Isogeny-basd cryptography V

CSIDH

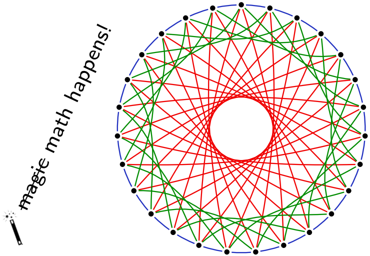
Tanja Lange
(with lots of slides by Lorenz Panny)

Eindhoven University of Technology

SAC – Post-quantum cryptography

CSIDH in one slide

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



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

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

Walking in the CSIDH graph

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

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ using Vélu's formulas.

Walking in the CSIDH graph

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

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the isogeny with kernel $\langle(x, y)\rangle$ using Vélu's formulas.

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

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
   Same test as above to find such a point.
2. Compute the isogeny with kernel $\langle(x, y)\rangle$ using Vélu's formulas.

Walking in the CSIDH graph

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

1. Find a point $(x, y) \in E$ of order ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ using Vélu's formulas.

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

1. Find a point $(x, y) \in E$ of order ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
   Same test as above to find such a point.
2. Compute the isogeny with kernel $\langle (x, y) \rangle$ using Vélu's formulas.

Upshot: With "x-only' arithmetic" everything happens over $\mathbb{F}_p$.

$\Longrightarrow$ Efficient to implement! There are several more speedups, such as pushing points through isogenies.

For math details see talk IV.

Abstract from Diffie-Hellman data flow

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

Abstract from Diffie-Hellman data flow

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

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

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

Abstract from Diffie-Hellman data flow

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

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

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

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

Abstract from Diffie-Hellman data flow

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

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

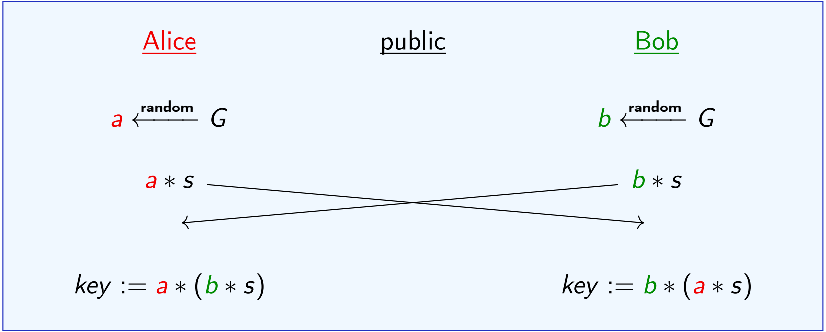
Example: [+, +, −, −, −, +, −, −] just becomes $(+1, \ 0, -3) \in \mathbb{Z}^3$.

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

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

Cryptographic group actions

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



Why no Shor?

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

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

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

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

CSIDH security

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

Size of key space:

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

Without quantum computer:

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

CSIDH security

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

Size of key space:

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

Without quantum computer:

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

With quantum computer:

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

CSIDH security:

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

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

Definition:

- $p = 4\prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \ldots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

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

Performance on typical Intel Skylake laptop core:

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

Security:

- Pre-quantum: at least 128 bits.

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

Definition:

- $p = 4\prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \ldots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

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

Performance on typical Intel Skylake laptop core:

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

Security:

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

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

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

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

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

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

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

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

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

  How to compare costs?

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

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

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

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

  How to compare costs?

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

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

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

Recent improvements to sieving target the $o(1)$.

Kuperberg applies to all commutative group actions.