CTIDH: Faster constant-time CSIDH

Gustavo Banegas Daniel J. Bernstein Fabio Campos
Tung Chou Tanja Lange Michael Meyer Benjamin Smith
Jana Sotáková

Affiliations and more at https://ctidh.isogeny.org/

April 5, 2022

CTIDH: Faster constant-time CSIDH

CSIDH [CLM+18]

is a post-quantum isogeny-based non-interactive key exchange protocol.

It uses a group action on a certain set of elliptic curves.

- Secret keys sampled from some keyspace sk $\in \mathcal{K}$ give group elements,
- Public keys are elliptic curves obtained by evaluating the group action $\star$

$$\mathrm{pk} = \mathrm{sk} \star E$$

CTIDH

is a new keyspace and a new constant-time algorithm for the group action in CSIDH.

- constant-time claims verified using `valgrind`
- speedups compared to previous best work:

`CSIDH-512`: 438006 multiplications (best previous 789000)
125.53 million Skylake cycles (best previous more than 200 million).

CTIDH: Faster constant-time CSIDH

CSIDH [CLM+18]

is a post-quantum isogeny-based non-interactive key exchange protocol.

It uses a group action on a certain set of elliptic curves.

- Secret keys sampled from some keyspace sk $\in \mathcal{K}$ give group elements,
- Public keys are elliptic curves obtained by evaluating the group action $\star$

$$\mathrm{pk} = \mathrm{sk} \star \mathrm{E}$$

CTIDH

is a new keyspace and a new constant-time algorithm for the group action in CSIDH.

- constant-time claims verified using `valgrind`
- speedups compared to previous best work:

`CSIDH-512`: 438006 multiplications (best previous 789000)
125.53 million Skylake cycles (best previous more than 200 million).

Today

1. CSIDH and the group action
2. Constant-time evaluation
3. Atomic blocks
4. New Keyspace
5. New algorithm and Matryoshka Isogeny

Supersingular elliptic curves

Start with a prime $p = 4 \cdot (\ell_1 \cdot \dots \cdot \ell_n) - 1$ with $\ell_1, \dots, \ell_n$ distinct odd primes.

Supersingular elliptic curves in Montgomery form

$E/\mathbb{F}_p$ supersingular elliptic curve with equation

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

Set of elliptic curves $\mathcal{E} = \{E_A : y^2 = x^3 + Ax^2 + x \text{ with } p+1 \text{ points over } \mathbb{F}_p\}$

Properties

- ✓ Abelian group with a algebraic group law,
- ✓ Montgomery form enables x-only arithmetic,
- ! The group structure

$$E(\mathbb{F}_p) \cong \mathbb{Z}/(p+1)\mathbb{Z} \cong \mathbb{Z}/4 \times \mathbb{Z}/\ell_1 \times \dots \times \mathbb{Z}/\ell_n$$

Supersingular elliptic curves

Start with a prime $p = 4 \cdot (\ell_1 \cdot \cdots \cdot \ell_n) - 1$ with $\ell_1, \ldots, \ell_n$ distinct odd primes.

Supersingular elliptic curves in Montgomery form

$E/\mathbb{F}_p$ supersingular elliptic curve with equation

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

Set of elliptic curves $\mathcal{E} = \{E_A : y^2 = x^3 + Ax^2 + x$ with $p + 1$ points over $\mathbb{F}_p\}$

Properties

- ✓ Abelian group with a algebraic group law,
- ✓ Montgomery form enables x-only arithmetic,
- ! The group structure

$$E(\mathbb{F}_p) \cong \mathbb{Z}/(p+1)\mathbb{Z} \cong \mathbb{Z}/4 \times \mathbb{Z}/\ell_1 \times \cdots \times \mathbb{Z}/\ell_n$$

Isogenies

Whenever have a point $P \in E(\mathbb{F}_p)$ of order ℓ, can construct an ℓ-isogeny:
a morphism of elliptic curves

$$\varphi : E_A \to E_{A'}$$

with kernel $\langle P \rangle$.

Unraveling the definition

- φ is given by rational maps in the x, y of E with coefficient in $\mathbb{F}_p$;
- φ is a group homomorphism: for all points Q and R we have

$$\varphi(Q + R) = \varphi(Q) + \varphi(R)$$

- the kernel of φ is the subgroup of E_A generated by P and has size ℓ;

! the isogeny acts like a "power-ℓ-map" on $E(\mathbb{F}_p)$:

if Q has order $\ell \cdot N$, then $\varphi(Q)$ has order N on $E_{A'}$

Computing an isogeny from a point

Suppose $P \in E(\mathbb{F}_p)$ is a point of order ℓ. Want to compute the isogeny with kernel $\langle P \rangle$:

$$\varphi : E_A \to E_{A'}$$

Recipe

1. Collect the points $\{[i]P : i \in S\}$ for some index set S,
2. Compute the product
$$h(X) = \prod_{i \in S} (x - x([i]P)),$$
3. Recover A' from $h(X)$

- Vélu's formulas [Vél71] use $S = \{1, 2, \ldots, \frac{\ell-1}{2}\}$;

  cost 6ℓ mult
- New $\sqrt{\text{é}}$lu formulas [BDFLS20] use $S = \{1, 3, 5, \ldots, \ell - 2\}$

  cost $\tilde{O}(\sqrt{\ell})$ mult

CSIDH magic

Prime $p = 4 \cdot (\ell_1 \dots \ell_n) - 1$,
set of elliptic curves $\mathcal{E} = \{E_A : y^2 = x^3 + Ax^2 + x \text{ with } p + 1 \text{ points}\}$

Every SEC has a distinguished ℓ_i-isogeny

For every $E_A \in \mathcal{E}$ and every $\ell \mid p + 1$, we can construct an ℓ-isogeny $\varphi : E_A \to E_{A'}$ using the points defined over $\mathbb{F}_p$:

$$E_A \longrightarrow E_{A'}$$

Claim

We have $E_{A'} \in \mathcal{E}$.

Group action

Complex multiplication magic

There is a finite abelian group G with a group action on $\mathcal{E}$ with the following properties:

- the action $E \mapsto g \star E$ is free and transitive action;
- for every $\ell_i \mid p+1$, there exists a group element g_i such that if $\varphi : E_A \to E_{A'}$ is the distinguished isogeny from before, then

$$g_i \star E_A = E_{A'}$$

- It only matters how many times we step in a particular direction, not the order in which we compute the isogenies.

Exponent vectors

Going back with isogenies

For every curve in $\mathcal{E}$ and every $\ell_i \mid p+1$, we have one ℓ_i-isogeny going forward, but also one going back:

$$E_A \xrightarrow{g_i} E_{A'} \xrightarrow{g_i^{-1}} E_A$$

This isogeny also easy to compute.

Exponent vector

$(e_1, \ldots, e_n) \in \mathbb{Z}^n$ encodes how many times we perform each isogeny.

$$(e_1, \ldots, e_n) : \qquad E_{A'} = \left(\prod_{i=1}^{n} g_i^{e_i} \right) \star E_A.$$

CSIDH key exchange

Diffie-Hellman flow

Alice and Bob agree on a starting curve $E_0 \in \mathcal{E}$:

1. Alice samples random exponent vector (e_i); Bob samples (f_i);
2. They compute action on E_0 as $E_A = \left(\prod g_i^{e_i}\right) \star E_0$ and $E_B = \left(\prod g_i^{f_i}\right) \star E_0$;
3. Exchange public keys: E_A, E_B;
4. They compute action on the curve just received:

$$\left(\prod g_i^{e_i}\right) \star E_B = \left(\prod g_i^{e_i+f_i}\right) \star E_0 = \left(\prod g_i^{f_i}\right) \star E_A$$

Constant-time evaluation

Secret keys $(e_1, \ldots, e_n) \in \mathbb{Z}^n$ used to evaluate the action

$$E_{A'} = \left(\prod_{i=1}^{n} g_i^{e_i} \right) \star E_A.$$

Every step is:

1. finding a point of order ℓ on some curve $E \in \mathcal{E}$,
2. an ℓ-isogeny computation from E.

Constant-time evaluation of the group action

If the input is a CSIDH curve and a private key, and the output is the result of the CSIDH action, then the algorithm time provides no information about the private key, and provides no information about the output.

Computing the group action

Computing one step

Simplified algorithm to compute the group action $E_{A'} = g_i \star E_A$ as an ℓ_i-isogeny:

1. find a point P of order ℓ_i on E_A:
   1.1 generate a point T of order $p+1$ on E_A,
   1.2 multiply $P = [\frac{p+1}{\ell_i}]T$.
2. Compute the ℓ_i-isogeny $\varphi : E_A \to E_{A'}$ with kernel P:
   2.1 enumerate the multiples $[i]P$ of the point P for $i \in S$,
   2.2 construct a polynomial $h(X) = \prod_{i \in S}(X - x([i]P))$,
   2.3 Compute the coefficient A' from $h(X)$.

Amortize the cost

Exponent vector $(1, 1, 1, 0, \ldots, 0)$

We compute ℓ_i-isogenies for $\ell_1 = 3$ and $\ell_2 = 5$ and $\ell_3 = 7$:

1. Find a suitable point:
   1.1 Generate a random point T of order $p + 1$,
   1.2 Compute $T_1 = \left[\frac{p+1}{3\cdot5\cdot7}\right] T$ has exact order ____
2. Compute the isogenies:
   2.1 3-isogeny:
      2.1.1 Compute $P_1 = [5 \cdot 7] T_1$ has order ____
      2.1.2 Use P_1 to construct 3-isogeny φ_1,
      2.1.3 Point $T_2 = \varphi_1(T_1)$ has order ____ on the new curve,
   2.2 5-isogeny:
      2.2.1 Compute $P_2 = [7] T_2$ has order ____,
      2.2.2 Construct 5-isogeny φ_2 with kernel P_2,
      2.2.3 The point $T_3 = \varphi_2(T_2)$ has order ____ on the new curve,
   2.3 7-isogeny: construct the isogeny φ_3 with kernel $P_3 = T_3$ which has order ____

Amortize the cost

Exponent vector $(1, 1, 1, 0, \ldots, 0)$

We compute ℓ_i-isogenies for $\ell_1 = 3$ and $\ell_2 = 5$ and $\ell_3 = 7$:

1. Find a suitable point:
   1.1 Generate a random point T of order $p + 1$,
   1.2 Compute $T_1 = \left[\frac{p+1}{3 \cdot 5 \cdot 7}\right] T$ has exact order ____
2. Compute the isogenies:
   2.1 3-isogeny:
      2.1.1 Compute $P_1 = [5 \cdot 7] T_1$ has order ____
      2.1.2 Use P_1 to construct 3-isogeny φ_1,
      2.1.3 Point $T_2 = \varphi_1(T_1)$ has order ____ on the new curve,
   2.2 5-isogeny:
      2.2.1 Compute $P_2 = [7] T_2$ has order ____,
      2.2.2 Construct 5-isogeny φ_2 with kernel P_2,
      2.2.3 The point $T_3 = \varphi_2(T_2)$ has order ____ on the new curve,
   2.3 7-isogeny: construct the isogeny φ_3 with kernel $P_3 = T_3$ which has order ____

Towards atomic blocks

Exponent vector $(1, 0, 1, 0, \ldots, 0)$

We compute ℓ_i-isogenies for $\ell_1 = 3$ and $\ell_3 = 7$ but no 5-isogeny:

1. Find a suitable point:
   1.1 Generate a random point T of order $p + 1$,
   1.2 Compute $T_1 = \left[\frac{p+1}{3 \cdot 5 \cdot 7}\right] T$ has exact order $3 \cdot 5 \cdot 7$,
2. Compute the isogenies:
   2.1 3-isogeny:
      2.1.1 Compute $P_1 = [5 \cdot 7] T_1$ has order 3,
      2.1.2 Use P_1 to construct 3-isogeny φ_1,
      2.1.3 Point $T_2 = \varphi_1(T_1)$ has order $5 \cdot 7$ on the new curve,
   2.2 No 5-isogeny:
      2.2.1 Compute the isogeny as before but throw away the results,
      2.2.2 Adjust to code to always compute $[5] T_2$,
      2.2.3 The point $T_3 = [5] T_2$ has order 7 on the same curve,
   2.3 7-isogeny: construct the isogeny φ_3 with kernel $P_3 = T_3$.

Atomic blocks

Definition (Atomic Blocks, simplified)

Let $I \subset \{1, \ldots, n\}$ be a subset of indices of size k, write $I = (i_1, \ldots, i_k)$.
An atomic block of length k is a probabilistic algorithm α_I:

- taking inputs A and $\epsilon \in \{0,1\}^k$,
- returning $A' \in \mathbb{F}_p$ such that $E_{A'} = (\prod_{j=1}^{k} g_{i_j}^{\epsilon_j}) \star E_A$,
- the time distribution of α_I is independent of ϵ.

Evaluating $3, 5$, and 7-isogeny

On the previous slide, we saw an atomic block α_I with $I = (1, 2, 3)$ that computes

$$E_{A'} = g_1^{\epsilon_1} g_2^{\epsilon_2} g_3^{\epsilon_3} \star E_A$$

for $(\epsilon_1, \epsilon_2, \epsilon_3) \in \{0,1\}^3$ without leaking timing information about $(\epsilon_1, \epsilon_2, \epsilon_3)$.

Why atomic blocks?

Definition (Atomic Blocks, simplified)

Let $I \subset \{1, \ldots, n\}$ be a subset of indices of size k, write $I = (i_1, \ldots, i_k)$.
An atomic block of length k is a probabilistic algorithm α_I:

- taking inputs A and $\epsilon \in \{0, 1\}^k$,
- returning $A' \in \mathbb{F}_p$ such that $E_{A'} = (\prod_{j=1}^{k} g_{i_j}^{\epsilon_j}) \star E_A$,
- the time distribution of α_I is independent of ϵ.

Because:

1. Previous CSIDH implementations are using atomic blocks implicitly;
2. Simpler framework to compute the group action:
   - 2.1 split the computation into atomic blocks independent of the secret;
   - 2.2 make sure each atomic block is constant-time.

Keyspace

Goal

For $(e_1, \ldots, e_n) \in \mathbb{Z}^n$, evaluate the group action

$$E_{A'} = \left(\prod_{i=1}^{n} g_i^{e_i} \right) \star E_A.$$

- Exponent vectors $(e_1, \ldots, e_n)$ sampled from some keyspace $\mathcal{K} \subset \mathbb{Z}^n$;
- Large enough keyspace: $\#\mathcal{K} \approx 2^{256}$;

Examples of keyspaces

1. Original CSIDH [CLM+18]: $|e_i| \leq m$ for all i with $(2m+1)^n \approx 2^{256}$,
2. [MCR19] use $0 \leq e_i \leq 10$ for `CSIDH-512`;
3. [CDRH20] allow the m_i to vary for efficiency.

Batching

Take `CSIDH-512` prime $p = 4 \cdot (3 \cdot 5 \cdot \cdots \cdot 373 \cdot 587) - 1$.

The batching idea

Consider exponent vector

primes	3	5	7	11	13	17	19	23	29	31	...
exponent vector	1	-2	0	3	-1	1	0	2	-1	0	...

New key space

Batching Keyspace

For B batches: For $N \in \mathbb{Z}^B_{>0}$ and $m \in \mathbb{Z}^B_{\geq 0}$, we define

$$\mathcal{K}_{N,m} := \left\{(e_1, \ldots, e_n) \in \mathbb{Z}^n \mid \sum_{j=1}^{N_i} |e_{i,j}| \leq m_i \text{ for } 1 \leq i \leq B\right\}.$$

Comparison for 6 primes

Atomic blocks for batches

Atomic blocks for batches

Suppose we have batches $\{3, 5, 7\}, \{11, 13, 17\}, \ldots$ And we want to compute one 5-isogeny and one 11-isogeny, i.e. exponent vector $(0, 1, 0, 1, 0, 0, 0, \ldots)$

1. Find a suitable point:
   1.1 Generate a random point T of order $p + 1$,
   1.2 Compute $T_1 = \left[\frac{p+1}{(3 \cdot 5 \cdot 7)(11 \cdot 13 \cdot 17)}\right] T$ has order $(3 \cdot 5 \cdot 7)(11 \cdot 13 \cdot 17)$.
2. Compute the isogenies:
   2.1 $\{3, 5, 7\}$-isogeny:
      2.1.1 Compute $P_1 = [(11 \cdot 13 \cdot 17)] T_1$ has order $(3 \cdot 5 \cdot 7)$,
      2.1.2 Use $[3 \cdot 7] P_1$ of order 5 to construct 5 -isogeny φ_1,
      2.1.3 Point $T_2 = [3 \cdot 7] \varphi_1(T_1)$ has order $11 \cdot 13 \cdot 17$ on the new curve,
   2.2 $\{11, 13, 17\}$-isogeny:
      2.2.1 Compute $P_2 = [13 \cdot 17] T_2$ has order 11,
      2.2.2 Construct 11-isogeny φ_2 with kernel P_2.

Atomic blocks for batches

Atomic blocks for batches

Suppose we have batches $\{3, 5, 7\}, \{11, 13, 17\}, \ldots$ And we want to compute one 5-isogeny and one 11-isogeny, i.e. exponent vector $(0, 1, 0, 1, 0, 0, 0, \ldots)$

1. Find a suitable point:
   1.1 Generate a random point T of order $p + 1$,
   1.2 Compute $T_1 = \left[\frac{p+1}{(3 \cdot 5 \cdot 7)(11 \cdot 13 \cdot 17)}\right] T$ has order $(3 \cdot 5 \cdot 7)(11 \cdot 13 \cdot 17)$.
2. Compute the isogenies:
   2.1 $\{3, 5, 7\}$-isogeny:
      2.1.1 Compute $P_1 = [(11 \cdot 13 \cdot 17)] T_1$ has order $(3 \cdot 5 \cdot 7)$,
      2.1.2 Use $[3 \cdot 7] P_1$ of order 5 to construct 5 -isogeny φ_1,
      2.1.3 Point $T_2 = [3 \cdot 7] \varphi_1(T_1)$ has order $11 \cdot 13 \cdot 17$ on the new curve,
   2.2 $\{11, 13, 17\}$-isogeny:
      2.2.1 Compute $P_2 = [13 \cdot 17] T_2$ has order 11,
      2.2.2 Construct 11-isogeny φ_2 with kernel P_2.

Matryoshka isogeny

How to construct the isogeny with the same code for all primes in the batch:

Matryoshka Isogeny for the batch $\{11, 13, 17\}$

Compute the 11-isogeny

1. enumerate the multiples $[i]P$ of the point P for $i \in S$, with $S = \{1, 2, \ldots, 5\}$
2. construct $h(X) = \prod_{i=1}^{5}(x - x([i]P))$,
3. Compute the coefficient A' from $h(X)$.

Matryoshka isogeny

How to construct the isogeny with the same code for all primes in the batch:

Matryoshka Isogeny for the batch $\{11, 13, 17\}$

Compute the $\cancel{11}$ 13-isogeny

1. enumerate the multiples $[i]P$ of the point P for $i \in S$, with $S = \{1, 2, \ldots, 5, 6\}$
2. construct $h(X) = \prod_{i=1}^{5}(x - x([i]P)) \cdot (x - x([6]P))$,
3. Compute the coefficient A' from $h(X)$.

Matryoshka isogeny

How to construct the isogeny with the same code for all primes in the batch:

Matryoshka Isogeny for the batch $\{11, 13, 17\}$

Compute the $\cancel{11}\ \cancel{13}\ 17$-isogeny

1. enumerate the multiples $[i]P$ of the point P for $i \in S$,
   with $S = \{1, 2, \ldots, 5, 6, 7, 8\}$
2. construct $h(X) = \prod_{i=1}^{5}(x - x([i]P)) \cdot (x - x([6]P)) \cdot (x - x([7]P))(x - x([8]P))$,
3. Compute the coefficient A' from $h(X)$.

Matryoshka isogenies

Matryoshka isogeny

- Compute the isogeny for any prime in the batch with the same code
- at the cost of computing isogeny for the largest prime,
- requires using dummy computations.

Known for Vélu formulas [BLMP19].
New for $\sqrt{\text{é}}$lu from [BDFLS20], newly used for batching.

Matryoshka for $\sqrt{\text{é}}$lu

The $\sqrt{\text{é}}$lu polynomial

Want to evaluate

$$h(X) = \prod_{i \in S} (x - x([i]P)),$$

for $S = \{1, 3, \ldots, \ell - 2\}$

Visual explanation for 29 and 31

1	9	17	25
3	11	19	27
5	13	21	
7	15	23	

Selection of the parameters

Evaluation cost function

Greedy algorithm to find efficient batching:

- For every batch configuration (number of batches, bounds of each batch), we can estimate the cost of the group action evaluation.
- Adaptively change batch configuration to find one with smaller cost (and large enough keyspace).

batch	size	primes	bound
1	2	3, 5	10
2	3	7, 11, 13	14
3	4	17, 19, 23, 29	16
4	4	31, 37, 41, 43	17
5	5	47, 53, 59, 61, 67	17
6	5	71, 73, 79, 83, 89	17
7	6	97, 101, 103, 107, 109, 113	18
8	7	127, 131, 137, 139, 149, 151, 157	18
9	7	163, 167, 173, 179, 181, 191, 193	18
10	8	197, 199, 211, 223, 227, 229, 233, 239	18
11	8	241, 251, 257, 263, 269, 271, 277, 281	18
12	6	283, 293, 307, 311, 313, 317	13
13	8	331, 337, 347, 349, 353, 359, 367, 373	13
14	1	587	1

valgrind constant time verification

Valgrind

Checking for constant-time

- We “poison” the secret data: declare undefined;
- *valgrind* will check if the undefined data corrupts branches or indices.

Speedups, comparison to previous works

pub	priv	DH	Mcyc	**M**	**S**	**a**	$1, 1, 0$	$1, 0.8, 0.05$	
512	220	1	89.11	228780	82165	346798	310945	311852	new
512	220	1	190.92	447000	128000	626000	575000	580700	[CCJR20]
512	220	2	93.23	238538	87154	361964	325692	326359	new
512	256	1	125.53	321207	116798	482311	438006	438762	new
512	256	1	—	624000	165000	893000	789000	800650	[ACR20]
512	256	2	129.64	330966	121787	497476	452752	453269	new
512	256	2	218.42	665876	189377	691231	855253	851939	[CDRH20]
512	256	2	238.51	632444	209310	704576	841754	835121	[HLKA20]
512	256	2	239.00	657000	210000	691000	867000	859550	[CCC+19]
512	256	2	—	732966	243838	680801	976804	962076	[OAYT19]
512	256	2	395.00	1054000	410000	1053000	1464000	1434650	[MCR19]
1024	256	1	469.52	287739	87944	486764	375683	382432	new
1024	256	1	—	552000	133000	924000	685000	704600	[ACR20]
1024	256	2	511.19	310154	99371	521400	409525	415721	new

Table: **pub**: size of p; **priv**: size of the keyspace; **DH** 1: group action evaluation, **DH** 2: group action evaluation and public key validation; **Mcyc** millions of cycles on a 3GHz Intel Xeon E3-1220 v5 (Skylake) CPU with Turbo Boost disabled; "**M**" multiplications; "**S**" squarings; "**a**" additions; "$1, 1, 0$" and "$1, 0.8, 0.05$" combinations of **M**, **S**, and **a**.

Summary

CTIDH

- New keyspace for CSIDH,
- New constant-time algorithm to evaluate the group action in CSIDH,
- Formalization of atomic blocks to compute the isogeny group action,
- constant-time verification using `valgrind`,
- speed records,

Find the article and the code at

https://ctidh.isogeny.org/

References I

Gora Adj, Jesús-Javier Chi-Domínguez, and Francisco Rodríguez-Henríquez.
On new Vélu's formulae and their applications to CSIDH and B-SIDH constant-time implementations, 2020.
https://eprint.iacr.org/2020/1109.

Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith.
Faster computation of isogenies of large prime degree, 2020.
https://eprint.iacr.org/2020/341.

Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny.
Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies, 2019.
https://eprint.iacr.org/2018/1059.

References II

Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith.
Stronger and faster side-channel protections for CSIDH, 2019.
https://eprint.iacr.org/2019/837.

Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez.
The SQALE of CSIDH: square-root Vélu quantum-resistant isogeny action with low exponents, 2020.
https://eprint.iacr.org/2020/1520.

Jesús-Javier Chi-Domínguez and Francisco Rodríguez-Henríquez.
Optimal strategies for CSIDH, 2020.
https://eprint.iacr.org/2020/417.

References III

- Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes.
  CSIDH: an efficient post-quantum commutative group action, 2018.
  https://eprint.iacr.org/2018/383.
- Aaron Hutchinson, Jason T. LeGrow, Brian Koziel, and Reza Azarderakhsh.
  Further optimizations of CSIDH: A systematic approach to efficient strategies, permutations, and bound vectors, 2020.
  https://eprint.iacr.org/2019/1121.
- Michael Meyer, Fabio Campos, and Steffen Reith.
  On Lions and Elligators: An efficient constant-time implementation of CSIDH, 2019.
  https://eprint.iacr.org/2018/1198.
- Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi.
  (Short paper) A faster constant-time algorithm of CSIDH keeping two points, 2019.
  https://eprint.iacr.org/2019/353.

References IV

Jacques Vélu.
Isogénies entre courbes elliptiques, 1971.
https://gallica.bnf.fr/ark:/12148/cb34416987n/date.