

Disorientation attacks on CSIDH

Gustavo Banegas¹, Juliane Krämer², Tanja Lange³, Michael Meyer², Lorenz Panny⁴, Krijn Reijnders⁵, Jana Sotáková⁶, Monika Trimoska⁵

Inria and Laboratoire d'Informatique de l'École polytechnique¹, University of Regensburg²
Eindhoven University of Technology³, Academia Sinica⁴
Radboud University⁵, University of Amsterdam/QuSoft⁶,
j.s.sotakova@uva.nl <https://jana-sotakova.github.io/>

CSIDH group action

CSIDH group action [2]: Fix a prime $p = 4 \cdot \ell_1 \dots \ell_n - 1$ with ℓ_i odd primes. Get a regular action of $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ on $\mathcal{E} = \{E/\mathbb{F}_p \text{ supersingular and in Montgomery form } y^2 = x^3 + Ax^2 + x\}$. Write

$$(\mathbf{a}, E) \mapsto \mathbf{a} \star E.$$

Easy to act with ideals $\mathfrak{l}_i = (\ell_i, \sqrt{-p} - 1)$, for efficiency work with $\mathbf{a} = \prod_{i=1}^n \mathfrak{l}_i^{e_i}$ for $(e_1, \dots, e_n) \in \mathcal{K} \subset \mathbb{Z}^n$ for some explicit choice of *keyspace* \mathcal{K} .

Note: in CSIDH-like cryptosystems, secrets are which isogenies we compute, and how many times we do so.

Computing one ℓ -isogeny

Find a point of order ℓ in $E(\mathbb{F}_{p^2})$:

- to act by $\mathbf{a} = \mathfrak{l}$: find $P = (x, y)$ with $x, y \in \mathbb{F}_p$,
- to act by $\mathbf{a} = \mathfrak{l}^{-1}$: find $P = (x, y)$ with $x \in \mathbb{F}_p$ and $y \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$,

and compute the isogeny $E \rightarrow E/\langle P \rangle =: \mathbf{a} \star E$. We usually compute with x -coordinates only.

Computing group action

Sampling points of order ℓ is expensive; for efficiency, we always evaluate multiple isogeny steps (with the same orientation).

Call one iteration of the while loop a *round*.

Algorithm 1: Evaluation of CSIDH group action

Input: $A \in \mathbb{F}_p$ and a list of integers (e_1, \dots, e_n) .
Output: $B \in \mathbb{F}_p$ such that $\prod [\mathfrak{l}_i]^{e_i} \star E_A = E_B$

- 1: **while** some $e_i \neq 0$ **do**
- 2: Sample a random $x \in \mathbb{F}_p$, defining a point P .
- 3: Set $s \leftarrow \text{IsSquare}(x^3 + Ax^2 + x)$.
- 4: Let $S = \{i \mid e_i \neq 0, \text{sign}(e_i) = s\}$. **Restart** with new x if S is empty.
- 5: Let $k \leftarrow \prod_{i \in S} \ell_i$ and compute $Q \leftarrow [\frac{p+1}{k}]P$.
- 6: **for each** $i \in S$ **do**
- 7: Compute $R \leftarrow [\frac{k}{\ell_i}]Q$. If $R = \infty$, **skip** this i .
- 8: Compute $\phi: E_A \rightarrow E_B$ with kernel $\langle R \rangle$.
- 9: Set $A \leftarrow B$ and $k \leftarrow k/\ell_i$ and $Q \leftarrow \phi(Q)$ and $e_i \leftarrow e_i - s$.
- 10: **return** A .

(Fine print: specific implementations impose their own conditions on the set of indices in S , but always choose steps with the same orientation.)

Orientation

Point $P = (x, y) \in E(\mathbb{F}_{p^2})$ with $x \in \mathbb{F}_p$ is oriented

- *positively* if $x^3 + Ax^2 + x$ is a square in \mathbb{F}_p ,
- *negatively* if $x^3 + Ax^2 + x$ is a non-square in \mathbb{F}_p .

Denote the orientation of the point by s .

Note: positively-oriented points will allow steps in positive direction \mathfrak{l} , negatively in negative directions \mathfrak{l}^{-1} .

General disorientation

What if we disorient the point P used in **Algorithm 1**? Assume that we disoriented in round r . If P had full order and orientation s , then

$$E^{r,s} = \prod_{i \in S} \mathfrak{l}_i^{-2s} \star E_B.$$

If P did not have full order, we obtain a different curve

$$E_t = \prod_{\ell_i \nmid \text{ord}(P)} \mathfrak{l}_i^{2s} \star E^{r,s}.$$

Suppose we keep disorienting points at exactly the same point in the evaluation of **Algorithm 1**.

Observations:

1. $\ell_i \mid \text{ord}(P)$ is more likely than $\ell_i \nmid \text{ord}(P)$ and so the curve $E^{r,s}$ will be the most common one;
2. all other curves E_t are connected to $E^{r,s}$ by a short isogeny walk:

- (a) this walk only includes degrees ℓ_i for $i \in S$,
- (b) direction of these walks reveals the orientation of P (and hence all ℓ_i for $i \in S$).

(Fine print: more ‘torsion behavior’ is possible.)

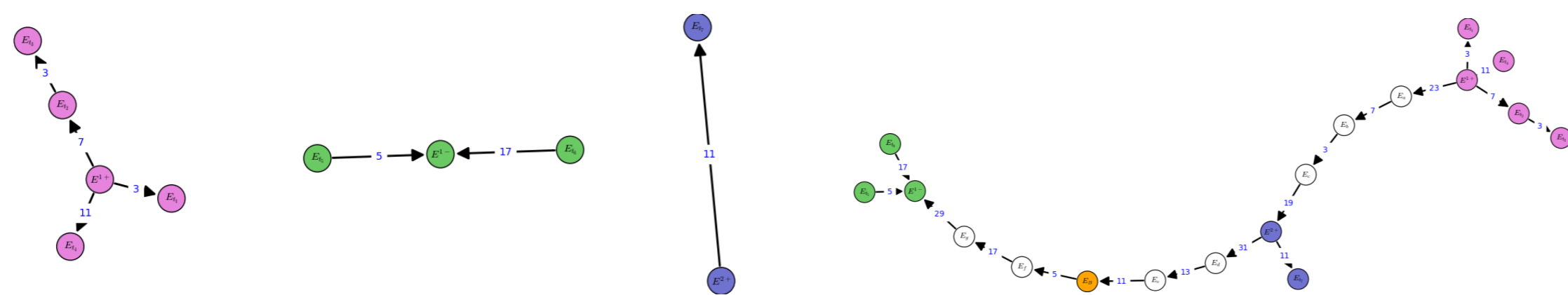
Toy example

We will illustrate the general algorithm to recover secret keys on a very toy example with 10 different primes ℓ_i and $-1 \leq e_i \leq 2$. For simplicity, assume $e_i \neq 0$, and let us examine what happens for the secret key (e_i) :

ℓ_i	3	5	7	11	13	17	19	23	29	31
e_i	1	-1	1	2	2	-1	1	1	-1	2

Assume we fault in round 1 and 2 of **Algorithm 1** repeatedly, generating faulty curves E_t . We also know the correct public key $E_B = \prod [\mathfrak{l}_i]^{e_i} \star E$.

1. From the faulty curves from round 1, pick the two most common ones, say $E^{1,a}$ and $E^{1,b}$ (we do not know the orientation yet). From the curves faulted in round 2, pick the most common curve not yet seen - this is most likely $E^{2,+}$.
2. Perform a small neighbourhood walk around the three curves and see if we see any of the faulty curves. We obtain three disjoint trees with the three curves as *roots*:



Note that the edges with labels ℓ_i actually corre-

spond to two steps in the isogeny graph.

3. This allows us to determine the orientation of the curves, and signs of e_i for some of the primes ℓ_i :
 - 3, 7, 11 point away from the root: positive;
 - 5, 17 point towards the root: negative.
4. Finally, we run **pubcrawl** to find the paths connecting the positive and negative curves: $E^{1,+} \rightarrow E^{2,+} \rightarrow E_B$ and $E_B \rightarrow E^{1,-}$. Note that no negatively oriented prime will ever occur in a positive path, and vice versa, which significantly speeds up the search.

5. Read off the secret key from the labels of the path!

Examples

CSIDH-512 [2] uses 74 primes $3, 5, \dots, 373, 587$, and the keyspace $\mathcal{K} = [-5, \dots, 5]^{74}$, so each $|e_i| \leq 5$.

Fault-injection attacks

Think of a device computing with secret data. Now consider the following magic power:

⚡ force a mistake at one point in the computation.

For instance, you can replace a value by a random value, or even skip an instruction (line) in the algorithm.

Disorientation

What if we want to compute $\mathfrak{l} \star E$ and generate a point P with the wrong orientation?

- What we wanted: $E_B = \mathfrak{l} \star E$,
- What we obtained: $E_B^t = \mathfrak{l}^{-1} \star E$.

These two curves are related: $E_B = \mathfrak{l}^2 \star E_B^t$.

How to force disorientations?

In **Algorithm 1**, we attack **Step 3: IsSquare** check is usually implemented as exponentiation $z \mapsto z^{\frac{p-1}{2}}$. Forcing a fault anywhere in this computation replaces the orientation of the point P with a random orientation, which is different from the orientation of P about half of the time. Another way to sample points is the Elligator 2 map, which can be attacked similarly.

Curves in different rounds

Notice that **Algorithm 1** is randomized: we will generate different points and orientations every time. Moreover, the computation in round r depends on what was computed in rounds $1, \dots, r-1$. Faulty curves from different rounds are again related by paths that reveal information on the secret key.

pubcrawl

Our main subroutine is finding a path in the isogeny graph between either the public key curve and a faulty curve, or between two faulty curves.

For this, we developed an optimized meet-in-the-middle brute-force search tool called **pubcrawl**. To find a path between E_1 to E_2 :

- specify primes ℓ_i to use as isogeny steps,
- specify orientation from E_1 to E_2 ,

and let **pubcrawl** do the work!

Results [1]

- We define a new class of fault attacks on CSIDH-like schemes we call *disorientation attacks*;
- We show that almost all current implementations are susceptible. In particular, batching techniques like SIMBA or CTIDH seem easier to attack because fewer isogenies are computed at each step;
- We argue these attacks are inherent to the way we compute group actions via isogenies, and so every cryptographic implementation needs to be strengthened. We propose lightweight countermeasures.
- We develop a tool **pubcrawl** for finding isogeny paths between (faulty) curves, optimized for the Meet-in-the-middle approach and for specifying the set of degrees among which we want to search;
- We consider a cryptographically more realistic scenario of not obtaining faulty curves E_t directly but only a *derived* value: think only seeing a *hash* $h(E_t)$.

References

- [1] G. Banegas, J. Krämer, T. Lange, M. Meyer, L. Panny, K. Reijnders, J. Sotáková, and M. Trimoska. “Disorientation attacks on CSIDH”. In: eprint soon!, 2022.
- [2] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. “CSIDH: An Efficient Post-Quantum Commutative Group Action”. In: *ASIACRYPT 2018*. Vol. 11274. LNCS. Springer, 2018, pp. 395–427.