Isogenies I & II

Lorenz Panny
Technische Universiteit Eindhoven

Executive School on Post-Quantum Cryptography,
Eindhoven, 2 July 2019
Please ask me anything!
Diffie–Hellman key exchange ’76

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

Fundamental reason this works:
- $a$ and $b$ are commutative!
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 $q$

\begin{align*}
\text{Alice} & \quad \text{public} \quad \text{Bob} \\
\begin{array}{c}
\begin{array}{c}
\text{random} \\
\text{random}
\end{array}
\end{array} & \begin{aligned}
\begin{array}{c}
\begin{array}{c}
\{0 \ldots q-1\} \\
\{0 \ldots q-1\}
\end{array}
\end{array}
\end{aligned}
\end{aligned}

\begin{align*}
s := (g^b)^a \\
&= \quad (g^a)^b
\end{align*}
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 $q$

Fundamental reason this works: $\cdot^a$ and $\cdot^b$ are commutative!
Diffie–Hellman: Bob vs. Eve

**Bob**

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

...  

$b−2$. Set $t \leftarrow t \cdot g$.

$b−1$. Set $t \leftarrow t \cdot g$.

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

Is this a good idea?

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

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

<table>
<thead>
<tr>
<th><strong>Bob</strong></th>
<th><strong>Attacker Eve</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set ( t \leftarrow g ).</td>
<td>1. Set ( t \leftarrow g ). If ( t = B ) return 1.</td>
</tr>
<tr>
<td>2. Set ( t \leftarrow t \cdot g ).</td>
<td>2. Set ( t \leftarrow t \cdot g ). If ( t = B ) return 2.</td>
</tr>
<tr>
<td>3. Set ( t \leftarrow t \cdot g ).</td>
<td>3. Set ( t \leftarrow t \cdot g ). If ( t = B ) return 3.</td>
</tr>
<tr>
<td>4. Set ( t \leftarrow t \cdot g ).</td>
<td>4. Set ( t \leftarrow t \cdot g ). If ( t = B ) return 3.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( b-2 ). Set ( t \leftarrow t \cdot g ).</td>
<td>( b-2 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b-2 ).</td>
</tr>
<tr>
<td>( b-1 ). Set ( t \leftarrow t \cdot g ).</td>
<td>( b-1 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b-1 ).</td>
</tr>
<tr>
<td>( b ). Publish ( B \leftarrow t \cdot g ).</td>
<td>( b ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b ).</td>
</tr>
<tr>
<td>( b+1 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b+1 ).</td>
<td>( b+1 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b+1 ).</td>
</tr>
<tr>
<td>( b+2 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b+2 ).</td>
<td>( b+2 ). Set ( t \leftarrow t \cdot g ). If ( t = B ) return ( b+2 ).</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

(This attacker is also kind of dumb, but that doesn’t matter for my point here.)
### 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 \( t \leftarrow t \cdot g \).

\[ \vdots \]

\( b-2 \). Set \( t \leftarrow t \cdot g \).
\( b-1 \). Set \( t \leftarrow t \cdot g \).
\( b \). Publish \( B \leftarrow t \cdot g \).

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

\[ \vdots \]

\( b-2 \). Set \( t \leftarrow t \cdot g \). If \( t = B \) return \( b-2 \).
\( b-1 \). Set \( t \leftarrow t \cdot g \). If \( t = B \) return \( b-1 \).
\( b \). Set \( t \leftarrow t \cdot g \). If \( t = B \) return \( b \).

\( b+1 \). Set \( t \leftarrow t \cdot g \). If \( t = B \) return \( b+1 \).
\( b+2 \). Set \( t \leftarrow t \cdot g \). If \( t = B \) return \( b+2 \).

\[ \vdots \]

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

(This attacker is also kind of dumb, but that doesn’t matter for my point here.)
Fast mixing: paths of length $\log(\#\text{nodes})$ to everywhere.
multiply

Fast mixing: paths of length \( \log(\# \text{nodes}) \) to everywhere.
Square-and-multiply
Square-and-multiply-and-square-and-multiply

Fast mixing: paths of length $\log(\# \text{nodes})$ to everywhere.
Square-and-multiply-and-square-and-multiply-and-squ
Square-and-multiply as graphs
Square-and-multiply as graphs

Fast mixing: paths of length $\log(\#\text{nodes})$ to everywhere.
Square-and-multiply as graphs

Fast mixing: paths of length \(\log(\#\text{nodes})\) to everywhere.
Square-and-multiply as graphs
Square-and-multiply as a graph

- Fast mixing: paths of length $\log(#\text{nodes})$ to everywhere.
Square-and-multiply as a graph

Fast mixing: paths of length $\log(\# \text{ nodes})$ to everywhere.
With square-and-multiply, applying $b$ takes $\Theta(\log \# G)$.

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

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

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

⇝ Exponential separation!

...and they lived happily ever after?
Shor’s algorithm quantumly computes $x$ from $g^x$ in any group in polynomial time.
Shor’s algorithm quantumly computes $x$ from $g^x$ in any group in polynomial time.

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

- Isogenies are a source of exponentially-sized graphs.
Big picture

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

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

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

- **Isogenies** are a source of *exponentially*-sized graphs.

- We can **walk efficiently** on these graphs.

- **Fast mixing**: short paths to (almost) all nodes.

- **No efficient*** algorithms to recover paths from endpoints. 
  *(Both classical and quantum!)*

- **Enough structure** to **navigate** the graph meaningfully.
  That is: some *well-behaved* ‘directions’ to describe paths. More later.
Big picture 🤔🤔

- **Isogenies** are a source of *exponentially*-sized graphs.

- We can **walk efficiently** on these graphs.

- **Fast mixing:** short paths to (almost) all nodes.

- **No efficient* algorithms to recover paths** from endpoints. *(Both classical and quantum!)*

- **Enough structure to navigate** the graph meaningfully. That is: some *well-behaved* ‘directions’ to describe paths. More later.

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

‘post-quantum Diffie–Hellman’.

(and more!)
Isogenies are well-behaved maps between elliptic curves.
Isogenies are well-behaved maps between elliptic curves.

Isogeny graph: Nodes are curves, edges are isogenies.
(We usually care about subgraphs with certain properties.)
The beauty and the beast

Components of well-chosen isogeny graphs look like this:
The beauty and the beast

Components of well-chosen isogeny graphs look like this:

Which of these is good for crypto?
The beauty and the beast

Components of well-chosen isogeny graphs look like this:

Which of these is good for crypto? **Both.**
The beauty and the beast

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

- **CSIDH** ['siː,saɪd]
  - $q = p$
  - [https://csidh.isogeny.org](https://csidh.isogeny.org)

- **SIDH**
  - $q = p^2$
  - [https://sike.org](https://sike.org)
CSIDH ['siːsaɪd]

(Castryck, Lange, Martindale, Panny, Renes; 2018)
Why CSIDH?

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

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

▶ Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)
Why CSIDH?

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

- Non-interactive key exchange (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)

- Small keys: starts at 64 bytes.*

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

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

- **Non-interactive key exchange** (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)

- **Small** keys: starts at 64 bytes.*

- **Competitive speed**: $\approx 55$ ms / full key exchange.* (Skylake)

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

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

▶ **Non-interactive key exchange** (full public-key validation); previously an open problem post-quantumly. (w/ reasonable speed)

▶ **Small** keys: starts at 64 bytes.*

▶ Competitive **speed**: \( \approx 55 \text{ ms} / \text{ full key exchange} \).* (Skylake)

▶ **Flexible**: compatible with 0-RTT protocols such as QUIC; yields signatures, (pre-quantum) VDFs, etc.

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

We’re going to do math.
An elliptic curve (modulo details) is given by an equation

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

A point on $E$ is a solution to this equation or the ‘fake’ point $\infty$. 

▶ The neutral element is $\infty$.

▶ The inverse of $(x, y)$ is $(x, -y)$.

▶ The sum of $(x_1, y_1)$ and $(x_2, y_2)$ is $$\left(\lambda^2 - x_1 - x_2, \lambda(x_1 + x_2 - \lambda^2) - y_1\right)$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $x_1 \neq x_2$ and $\lambda = 3x_1^2 + a$ otherwise.
An elliptic curve (modulo details) is given by an equation

\[ E: y^2 = x^3 + ax + b. \]

A point on \( E \) is a solution to this equation or the ‘fake’ point \( \infty \).

\( E \) is an abelian group: we can ‘add’ and ‘subtract’ points.

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

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

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

$$[m]: E \rightarrow E$$

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

$$E[m] \cong \mathbb{Z}/m \times \mathbb{Z}/m.$$
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 $\iota: (x, y) \mapsto (-x, \sqrt{-1} \cdot y)$ 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 $\{ \infty \}$. 
An isogeny of elliptic curves is a non-zero map $E \rightarrow E'$

- 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

\[
\{ y^2 = x^3 + x \} \rightarrow \{ y^2 = x^3 - 3x + 3 \}
\]

over $\mathbb{F}_{71}$. Its kernel is $\{(2, 9), (2, -9), \infty\}$. 
Choose some small odd primes $\ell_1, \ldots, \ell_n$.

Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.

Let $X = \{ y^2 = x^3 + Ax^2 + x \text{ over } \mathbb{F}_p \text{ with } p + 1 \text{ points} \}$.

Look at the $\ell_i$-isogenies defined over $\mathbb{F}_p$ within $X$.

Walk 'left' and 'right' on any $\ell_i$-subgraph is efficient.

We can represent $E \in X$ as a single coefficient $A \in \mathbb{F}_p$. 

$p = 419$

$\ell_1 = 3$

$\ell_2 = 5$

$\ell_3 = 7$
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.
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 $X = \{ y^2 = x^3 + Ax^2 + x \text{ over } \mathbb{F}_p \text{ with } p+1 \text{ points} \}$. 

$p = 419$

$\ell_1 = 3$

$\ell_2 = 5$

$\ell_3 = 7$
Choose some small odd primes $\ell_1, \ldots, \ell_n$.

Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.

Let $X = \{y^2 = x^3 + Ax^2 + x \text{ over } \mathbb{F}_p \text{ with } p+1 \text{ points}\}$.

Look at the $\ell_i$-isogenies defined over $\mathbb{F}_p$ within $X$. 

Walking 'left' and 'right' on any $\ell_i$-subgraph is efficient.

We can represent $E \in X$ as a single coefficient $A \in \mathbb{F}_p$. 

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

Walking ‘left’ and ‘right’ on any $\ell_i$-subgraph is efficient.
CSIDH in one slide

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

$p = 419$

$\ell_1 = 3$

$\ell_2 = 5$

$\ell_3 = 7$

- Walking ‘left’ and ‘right’ on any $\ell_i$-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 $\ell_i$-subgraph.

1. Find a point $(x, y) \in E$ of order $\ell_i$ with $x, y \in \mathbb{F}_p$.
   This uses scalar multiplication by $(p + 1)/\ell_i$.

2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Taking a ‘negative’ step on the $\ell_i$-subgraph.

1. Find a point $(x, y) \in E$ of order $\ell_i$ with $x \in \mathbb{F}_p$ but $y \not\in \mathbb{F}_p$.

2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Upshot: With ‘$x$-only arithmetic’ everything happens over $\mathbb{F}_p = \Rightarrow$ Efficient to implement!
Walking in the CSIDH graph

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

Taking a ‘negative’ step on the $\ell_i$-subgraph.
1. Find a point $(x, y) \in E$ of order $\ell_i$ with $x \in \mathbb{F}_p$ 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 $\mathbb{F}_p$.
$\Rightarrow$ Efficient to implement!
Walking in the CSIDH graph

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

Taking a ‘negative’ step on the $\ell_i$-subgraph.
1. Find a point $(x, y) \in E$ of order $\ell_i$ with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
   This uses scalar multiplication by $(p + 1)/\ell_i$.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ (see next slide).

Upshot: With ‘$x$-only arithmetic’ everything happens over $\mathbb{F}_p$.
$\implies$ Efficient to implement!
For any finite subgroup $G$ of $E$, there exists a unique\(^1\) separable isogeny $\varphi_G: E \to E'$ with kernel $G$.

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

If $G$ is defined over $k$, then $\varphi_G$ 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 $\varphi_G : E \rightarrow E'$ with kernel $G$.

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

If $G$ is defined over $k$, then $\varphi_G$ and $E/G$ are also defined over $k$.

Vélu ’71:
Formulas for computing $E/G$ and evaluating $\varphi_G$ at a point.
Complexity: $\Theta(\#G) \sim$ 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 $\varphi_G : E \to E'$ with kernel $G$.

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

If $G$ is defined over $k$, then $\varphi_G$ and $E/G$ are also defined over $k$.

Vélu ’71:
Formulas for computing $E/G$ and evaluating $\varphi_G$ at a point.

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

Vélu operates in the field where the points in $G$ live.
$\leadsto$ need to make sure extensions stay small for desired $\#G$
$\leadsto$ this is why we use special $p$ and curves with $p + 1$ points!

\(^1\)(up to isomorphism of $E'$)
CSIDH key exchange

Alice
[+, +, −, −]

Bob
[−, +, −, −]
CSIDH key exchange

Alice
$[\uparrow, \uparrow, -, -]$

Bob
$[\downarrow, \uparrow, -, -]$
CSIDH key exchange

Alice

Bob

\([+, +, -, -] \uparrow \)
CSIDH key exchange

Alice
[+, +, −, −]

Bob
[−, +, −, −]
CSIDH key exchange

Alice

[+, +, -, -]

Bob

[-, +, -, -]
CSIDH key exchange

Alice
\[+, +, -, -\]

Bob
\[-, +, -, -\]
CSIDH key exchange

Alice
\[ [+ , + , - , - ] \]

Bob
\[ [ - , + , - , - ] \]
CSIDH key exchange

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

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

Graphical representation of the key exchange protocol.
CSIDH key exchange

Alice

\[ [+ , + , - , - ] \]

Bob

\[ [- , + , - , - ] \]
CSIDH key exchange

Alice

Bob

$[+ , + , - , - ]$

$[- , + , - , - ]$
CSIDH key exchange

Alice
\[+, +, -, -\]

Bob
\[-, +, -, -\]
Has anyone seen my group action?

“CSIDH: an efficient post-quantum commutative group action”
Has anyone seen my group action?

“CSIDH: an efficient post-quantum commutative group action”

Cycles are compatible: \([\text{right then left}] = [\text{left then right}]\)

\[\Rightarrow\] only need to keep track of total step counts for each \(\ell_i\).

Example: \([+, +, -, -, -, +, -, -]\) just becomes \((+1, 0, -3) \in \mathbb{Z}^3\).
Has anyone seen my group action?

“CSIDH: an efficient post-quantum commutative group action”

Cycles are compatible: [right then left] = [left then right] 
→ only need to keep track of total step counts for each $\ell_i$.
Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, 0, -3) \in \mathbb{Z}^3$.

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

“CSIDH: an efficient post-quantum commutative group action”

Cycles are compatible: \([\text{right then left}] = [\text{left then right}]\)\n\(~\mapsto\) only need to keep track of total step counts for each \(\ell_i\).
Example: \([+, +, -, -, -, +, -, -]\) just becomes \((+1, 0, -3) \in \mathbb{Z}^3\).

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 \(\text{cl}(\mathbb{Z}[\sqrt{-p}])\).
Cryptographic group actions

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

Alice \hspace{1cm} public \hspace{1cm} Bob

\[ a \xleftarrow{\text{random}} G \quad \quad \quad b \xleftarrow{\text{random}} G \]

\[ a \ast s \quad \quad b \ast s \]

\[ key := a \ast (b \ast s) \quad \quad key := b \ast (a \ast s) \]
Why no Shor?

Recall from Dan’s talk:
Shor computes $\alpha$ from $h = g^\alpha$ by finding the kernel of the map

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

For general group actions, we cannot compose $a \ast s$ and $b \ast s$!
Security of CSIDH

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

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

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

Solving abelian hidden shift breaks CSIDH.

⇝ quantum subexponential attack (Kuperberg's algorithm).
Security of CSIDH

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

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

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

Solving abelian hidden shift breaks CSIDH.

\[ \leadsto \] quantum subexponential attack (Kuperberg's algorithm).
CSIDH vs. Kuperberg

Kuperberg’s algorithm consists of two components:
1. **Evaluate** the group action many times. (‘oracle calls’)
2. **Combine** the results in a certain way. (‘sieving’)

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?)

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

...but all known attacks cost $\exp\left(\frac{\log p}{2} + o(1)\right)$. $!^{23/33}$
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.
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 ≈ one bit operation? a hundred? millions?)
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?)

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

...but all known attacks cost $\exp\left(\left(\log p\right)^{1/2+o(1)}\right)!$
Can we avoid Kuperberg’s algorithm?

With great commutative group action comes great subexponential attack.
Can we avoid Kuperberg’s algorithm?

> With great commutative group action comes great subexponential attack.

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

- SIDH uses the full $\mathbb{F}_{p^2}$-isogeny graph. No group action!
Can we avoid Kuperberg’s algorithm?

With great commutative group action comes great subexponential attack.

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

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

- Problem: also no more intrinsic sense of direction.

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

  need extra information to let Alice & Bob’s walks commute.
Now: SIDH  (Jao, De Feo; 2011)

(...whose name doesn’t allow for nice pictures of beaches...)
“While several steps of SIDH involve complex isogeny calculations, the overall flow of SIDH for parties A and B is straightforward for those familiar with a Diffie–Hellman key exchange or its elliptic curve variant. [...]”
“While several steps of SIDH involve complex isogeny calculations, the overall flow of SIDH for parties A and B is straightforward for those familiar with a Diffie–Hellman key exchange or its elliptic curve variant. [...]”

**Setup.**

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

**Key exchange.**

1A. A generates two random integers $m_A, n_A < (w_A)^{e_A}$.
2A. A generates $R_A := m_A \cdot (P_A) + n_A \cdot (Q_A)$.
3A. A uses the point $R_A$ to create an isogeny mapping $\phi_A : E \rightarrow E_A$ and curve $E_A$ isogenous to $E$.
4A. A applies $\phi_A$ to $P_B$ and $Q_B$ to form two points on $E_A$: $\phi_A(P_B)$ and $\phi_A(Q_B)$.
5A. A sends to B $E_A, \phi_A(P_B)$, and $\phi_A(Q_B)$.

1B–4B. Same as A1 through A4, but with A and B subscripts swapped.

5B. B sends to A $E_B, \phi_B(P_A)$, and $\phi_B(Q_A)$.

6A. A has $m_A, n_A, \phi_B(P_A)$, and $\phi_B(Q_A)$ and forms $S_{BA} := m_A(\phi_B(P_A)) + n_A(\phi_B(Q_A))$.
7A. A uses $S_{BA}$ to create an isogeny mapping $\psi_{BA}$.
8A. A uses $\psi_{BA}$ to create an elliptic curve $E_{BA}$ which is isogenous to $E$.
9A. A computes $K := j$-invariant $(j_{BA})$ of the curve $E_{BA}$.

6B. Similarly, B has $m_B, n_B, \phi_A(P_B)$, and $\phi_A(Q_B)$ and forms $S_{AB} = m_B(\phi_A(P_B)) + n_B(\phi_A(Q_B))$.
7B. B uses $S_{AB}$ to create an isogeny mapping $\psi_{AB}$.
8B. B uses $\psi_{AB}$ to create an elliptic curve $E_{AB}$ which is isogenous to $E_k$.
9B. B computes $K := j$-invariant $(j_{AB})$ of the curve $E_{AB}$.

The curves $E_{AB}$ and $E_{BA}$ are guaranteed to have the same j-invariant.”
Alice & Bob pick secret subgroups $A$ and $B$ of $E$.

Alice computes $\varphi_A : E \rightarrow E/A$; Bob computes $\varphi_B : E \rightarrow E/B$.

(A 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 $(E/B)/A' \sim = E/\langle A, B \rangle \sim = (E/A)/B'$.
SIDH: High-level view

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

Alice computes $\varphi_A : E \rightarrow E/A$; Bob computes $\varphi_B : E \rightarrow E/B$. (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 $(E/B)/A' \sim = E/\langle A, B \rangle \sim = (E/A)/B'$. 

Diagram:

\[
\begin{array}{c}
E & \xrightarrow{\varphi_A} & E/A \\
\downarrow{\varphi_B} & & \downarrow{\varphi_B'} \\
E/B & \xrightarrow{\varphi_A'} & E/\langle A, B \rangle \\
\end{array}
\]
SIDH: High-level view

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

- Alice computes $\varphi_A : E \rightarrow E/A$; Bob computes $\varphi_B : E \rightarrow E/B$.
  (These isogenies correspond to walking on the isogeny graph.)
SIDH: High-level view

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

Alice computes $\varphi_A : E \rightarrow E/A$; Bob computes $\varphi_B : E \rightarrow E/B$.

(These isogenies correspond to walking on the isogeny graph.)

Alice and Bob transmit the values $E/A$ and $E/B$. 
SIDH: High-level view

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

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

Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
(These isogenies correspond to walking on the isogeny graph.)

Alice and Bob transmit the values $E/A$ and $E/B$.

Alice somehow obtains $A' := \varphi_B(A)$. (Similar for Bob.)

They both compute the shared secret

\[(E/B)/A' \cong E/\langle A, B \rangle \cong (E/A)/B'.\]
SIDH’s auxiliary points

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

Alice knows only $A$, Bob knows only $\varphi_B$. Hm.
SIDH’s auxiliary points

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

Alice knows only $A$, Bob knows only $\varphi_B$. Hm.

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

Previous slide: “Alice somehow obtains \( A' := \varphi_B(A) \).”

Alice knows only \( A \), Bob knows only \( \varphi_B \). Hm.

Solution: \( \varphi_B \) 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.

\( \implies \) Now Alice can compute \( A' \) as \( \langle \varphi_B(P) + [a]\varphi_B(Q) \rangle \)!

![Diagram showing the relationship between points and homomorphism](image-url)
SIDH in one slide

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

<table>
<thead>
<tr>
<th>Alice</th>
<th>public</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a ) \stackrel{\text{random}}{\leftarrow} {0 \ldots 2^n - 1}</td>
<td>( b ) \stackrel{\text{random}}{\leftarrow} {0 \ldots 3^m - 1}</td>
<td>( b ) \stackrel{\text{random}}{\leftarrow} {0 \ldots 3^m - 1}</td>
</tr>
<tr>
<td>( A := \langle P + [a]Q \rangle )</td>
<td>( B := \langle R + [b]S \rangle )</td>
<td>( B := \langle R + [b]S \rangle )</td>
</tr>
<tr>
<td>compute ( \varphi_A : E \rightarrow E/A )</td>
<td>compute ( \varphi_B : E \rightarrow E/B )</td>
<td>compute ( \varphi_B : E \rightarrow E/B )</td>
</tr>
<tr>
<td>( E/A, \varphi_A(R), \varphi_A(S) )</td>
<td>( E/B, \varphi_B(P), \varphi_B(Q) )</td>
<td>( E/B, \varphi_B(P), \varphi_B(Q) )</td>
</tr>
<tr>
<td>( A' := \langle \varphi_B(P) + [a]\varphi_B(Q) \rangle )</td>
<td>( B' := \langle \varphi_A(R) + [b]\varphi_A(S) \rangle )</td>
<td>( B' := \langle \varphi_A(R) + [b]\varphi_A(S) \rangle )</td>
</tr>
<tr>
<td>( s := j((E/B)/A') )</td>
<td>( s := j((E/A)/B') )</td>
<td>( s := j((E/A)/B') )</td>
</tr>
</tbody>
</table>
Decomposing smooth isogenies

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

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

Decomposing smooth isogenies

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

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

\[
\begin{array}{cccccc}
E \xrightarrow{\psi_1} E_1 & \xrightarrow{\psi_2} & \cdots & \xrightarrow{\psi_{k-1}} & E_{k-1} & \xrightarrow{\psi_k} E/G \\
\varphi_G
\end{array}
\]

Complexity: \( O(k^2 \cdot \ell) \). Exponentially smaller than \( \ell^k \).

BTW: The choice of \( p \) makes sure everything stays over \( F_{p^2} \).
Decomposing smooth isogenies

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

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

\[
\begin{array}{cccccc}
E & \xrightarrow{\psi_1} & E_1 & \xrightarrow{\psi_2} & \cdots & \xrightarrow{\psi_{k-1}} & E_{k-1} & \xrightarrow{\psi_k} & E/G \\
& \varphi_G & & & & & & \\
\end{array}
\]

\[\leadsto\] Complexity: \( O(k^2 \cdot \ell) \). Exponentially smaller than \( \ell^k \)!
‘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 \( \varphi_G : E \rightarrow E/G \).

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

\[
E \xrightarrow{\psi_1} E_1 \xrightarrow{\psi_2} \cdots \xrightarrow{\psi_{k-1}} E_{k-1} \xrightarrow{\psi_k} E/G
\]

\( \varphi_G \)

\( \sim \) Complexity: \( O(k^2 \cdot \ell) \). Exponentially smaller than \( \ell^k \! \).
  ‘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 $\varphi_A, \varphi_B$ is a walk of about $\log p/2$ steps.
(Alice & Bob can choose from about $\sqrt{p}$ secret keys each.)
The SIDH graph has size $\lfloor p/12 \rfloor + \varepsilon$. Each secret isogeny $\varphi_A, \varphi_B$ is a walk of about $\log p/2$ steps. (Alice & Bob can choose from about $\sqrt{p}$ secret keys each.)

Classical attacks:

- Cannot reuse keys without extra caution. (next slide)
- Meet-in-the-middle: $\tilde{O}(p^{1/4})$ time & space.
- Collision finding: $\tilde{O}(p^{3/8}/\sqrt{\text{memory/cores}})$. 
Security of SIDH

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

Classical attacks:
- Cannot reuse keys without extra caution. (next slide)
- Meet-in-the-middle: \( \tilde{O}(p^{1/4}) \) time & space.
- Collision finding: \( \tilde{O}(p^{3/8}/\sqrt{\text{memory/cores}}) \).

Quantum attacks:
- Claw finding: claimed \( \tilde{O}(p^{1/6}) \). Newer paper says \( \tilde{O}(p^{1/4}) \):
  “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 $s$. 


$\Rightarrow$ Validating that Bob is honest is $\approx$ as hard as breaking SIDH. Only usable with ephemeral keys or as a KEM 'SIKE'.
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 $A'' = \langle P' + [a]Q'' \rangle$. 

Validating that Bob is honest is $\approx$ as hard as breaking SIDH. Only usable with ephemeral keys or as a KEM 'SIKE'.
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 $a = 2u+1$: $[a]Q'' = [a]Q' + [u][2^n]P' + [2^{n-1}]P' = [a]Q' + [2^{n-1}]P'$. 

Validating that Bob is honest is $\approx$ as hard as breaking SIDH. Only usable with ephemeral keys or as a KEM 'SIKE'.

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

  $\implies$ Bob learns the parity of $a$. 

Validating that Bob is honest is $\approx$ as hard as breaking SIDH. Only usable with ephemeral keys or as a KEM 'SIKE'.

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

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

$\implies$ Bob learns the parity of $a$.

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

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

$\implies$ only usable with ephemeral keys or as a KEM ‘SIKE’.
Questions?