
  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- For every $E_A \in S$, the ring of $\mathbb{F}_p$-rational endomorphisms $\mathrm{End}_{\mathbb{F}_p}(E_A)$ is isomorphic to $\mathbb{Z}[\sqrt{-p}]$.
- Replace $\mathbb{Z}$ by the commutative group $\mathrm{cl}(\mathbb{Z}\sqrt{-p}])$.
- An ideal in $\mathrm{cl}(\mathrm{End}_{\mathbb{F}_p}(E_A))$ is the kernel of an isogeny from E_A.
- The action of a well-chosen $\mathfrak{l} \in \mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ on S moves the elliptic curves one step around one of the cycles.

$$\begin{array}{rcl} \mathrm{cl}(\mathbb{Z}[\sqrt{-p}]) \times S & \to & S \\ (\mathfrak{l}_7, E) & \mapsto & \mathfrak{l}_7 * E. \end{array}$$

Diffie and Hellman go to the CSIDH

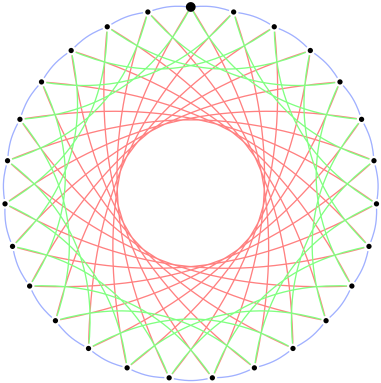
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$



Bob

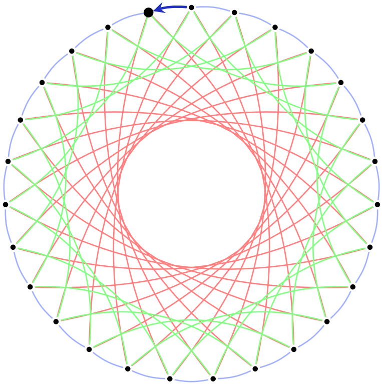
$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$



Diffie and Hellman go to the CSIDH

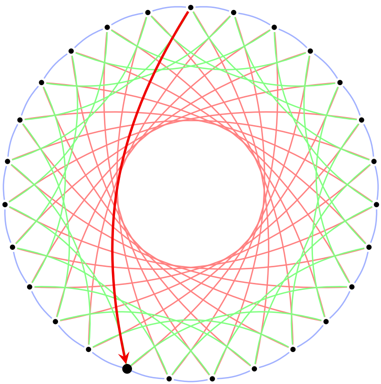
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$
$\uparrow$



Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$
$\uparrow$

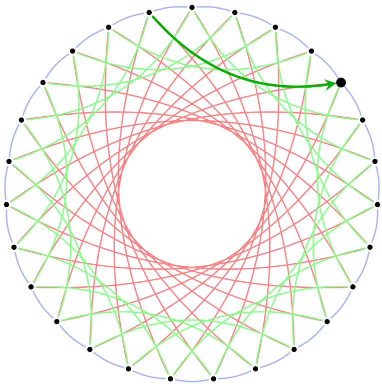


Diffie and Hellman go to the CSIDH

Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$

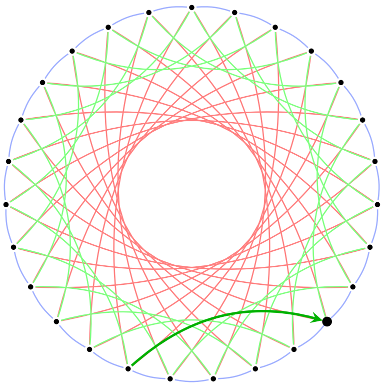
$\uparrow$



Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$

$\uparrow$



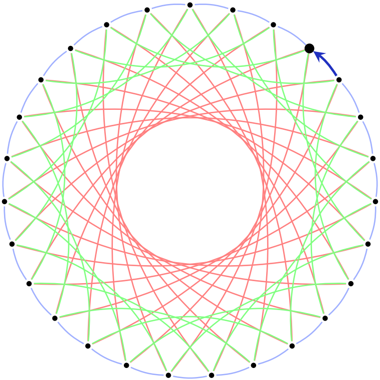
Diffie and Hellman go to the CSIDH

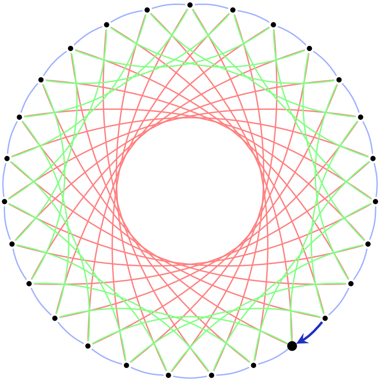
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$
$\uparrow$

Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$
$\uparrow$





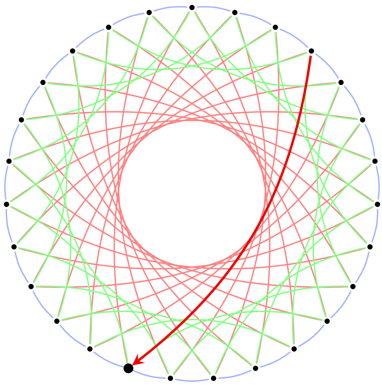
Diffie and Hellman go to the CSIDH

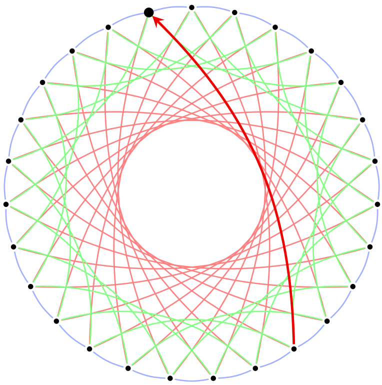
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \underset{\uparrow}{\mathfrak{l}_5{}^{-1}}]$

Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \underset{\uparrow}{\mathfrak{l}_5}]$

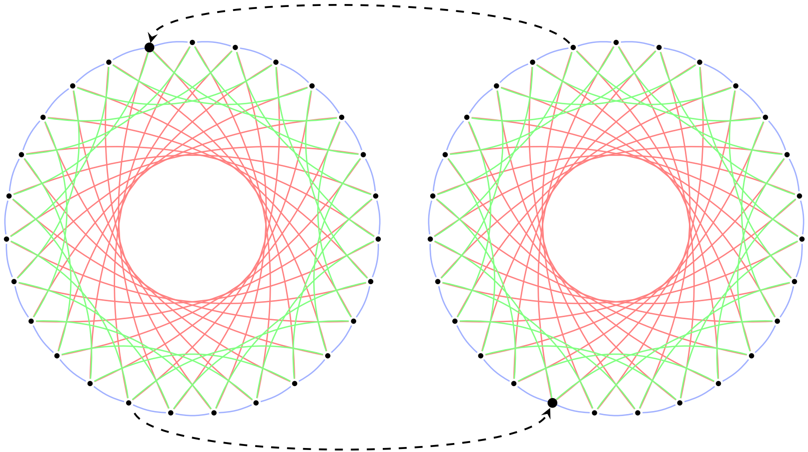




Diffie and Hellman go to the CSIDH

Alice
$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$

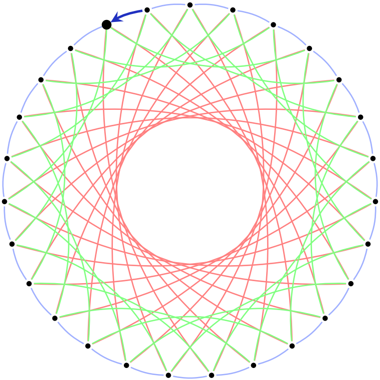
Bob
$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$



Diffie and Hellman go to the CSIDH

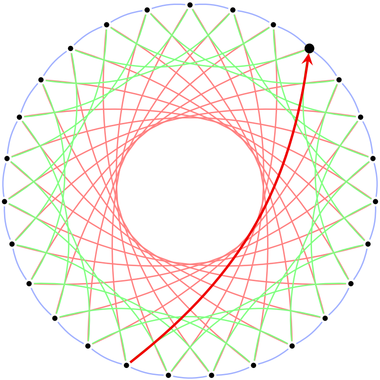
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7^{-1}, \mathfrak{l}_3, \mathfrak{l}_5^{-1}]$
$\uparrow$



Bob

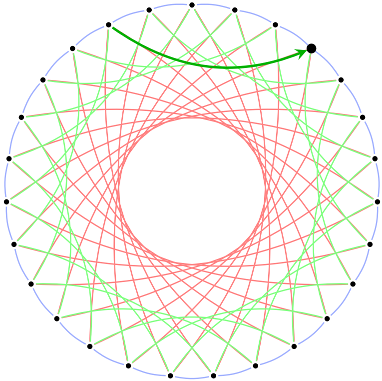
$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3^{-1}, \mathfrak{l}_5]$
$\uparrow$



Diffie and Hellman go to the CSIDH

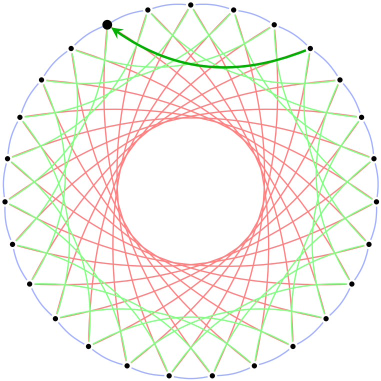
Alice

$[\mathfrak{l}_3, \underset{\uparrow}{\mathfrak{l}_7}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$



Bob

$[\mathfrak{l}_5, \underset{\uparrow}{\mathfrak{l}_7}, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$

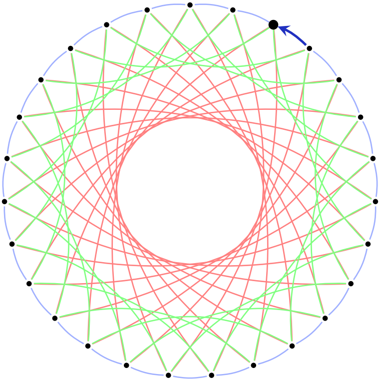


Diffie and Hellman go to the CSIDH

Alice

$[\mathfrak{l}_3, \mathfrak{l}_7^{-1}, \mathfrak{l}_3, \mathfrak{l}_5^{-1}]$

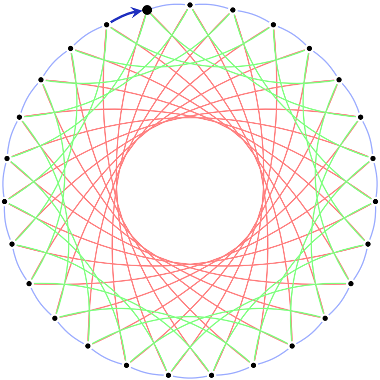
$\uparrow$



Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3^{-1}, \mathfrak{l}_5]$

$\uparrow$

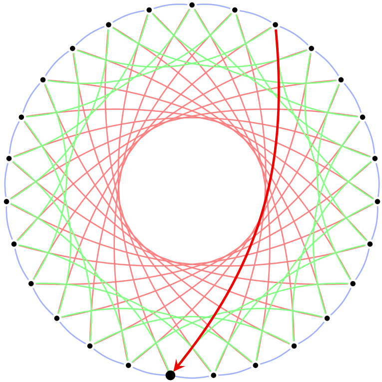


Diffie and Hellman go to the CSIDH

Alice

$[\mathfrak{l}_3, \mathfrak{l}_7^{-1}, \mathfrak{l}_3, \mathfrak{l}_5^{-1}]$

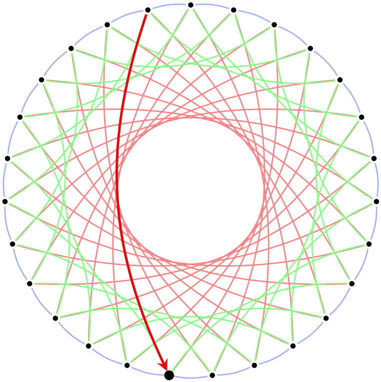
$\uparrow$



Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3^{-1}, \mathfrak{l}_5]$

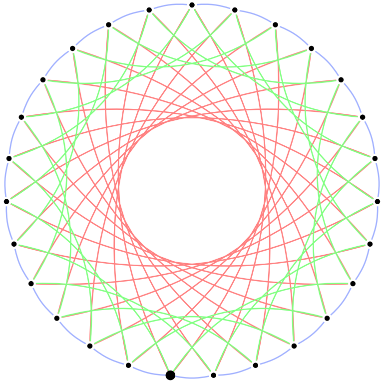
$\uparrow$



Diffie and Hellman go to the CSIDH

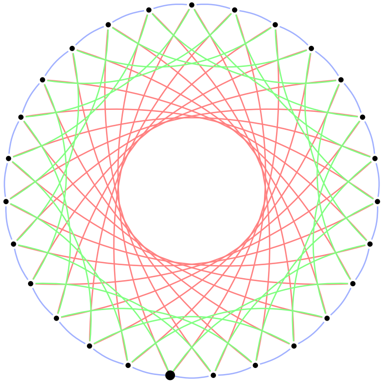
Alice

$[\mathfrak{l}_3, \mathfrak{l}_7{}^{-1}, \mathfrak{l}_3, \mathfrak{l}_5{}^{-1}]$



Bob

$[\mathfrak{l}_5, \mathfrak{l}_7, \mathfrak{l}_3{}^{-1}, \mathfrak{l}_5]$



Choosing parameters

In [CLMPR18], parameters are chosen as follows:

- $\ell_1, \ldots, \ell_{n-1}$ the first $n-1$ odd primes.
- $\ell_n > \ell_{n-1}$ the smallest prime such that $p = 4\ell_1 \cdots \ell_n - 1$ is prime.

Then:

- $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ correspond to kernels of $\mathbb{F}_p$-rational isogenies (see next slide) — fast.
- Allowing up to 5 actions of each $\mathfrak{l}_i^{(-1)}$ covers* the whole class group — security then depends on size of class group.

Choosing parameters

In [CLMPR18], parameters are chosen as follows:

- $\ell_1, \ldots, \ell_{n-1}$ the first $n-1$ odd primes.
- $\ell_n > \ell_{n-1}$ the smallest prime such that $p = 4\ell_1 \cdots \ell_n - 1$ is prime.

Then:

- $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ correspond to kernels of $\mathbb{F}_p$-rational isogenies (see next slide) — fast.
- Allowing up to 5 actions of each $\mathfrak{l}_i^{(-1)}$ covers* the whole class group — security then depends on size of class group.

*Any $I \in \mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ can be written as $\prod \mathfrak{l}_i^{e_i}$ with $e_i \in [-5, \ldots, 5]$.

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p+1$ or $(p+1)/2$.

- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p+1$ or $(p+1)/2$.
  - For every odd prime $\ell | (p+1)$, the point $\frac{p+1}{\ell} P$ is a point of order ℓ.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p+1$ or $(p+1)/2$.
  - For every odd prime $\ell | (p+1)$, the point $\frac{p+1}{\ell}P$ is a point of order ℓ.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).
  - Given a $\mathbb{F}_p$-rational point of order ℓ, the isogeny computations can be done over $\mathbb{F}_p$.

Representing nodes of the graph

- Every node of G_{ℓ_i} is

$$E_A : y^2 = x^3 + Ax^2 + x.$$

Representing nodes of the graph

- Every node of G_{ℓ_i} is

$$E_A : y^2 = x^3 + Ax^2 + x.$$

$\Rightarrow$ Can compress every node to a single value $A \in \mathbb{F}_p$.

Representing nodes of the graph

- Every node of G_{ℓ_i} is

$$E_A : y^2 = x^3 + Ax^2 + x.$$

$\Rightarrow$ Can compress every node to a single value $A \in \mathbb{F}_p$.

$\Rightarrow$ Tiny keys!

Does any A work?

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p+1$ points.

Does any A work?

No.

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Does any A work?

No.

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Does any A work?

No.

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
- Public-key validation: Check that E_A has $p + 1$ points.

  Easy Monte-Carlo algorithm: Pick random P on E_A and check $[p+1]P = \infty$.[1]

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

- Kuperberg's algorithm [Kup1] requires a subexponential number of queries, and a subexponential number of operations on a subexponential number of qubits.

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

- Kuperberg's algorithm [Kup1] requires a subexponential number of queries, and a subexponential number of operations on a subexponential number of qubits.
- Variant by Regev [Reg] uses polynomial number of qubits at the expense of time.

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

- Kuperberg's algorithm [Kup1] requires a subexponential number of queries, and a subexponential number of operations on a subexponential number of qubits.
- Variant by Regev [Reg] uses polynomial number of qubits at the expense of time.
- Kuperberg later [Kup2] gave more trade-off options for quantum and classical memory vs. time.

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

- Kuperberg's algorithm [Kup1] requires a subexponential number of queries, and a subexponential number of operations on a subexponential number of qubits.
- Variant by Regev [Reg] uses polynomial number of qubits at the expense of time.
- Kuperberg later [Kup2] gave more trade-off options for quantum and classical memory vs. time.
- Childs-Jao-Soukharev [CJS] applied Kuperberg/Regev to CRS – their attack also applies to CSIDH.

Quantum Security

Hidden-shift algorithms: Subexponential complexity (Kuperberg, Regev).

- Kuperberg's algorithm [Kup1] requires a subexponential number of queries, and a subexponential number of operations on a subexponential number of qubits.
- Variant by Regev [Reg] uses polynomial number of qubits at the expense of time.
- Kuperberg later [Kup2] gave more trade-off options for quantum and classical memory vs. time.
- Childs-Jao-Soukharev [CJS] applied Kuperberg/Regev to CRS – their attack also applies to CSIDH.
- Part of CJS attack computes many paths in superposition.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.
- For fastest variant of Kuperberg, total cost of CSIDH-512 attack is at least 2^{56} qubit operations.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.
- For fastest variant of Kuperberg, total cost of CSIDH-512 attack is at least 2^{56} qubit operations.
- Overheads from error correction, high quantum memory etc., not yet understood.

Better parameters - SQALE

[CCJR22] propose the SQALE of CSIDH.

- Uses huge $p = 4\ell_1 \cdots \ell_n - 1$

Better parameters - SQALE

[CCJR22] propose the SQALE of CSIDH.

- Uses huge $p = 4\ell_1 \cdots \ell_n - 1$
- Uses only $\mathfrak{l}_i^{\pm 1}$

Better parameters - SQALE

[CCJR22] propose the SQALE of CSIDH.

- Uses huge $p = 4\ell_1 \cdots \ell_n - 1$
- Uses only $\mathfrak{l}_i^{\pm 1}$
- Tiny fraction of class group used

Better parameters - SQALE

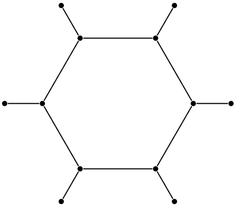
[CCJR22] propose the SQALE of CSIDH.

- Uses huge $p = 4\ell_1 \cdots \ell_n - 1$
- Uses only $\mathfrak{l}_i^{\pm 1}$
- Tiny fraction of class group used
- Not a subgroup $\rightsquigarrow$ Kuperberg has to use huge group

Better parameters - CSURF

Q: What about 2-isogenies?

- The 2-isogeny graph looks like this:

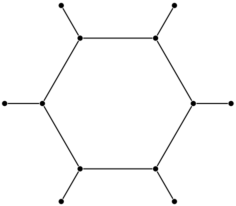


- This is called an isogeny volcano.

Better parameters - CSURF

Q: What about 2-isogenies?

- The 2-isogeny graph looks like this:

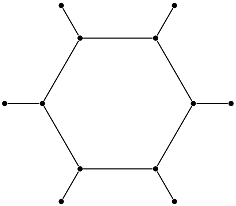


- This is called an isogeny volcano.
- Edges on the cycle are horizontal.

Better parameters - CSURF

Q: What about 2-isogenies?

- The 2-isogeny graph looks like this:

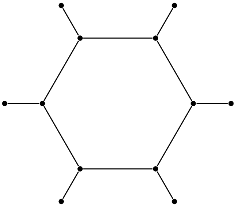


- This is called an isogeny volcano.
- Edges on the cycle are horizontal.
- Away / back to the cycle is descending / ascending.

Better parameters - CSURF

Q: What about 2-isogenies?

- The 2-isogeny graph looks like this:



- This is called an isogeny volcano.
- Edges on the cycle are horizontal.
- Away / back to the cycle is descending / ascending.

$\rightsquigarrow$ How to compute 'on the surface'?

Better parameters - CSURF

[CD19] solve these problems:

- Set $p = 4f\ell_1 \cdots \ell_n - 1$ where $\ell_1 = 2$.

Better parameters - CSURF

[CD19] solve these problems:

- Set $p = 4f\ell_1 \cdots \ell_n - 1$ where $\ell_1 = 2$.
- Set $E_0/\mathbb{F}_p : y^2 = x^3 - x$. Then E_0 is 'on the surface'.

Better parameters - CSURF

[CD19] solve these problems:

- Set $p = 4f\ell_1 \cdots \ell_n - 1$ where $\ell_1 = 2$.
- Set $E_0/\mathbb{F}_p : y^2 = x^3 - x$. Then E_0 is 'on the surface'.
- For any curve on the surface, the 2-isogeny with kernel $\langle(0, 0)\rangle$ is horizontal.

Venturing further beyond the CSIDH

A selection of more advances since original publication (2018):

- sqrtVelu [BDLS20]: square-root speed-up on computation of large-degree isogenies.
- Radical isogenies [CDV20]: significant speed-up on isogenies of small-ish degree.
- Some work on different curve forms (e.g. Edwards).
- Knowledge of $\mathrm{End}(E_0)$ and $\mathrm{End}(E_A)$ breaks CSIDH in classical polynomial time [Wes21].
- CTIDH [$\mathrm{B}^2\mathrm{C}^2\mathrm{LMS}^2$]: Efficient constant-time CSIDH-style construction.

What about signatures? (S '06, DG '18, BKV '19, DFKLMPW '23)

Identification protocol:

- Alice generates $(\mathsf{sk}_A, \mathsf{pk}_A)$, publishes pk_A.
- Alice proves to Bob that she knows sk_A.
- Bob verifies Alice's proof.

What about signatures? (S '06, DG '18, BKV '19, DFKLMPW '23)

Identification protocol:

- Alice generates $(\mathsf{sk}_A, \mathsf{pk}_A)$, publishes pk_A.
- Alice proves to Bob that she knows sk_A.
- Bob verifies Alice's proof.

Typically:

1. Prover: generates ephemeral $(\mathsf{esk}, \mathsf{epk})$, publishes epk.

What about signatures? (S '06, DG '18, BKV '19, DFKLMPW '23)

Identification protocol:

- Alice generates $(\mathsf{sk}_A, \mathsf{pk}_A)$, publishes pk_A.
- Alice proves to Bob that she knows sk_A.
- Bob verifies Alice's proof.

Typically:

1. Prover: generates ephemeral $(\mathsf{esk}, \mathsf{epk})$, publishes epk.
2. Verifier: sends Prover a challenge c.

What about signatures? (S '06, DG '18, BKV '19, DFKLMPW '23)

Identification protocol:

- Alice generates $(\mathsf{sk}_A, \mathsf{pk}_A)$, publishes pk_A.
- Alice proves to Bob that she knows sk_A.
- Bob verifies Alice's proof.

Typically:

1. Prover: generates ephemeral $(\mathsf{esk}, \mathsf{epk})$, publishes epk.
2. Verifier: sends Prover a challenge c.
3. Prover: c, esk, $\mathsf{sk} \rightsquigarrow$ proof-of-knowledge P.

What about signatures? (S '06, DG '18, BKV '19, DFKLMPW '23)

Identification protocol:

- Alice generates $(\mathsf{sk}_A, \mathsf{pk}_A)$, publishes pk_A.
- Alice proves to Bob that she knows sk_A.
- Bob verifies Alice's proof.

Typically:

1. Prover: generates ephemeral $(\mathsf{esk}, \mathsf{epk})$, publishes epk.
2. Verifier: sends Prover a challenge c.
3. Prover: c, esk, $\mathsf{sk} \rightsquigarrow$ proof-of-knowledge P.
4. Verifier: P, pk, $\mathsf{epk} \rightsquigarrow$ valid (or not!)

Identification scheme from $H \times S \to S$

Prover	**Public**	**Verifier**
	$E \in S, \mathfrak{l}_i \in H$	
$s_i \xleftarrow{\$} \mathbb{Z}$		
$\mathsf{sk} = \prod \mathfrak{l}_i^{s_i},$		
$\mathsf{pk} = \mathsf{sk} * E \xrightarrow{\mathsf{pk}}$	pk	
$t_i \xleftarrow{\$} \mathbb{Z}$		
$\mathsf{esk} = \prod \mathfrak{l}_i^{t_i},$		
$\mathsf{epk}_1 = \mathsf{esk} * E, \xrightarrow{\mathsf{epk}_1}$		$c \xleftarrow{\$} \{0, 1\}$
$\mathsf{epk}_2 = \mathsf{esk} \cdot \mathsf{sk}^{-c} \xleftarrow{c}$		
$\xrightarrow{\mathsf{pk}, \mathsf{epk}_1, \mathsf{epk}_2}$		check:
		$\mathsf{epk}_1 = \mathsf{epk}_2 * ([\mathsf{sk}^c] * E).$

After k challenges c, an imposter succeeds with prob 2^{-k}.

From SeaSign to SCALLOP

- [S06] proposed for CRS

From SeaSign to SCALLOP

- [S06] proposed for CRS
- [DG18] proposed SeaSign for CSIDH

From SeaSign to SCALLOP

- [S06] proposed for CRS
- [DG18] proposed SeaSign for CSIDH
- Downfall: class group structure needed for classical efficiency

From SeaSign to SCALLOP

- [S06] proposed for CRS
- [DG18] proposed SeaSign for CSIDH
- Downfall: class group structure needed for classical efficiency
- [BKV19] proposed CSI-FiSh: computed class group for smallest parameters

From SeaSign to SCALLOP

- [S06] proposed for CRS
- [DG18] proposed SeaSign for CSIDH
- Downfall: class group structure needed for classical efficiency
- [BKV19] proposed CSI-FiSh: computed class group for smallest parameters
- [DFKLMPW23] proposed SCALLOP: constructs class group with large parameters (c.f. SQALE)

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find $\mathfrak{a} \in H$ such that
$\mathfrak{a} * E = E'$.

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign is a signature scheme based on this idea:

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign is a signature scheme based on this idea:

$$\begin{tikzcd} E \arrow[d] \\ E_{\mathrm{pk}} \end{tikzcd}$$

public, secret, ephemeral secret, public challenge, public proof

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign is a signature scheme based on this idea:

$$\begin{tikzcd} E \arrow[r] \arrow[d] & E_{\mathrm{epk}} \\ E_{\mathrm{pk}} & \end{tikzcd}$$

public, secret, ephemeral secret, public challenge, public proof

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign is a signature scheme based on this idea:

$$\begin{tikzcd} E \arrow[r] \arrow[d] & E_{\mathrm{epk}} \arrow[d] \\ E_{\mathrm{pk}} & E_{\mathrm{ver}} \end{tikzcd}$$

public, secret, ephemeral secret, public challenge, public proof

SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny $E \to E'$

SQISign is a signature scheme based on this idea:

$$\begin{tikzcd} E \arrow[r] \arrow[d] & E_{\mathrm{epk}} \arrow[d] \\ E_{\mathrm{pk}} \arrow[r] & E_{\mathrm{ver}} \end{tikzcd}$$

public, secret, ephemeral secret, public challenge, public proof

Evolution of key exchange

$$
\begin{tikzcd}
 & E_A = \alpha * E_0 \arrow[r, "\beta * (-)"] \arrow[d, dashed, shift right] & \beta * E_A \arrow[dd, phantom, "\wr\|"] \\
E_0 \arrow[ur, "\alpha * (-)"] \arrow[dr, "\beta * (-)"'] & \boxed{E_A, E_B} \arrow[u, dashed, shift right] \arrow[d, dashed, shift right] & \\
 & E_B = \beta * E_0 \arrow[r, "\alpha * (-)"'] \arrow[u, dashed, shift right] & \alpha * E_B
\end{tikzcd}
$$

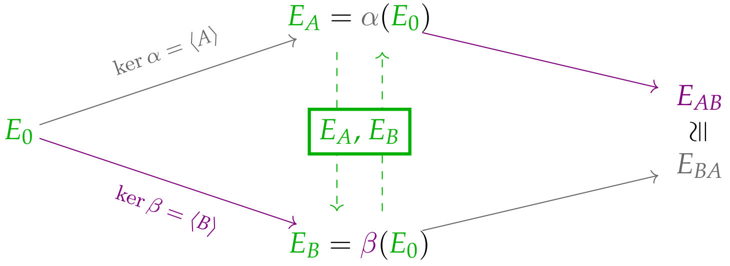
Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange

$$
\begin{tikzcd}
 & E_A = \alpha(E_0) \arrow[rd, "\beta(-)"] \arrow[dd, dashed, shift right=3] & \\
E_0 \arrow[ru, "\alpha(-)"] \arrow[rd, "\beta(-)"'] & \boxed{E_A, E_B} & E_{AB} \arrow[d, phantom, "\wr\|"] \\
 & E_B = \beta(E_0) \arrow[r, "\alpha(-)"'] \arrow[uu, dashed, shift right=3] & E_{BA}
\end{tikzcd}
$$

Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange



Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange

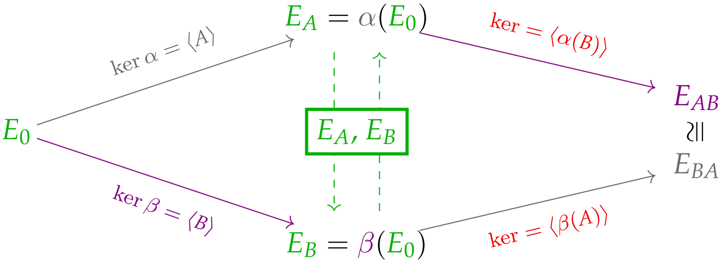
CRS or CSIDH

$$\begin{tikzcd}
 & E_A = \alpha(E_0) \arrow[dd, dashed, shift right] \arrow[dr, "\ker = \langle\alpha(B)\rangle"] & \\
 & & E_{AB} \\
E_0 \arrow[uur, "\ker\alpha = \langle A\rangle"] \arrow[ddr, "\ker\beta = \langle B\rangle"'] & \boxed{E_A, E_B} \arrow[dd, dashed, shift right] \arrow[uu, dashed, shift left] & \wr\| \\
 & & E_{BA} \\
 & E_B = \beta(E_0) \arrow[uu, dashed, shift left] \arrow[ur, "\ker = \langle\beta(A)\rangle"'] &
\end{tikzcd}$$

Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange

From CRS to SIDH



Colour code: Public, Alice's secret, Bob's secret, ?!

Evolution of key exchange

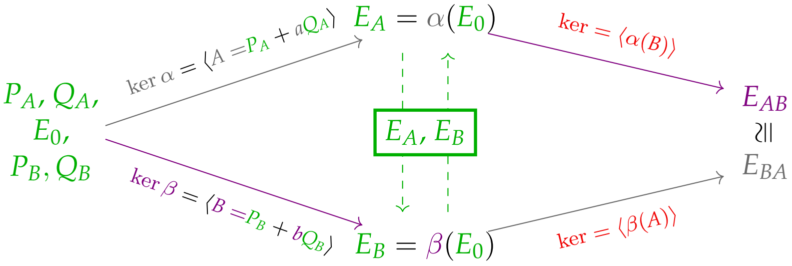
From CRS to SIDH

$$
\begin{tikzcd}
 & E_A = \alpha(E_0) \arrow[dr, "\ker = \langle \alpha(B) \rangle"] \arrow[dd, dashed, shift right] & \\
E_0 \arrow[ur, "\ker \alpha = \langle A \rangle"] \arrow[dr, "\ker \beta = \langle B \rangle"'] & \boxed{E_A, E_B} & E_{AB} \cong E_{BA} \\
 & E_B = \beta(E_0) \arrow[ur, "\ker = \langle \beta(A) \rangle"'] \arrow[uu, dashed, shift right] &
\end{tikzcd}
$$

Colour code: Public, Alice's secret, Bob's secret, ?!

Evolution of key exchange

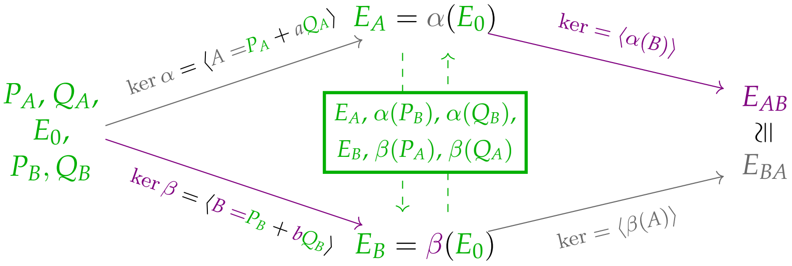
From CRS to SIDH



Colour code: Public, Alice's secret, Bob's secret, ?!

Evolution of key exchange

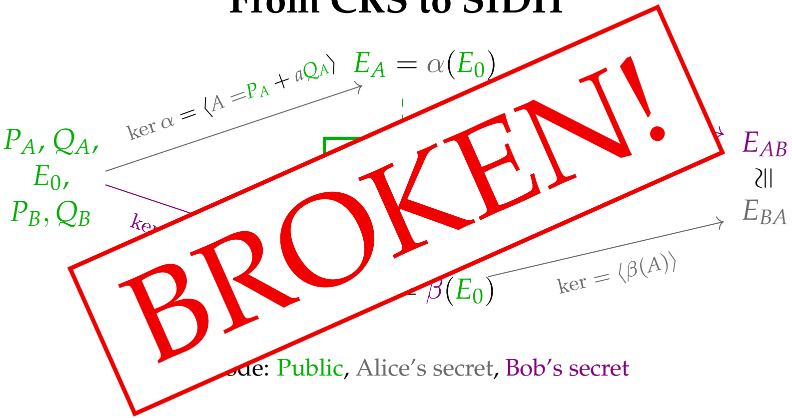
From CRS to SIDH



Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange

From CRS to SIDH



Evolution of key exchange

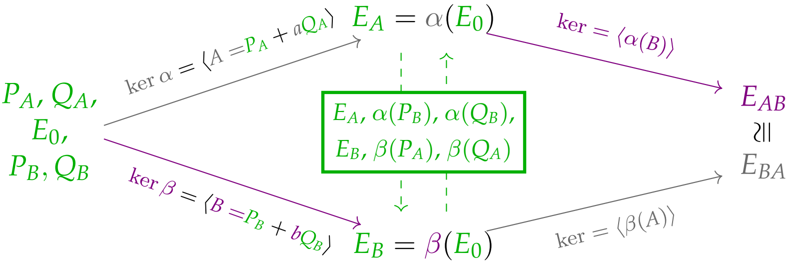
From CRS to SIDH

$$
\begin{tikzcd}
 & E_A = \alpha(E_0) \arrow[dr, "\ker = \langle \alpha(B) \rangle"] & \\
\begin{matrix} P_A, Q_A, \\ E_0, \\ P_B, Q_B \end{matrix} \arrow[ur, "\ker \alpha = \langle A = P_A + aQ_A \rangle"] \arrow[ddr, "\ker \beta = \langle B = P_B + bQ_B \rangle"'] & \boxed{\begin{matrix} E_A, \alpha(P_B), \alpha(Q_B), \\ E_B, \beta(P_A), \beta(Q_A) \end{matrix}} \arrow[u, dashed] \arrow[dd, dashed] & E_{AB} \arrow[d, phantom, "\wr\|"] \\
 & & E_{BA} \\
 & E_B = \beta(E_0) \arrow[ur, "\ker = \langle \beta(A) \rangle"'] &
\end{tikzcd}
$$

Colour code: Public, Alice's secret, Bob's secret

Evolution of key exchange

SIDH



Colour code: Public, Alice's secret, Bob's secret

Summary of hard problems

- CRS / CSIDH – Finding α given E and $\alpha * E$.

Summary of hard problems

- CRS / CSIDH – Finding α given E and $\alpha * E$.
- All isogeny-based schemes – Given elliptic curves E_0 and E_A, compute an isogeny $\alpha : E_0 \to E_A$ if it exists.

Summary of hard problems

- CRS / CSIDH – Finding α given E and $\alpha * E$.
- All isogeny-based schemes – Given elliptic curves E_0 and E_A, compute an isogeny $\alpha : E_0 \to E_A$ if it exists.
- All isogeny-based schemes – Given a random supersingular elliptic curve E, compute $\mathrm{End}(E)$.

Summary of hard problems

- CRS / CSIDH – Finding α given E and $\alpha * E$.
- All isogeny-based schemes – Given elliptic curves E_0 and E_A, compute an isogeny $\alpha : E_0 \to E_A$ if it exists.
- All isogeny-based schemes – Given a random supersingular elliptic curve E, compute $\text{End}(E)$.
- SIDH –

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

*Details for the elliptic curve lovers:

p a large prime; $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ supersingular; $\deg(\alpha)$, B public large smooth coprime integers; points P_B, Q_B chosen such that $\langle P_B, Q_B \rangle = E_0[B]$.

Summary of hard problems

- CRS / CSIDH – Finding α given E and $\alpha * E$.
- All isogeny-based schemes – Given elliptic curves E_0 and E_A, compute an isogeny $\alpha : E_0 \to E_A$ if it exists.
- All isogeny-based schemes – Given a random supersingular elliptic curve E, compute $\mathrm{End}(E)$.
- SIDH –

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given $\alpha(E_0[B])$, compute α. (modulo technical restrictions)*

*Details for the elliptic curve lovers:

p a large prime; $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ supersingular; $\deg(\alpha)$, B public large smooth coprime integers; points P_B, Q_B chosen such that $\langle P_B, Q_B \rangle = E_0[B]$.

History of the SIDH problem

2011 Problem introduced by De Feo, Jao, and Plut

2016 Galbraith, Petit, Shani, Ti give active attack

2017 Petit gives passive attack on some parameter sets

2020 de Quehen, Kutas, Leonardi, M., Panny, Petit, Stange give passive attack on more parameter sets

2022 Castryck-Decru and Maino-M. give passive attack on SIKE parameter sets; Robert extends to all parameter sets

- CD and MM attack is subexponential in most cases
- CD attack polynomial-time when $\mathrm{End}(E_0)$ known
- Robert attack polynomial-time in all cases
- Panny and Pope implement MM attack; Wesolowski independently discovers direct recovery method

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd} E_0 \arrow[rr, bend left, "\alpha"] & & E_A \arrow[ll, bend left, "\widehat{\alpha}"] \end{tikzcd}$$

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd} E_0 \arrow[loop left, "\iota"] \arrow[r, bend left, "\alpha"] & E_A \arrow[l, bend left, "\widehat{\alpha}"] \end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd}
E_0 \arrow[loop left, "\iota"] \arrow[rr, bend left, "\alpha"] & & E_A \arrow[ll, bend left, "\widehat{\alpha}"] \arrow[loop right, "\theta = \alpha \circ \iota \circ \widehat{\alpha}\ (+[n])"]
\end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd}
E_0 \arrow[loop left, "\iota"] \arrow[r, bend left, "\alpha"] & E_A \arrow[l, bend left, "\widehat{\alpha}"] \arrow[loop right, "\theta = \alpha \circ \iota \circ \widehat{\alpha}\ (+[n])"]
\end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.
- Know $\alpha(E_0[B])$ (and $\widehat{\alpha}(E_A[B])$) from public torsion points.

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd} E_0 \arrow[loop left, "\iota"] \arrow[rr, bend left, "\alpha"] & & E_A \arrow[ll, bend left, "\widehat{\alpha}"] \arrow[loop right, "\theta = \alpha \circ \iota \circ \widehat{\alpha}\ (+[n])"] \end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.
- Know $\alpha(E_0[B])$ (and $\widehat{\alpha}(E_A[B])$) from public torsion points.
- Know $\deg(\theta) = \deg(\alpha)^2 \deg(\iota) + n^2$.

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd}
E_0 \arrow[loop left, "\iota"] \arrow[r, bend left, "\alpha"] & E_A \arrow[l, bend left, "\widehat{\alpha}"] \arrow[loop right, "\theta = \alpha \circ \iota \circ \widehat{\alpha}\ (+[n])"]
\end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.
- Know $\alpha(E_0[B])$ (and $\widehat{\alpha}(E_A[B])$) from public torsion points.
- Know $\deg(\theta) = \deg(\alpha)^2 \deg(\iota) + n^2$.
- Restriction # 2: If there exist ι, n such that $\deg(\theta) = B$, then can completely determine θ, and α, in polynomial-time.

Petit's trick: torsion points to isogenies

Finding the secret isogeny α of known degree, given $\alpha(E_0[B])$.

$$\begin{tikzcd} E_0 \arrow[loop left, "\iota"] \arrow[r, bend left, "\alpha"] & E_A \arrow[l, bend left, "\widehat{\alpha}"] \arrow[loop right, "\theta = \alpha \circ \iota \circ \widehat{\alpha}\ (+[n])"] \end{tikzcd}$$

- Restriction # 1: Assume we can choose $\iota : E_0 \to E_0$.
- Know $\alpha(E_0[B])$ (and $\widehat{\alpha}(E_A[B])$) from public torsion points.
- Know $\deg(\theta) = \deg(\alpha)^2 \deg(\iota) + n^2$.
- Restriction # 2: If there exist ι, n such that $\deg(\theta) = B$, then can completely determine θ, and α, in polynomial-time.
- Restriction # 2 rules out SIKE parameters, where $B \approx \deg(\alpha)$ (and $p \approx B \cdot \deg \alpha$).

Enter Kani

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

Enter Kani

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

Problem:
Not enough choices $\theta : E_A \to E_A$.
'No θ of degree N.'

Enter Kani

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

Problem:
Not enough choices $\theta : E_A \to E_A$.
'No θ of degree N.'

Solution? $\theta : E_0 \times E_A \to E_0 \times E_A$?
$\rightsquigarrow$ still not enough.

Enter Kani

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

Problem:
Not enough choices $\theta : E_A \to E_A$.
'No θ of degree N.'

Solution? $\theta : E_0 \times E_A \to E_0 \times E_A$?
$\rightsquigarrow$ still not enough. But!

Enter Kani

There are public elliptic curves E_0 and E_A, and a secret isogeny $\alpha : E_0 \to E_A$. Given the points P_B, Q_B on E_0 and $\alpha(P_B)$, $\alpha(Q_B)$, compute α. (modulo technical restrictions)*

Problem:
Not enough choices $\theta : E_A \to E_A$.
'No θ of degree N.'

Solution? $\theta : E_0 \times E_A \to E_0 \times E_A$?
$\rightsquigarrow$ still not enough. But! Kani's lemma:

- Constructs E_1, E_2 such that there exists a (structure-preserving) isogeny

$$E_1 \times E_A \to E_0 \times E_2$$

  of the right degree, N^2.
- Petit's trick then applies.

Recovering the secret

Finding the secret isogeny α of known degree.

$$\begin{tikzcd}
E_0 \arrow[loop left, "\iota"] \arrow[rr, bend left, "\alpha"] & & E_A \arrow[ll, bend left, "\widehat{\alpha}"]
\end{tikzcd}$$

Recovering the secret

Finding the secret isogeny α of known degree.

$$\begin{tikzcd} E_0 \arrow[rr, bend left, "\alpha"] & & E_A \\ E_1 \arrow[u, "\varphi"] & & \end{tikzcd}$$

Recovering the secret

Finding the secret isogeny α of known degree.

$$\begin{tikzcd}
E_0 \arrow[r, bend left, "\alpha"] & E_A \\
E_1 \arrow[u, dashed, "\varphi"] \arrow[ur, dashed, "f"] \arrow[r, dashed] & E_2 \arrow[u, dashed]
\end{tikzcd}$$

Kani's lemmaconstructs the above such that

$$\Phi = \begin{pmatrix} \varphi & -\widehat{\alpha} \\ * & * \end{pmatrix} : E_1 \times E_A \to E_0 \times E_2$$

is a structure preserving isogeny of degree N^2, and

$$\ker(\Phi) = \{(\deg(\alpha)P, f(P)) : P \in E_1[N]\}$$

$\rightsquigarrow$ can compute Φ and read off secret α!

Recovering the secret with Robert's trick

Finding the secret isogeny α of known degree.

$$\begin{tikzcd}
{E_0}^4 \arrow[rrr, bend left, "\alpha^4"] & & & {E_A}^4 \\
E_0^4 \arrow[u, dashed, "\varphi"] \arrow[urrr, dashed, "f"] \arrow[rrr, dashed] & & & E_A^4 \arrow[u, dashed]
\end{tikzcd}$$

constructs the above such that

$$\Phi = \begin{pmatrix} \varphi & -\widehat{\alpha}^4 \\ * & * \end{pmatrix} : E_0^4 \times E_A^4 \to E_0^4 \times E_A^4$$

is a structure preserving isogeny of degree N^2, and

$$\ker(\Phi) \text{ is known}$$

$\rightsquigarrow$ can compute Φ and read off secret α!

Power unleashed

Consequence 1: Factoring isogenies.

$$\begin{tikzcd}
E_0 \arrow[rrr, "\alpha"] & & & E_A \\
E_1 \arrow[u, "\varphi"] \arrow[urrr, dashed, "f"] \arrow[rrr, dashed] & & & E_2 \arrow[u, dashed]
\end{tikzcd}$$

Kani's lemma states that

$$\Phi = \begin{pmatrix} \varphi & -\widehat{\alpha} \\ * & * \end{pmatrix} : E_1 \times E_A \to E_0 \times E_2$$

is a structure preserving isogeny of degree B^2, and

$$\ker(\Phi) = \{(\deg(\alpha)P, f(P)) : P \in E_1[B]\}.$$

Power unleashed

Consequence 1: Factoring isogenies.

$$\begin{tikzcd}
E_0 \arrow[rr, "\alpha"] & & E_A \\
E_1 \arrow[u, "\varphi"] \arrow[urr, dashed, "f"] \arrow[rr, dashed] & & E_2 \arrow[u, dashed]
\end{tikzcd}$$

Kani's lemma states that

$$\Phi = \begin{pmatrix} \varphi & -\widehat{\alpha} \\ * & * \end{pmatrix} : E_1 \times E_A \to E_0 \times E_2$$

is a structure preserving isogeny of degree B^2, and

$$\ker(\Phi) = \{(\deg(\alpha)P, f(P)) : P \in E_1[B]\}.$$

Power unleashed

Consequence 2: Let

- $\alpha : E_0 \to E_A$ be an isogeny.
- B a smooth integer, $\langle P_B, Q_B \rangle = E_0[B]$.

Then:

- α can be stored efficiently as $\alpha(P_B)$, $\alpha(Q_B)$.
- images under α can be efficiently computed from this representation.
  Doesn't require $\deg(\alpha)$ to be smooth!

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

E_0

,

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1}$$

,

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A$$

,

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A \quad \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}^* \quad E_A$$

,

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A \xrightarrow{\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}_*} E_A$$

$$P_0, Q_0 \qquad\qquad P_{A,1}, Q_{A,1} \qquad\qquad P_A, Q_A \qquad\qquad \mu_1 P_A, \mu_2 Q_A$$

,

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A \xrightarrow{\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}_*} E_A$$

P_0, Q_0 $\qquad$ $P_{A,1}, Q_{A,1}$ $\qquad$ P_A, Q_A $\qquad$ $\mu_1 P_A, \mu_2 Q_A$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix},$

  ,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **KeyGen**

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A \quad \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}^* \quad E_A$$

$$P_0, Q_0 \qquad P_{A,1}, Q_{A,1} \qquad P_A, Q_A \qquad \mu_1 P_A, \mu_2 Q_A$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$

,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Bob: **Encrypt** $B \in \mathbf{Mat}_{2\times 2}[\mathbb{Z}/2^{3a}\mathbb{Z}]$

$$E_0 \xrightarrow{\varphi_{A,d_{A,1}}} E_{A,1} \xrightarrow{\varphi_{A,3^b}} E_A \quad \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}_* \quad E_A$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Bob: **Encrypt** $B \in \mathbf{Mat}_{2\times 2}[\mathbb{Z}/2^{3a}\mathbb{Z}]$

$$
\begin{tikzcd}
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"'] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] & E_A & \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}^* & E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] \\
E_1 & & & & E_2
\end{tikzcd}
$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Bob: **Encrypt** $B \in \mathbf{Mat}_{2\times 2}[\mathbb{Z}/2^{3a}\mathbb{Z}]$

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"'] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] & E_A \arrow[r, phantom, "\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} *"] & E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] \\
E_1 \arrow[d, dashed, "B*"'] & & & E_2 \arrow[d, dashed, "B*"] \\
E_1 & & & E_2
\end{tikzcd}$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Bob: **Encrypt** $B \in \mathbf{Mat}_{2\times 2}[\mathbb{Z}/2^{3a}\mathbb{Z}]$

$$\begin{tikzcd}
\begin{matrix} P_0 \\ Q_0 \end{matrix} & E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] & E_A \arrow[r, "\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} *"] & E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] & \begin{matrix} \mu_1 P_A \\ \mu_2 Q_A \end{matrix} \\
\begin{matrix} P_1 \\ Q_1 \end{matrix} & E_1 \arrow[d, dashed, "B*"] & & & E_2 \arrow[d, dashed, "B*"] & \begin{matrix} P_2 \\ Q_2 \end{matrix} \\
\begin{matrix} P_{1,B} \\ Q_{1,B} \end{matrix} & E_1 & & & E_2 & \begin{matrix} P_{2,B} \\ Q_{2,B} \end{matrix}
\end{tikzcd}$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$,

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Bob: **Encrypt** $B \in \mathbf{Mat}_{2\times 2}[\mathbb{Z}/2^{3a}\mathbb{Z}]$

$$
\begin{tikzcd}
\begin{smallmatrix} P_0 \\ Q_0 \end{smallmatrix} & E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, "\varphi_{B,d_1}"] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] & E_A \arrow[r, "\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} *"] & E_A \arrow[d, "\varphi_{B,3^{2b}}"] & \begin{smallmatrix} \mu_1 P_A \\ \mu_2 Q_A \end{smallmatrix} \\
\begin{smallmatrix} P_1 \\ Q_1 \end{smallmatrix} & E_1 \arrow[d, "B*"] & & & E_2 \arrow[d, "B*"] & \begin{smallmatrix} P_2 \\ Q_2 \end{smallmatrix} \\
\begin{smallmatrix} P_{1,B} \\ Q_{1,B} \end{smallmatrix} & E_1 & & & E_2 & \begin{smallmatrix} P_{2,B} \\ Q_{2,B} \end{smallmatrix}
\end{tikzcd}
$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$, $\mathrm{enc}(B) \leftarrow E_1, P_{1,B}, Q_{1,B}, E_2, P_{2,B}, Q_{2,B}$

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **Decrypt** B

$$
\begin{tikzcd}
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] & E_A \cdot \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} * \cdot E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] \\
E_1 \arrow[d, dashed, "B*"] & & E_2 \arrow[d, dashed, "B*"] \\
E_1 & & E_2
\end{tikzcd}
$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$, $\mathrm{enc}(B) \leftarrow E_1, P_{1,B}, Q_{1,B}, E_2, P_{2,B}, Q_{2,B}$

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **Decrypt** B

$$
\begin{tikzcd}
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"'] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] \arrow[d, dashed, "B*"] & E_A \arrow[r, dotted, "\begin{pmatrix}\mu_1 & \\ & \mu_2\end{pmatrix}*"] & E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] \\
E_1 \arrow[d, dashed, "B*"'] & E_{A,1} \arrow[rrd, "\deg = 3^{3b}"] & & E_2 \arrow[d, dashed, "B*"] \\
E_1 \arrow[ru, "\deg = d_1 d_{A,1}"] & & & E_2
\end{tikzcd}
$$

$P_{1,B}, Q_{1,B}$ $\qquad\qquad$ $\mu_1^{-1} d_1 P_{2,B}, \mu_2^{-1} d_1 Q_{2,B}$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix}\mu_1 & \\ & \mu_2\end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$, $\mathrm{enc}(B) \leftarrow E_1, P_{1,B}, Q_{1,B}, E_2, P_{2,B}, Q_{2,B}$

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **Decrypt** B

$$
\begin{tikzcd}
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, dashed, "\varphi_{B,d_1}"'] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] \arrow[d, dashed, "B*"] & E_A \arrow[r, dotted, "\begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} *"] & E_A \arrow[d, dashed, "\varphi_{B,3^{2b}}"] \\
E_1 \arrow[d, dashed, "B*"'] & E_{A,1} \arrow[drr, "\deg = 3^{3b}"] & & E_2 \arrow[d, dashed, "B*"] \\
E_1 \arrow[ur, "\deg = d_1 d_{A,1}"] & & & E_2
\end{tikzcd}
$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$, $\mathrm{enc}(B) \leftarrow E_1, P_{1,B}, Q_{1,B}, E_2, P_{2,B}, Q_{2,B}$

QFESTA: a PKE

Colour code: Public, Alice's secret, Bob's secret, unknown

Alice: **Decrypt** B

$$
\begin{tikzcd}
 & P_{A,1}, Q_{A,1} & & \\
E_0 \arrow[r, "\varphi_{A,d_{A,1}}"] \arrow[d, "\varphi_{B,d_1}"] & E_{A,1} \arrow[r, "\varphi_{A,3^b}"] \arrow[d, "B*"] & E_A \cdot \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix} * \cdot & E_A \arrow[d, "\varphi_{B,3^{2b}}"] \\
E_1 \arrow[d, "B*"] & E_{A,1} \arrow[rrd, "\deg = 3^{3b}"] & & E_2 \arrow[d, "B*"] \\
P_{1,B},\ Q_{1,B} \quad E_1 \arrow[ru, "\deg = d_1 d_{A,1}"] & & & E_2
\end{tikzcd}
$$

- $\mathsf{sk}_A \leftarrow E_{A,1}, P_{A,1}, Q_{A,1}, \begin{pmatrix} \mu_1 & \\ & \mu_2 \end{pmatrix}$, $\mathsf{pk}_A \leftarrow E_A, \mu_1 P_A, \mu_2 Q_A$
- $B \in \mathrm{Mat}_{2\times 2}$, $\mathrm{enc}(B) \leftarrow E_1, P_{1,B}, Q_{1,B}, E_2, P_{2,B}, Q_{2,B}$

Summary

Three main tools in isogeny-based cryptography:

- The class-group action.
  - NIKE: CRS, CSIDH, CSURF, SQALE, OSIDH (cf. Eli)
  - Signatures: Seasign, CSI-FISh, SCALLOP
- The Deuring correspondence.
  - Signatures: SQISign, SQISign2D (also uses Kani)
- Kani's lemma.
  - PKE: (Q)FESTA
  - Signatures: SQISign2D

Summary

Three main tools in isogeny-based cryptography:

- The class-group action.
  - NIKE: CRS, CSIDH, CSURF, SQALE, OSIDH (cf. Eli)
  - Signatures: Seasign, CSI-FISh, SCALLOP
- The Deuring correspondence.
  - Signatures: SQISign, SQISign2D (also uses Kani)
- Kani's lemma.
  - PKE: (Q)FESTA
  - Signatures: SQISign2D

Thank you!

References

[B²C²LMS²] ctidh.isogeny.org

[BD17] ia.cr/2017/334

[BDLS20] velusqrt.isogeny.org

[BEG19] ia.cr/2019/485

[BLMP19] quantum.isogeny.org

[CCJR22] ia.cr/2020/1520

[CD19] ia.cr/2019/1404

[CDV20] ia.cr/2020/1108

[FM19] ia.cr/2019/555

[GMT19] ia.cr/2019/431

[Wes21] ia.cr/2021/1583