Stronger and Faster Side-Channel Protections for CSIDH

Daniel Cervantes-Vázquez [1] Mathilde Chenu [2,3]
Jesús-Javier Chi-Domínguez [1] and Luca De Feo [4] and
Francisco Rodríguez-Henríquez [1] and Benjamin Smith [2,3]

[1]Computer Science Department, Cinvestav - IPN, Mexico City, Mexico
[2]École polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[3]Inria, équipe-projet GRACE, Université Paris–Saclay, France
[4]Université Paris Saclay – UVSQ, Versailles, France

October 2, 2019

Overview

1. CSIDH overview
2. Constant-time CSIDH algorithm
3. Improvements to constant-time CSIDH algorithm
   - Fixing random point selection
   - Twisted Edwards or Montgomery curves?
   - Addition chains for a faster scalar multiplication
   - Removing dummy operations
4. Experimental results
5. Conclusions

Outline

Timeline of CSIDH

Before CSIDH (ordinary curves):

- Alexander Rostovtsev and Anton Stolbunov [10];
- Jean-Marc Couveignes [4];

Timeline of CSIDH

Before CSIDH (ordinary curves):

- Alexander Rostovtsev and Anton Stolbunov [10];
- Jean-Marc Couveignes [4];
- Anton Stolbunov [11];

Timeline of CSIDH

Before CSIDH (ordinary curves):

- Alexander Rostovtsev and Anton Stolbunov [10];
- Jean-Marc Couveignes [4];
- Anton Stolbunov [11];
- Luca De Feo, Jean Kieffer, and Benjamin Smith [5];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];
- Constant-time implementations:
  - August: Jalali *et al.* [6];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];
- Constant-time implementations:
  - August: Jalali *et al.* [6];
  - October: Bernstein, Lange, Martindale, and Panny [2];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];
- Constant-time implementations:
  - August: Jalali *et al.* [6];
  - October: Bernstein, Lange, Martindale, and Panny [2];
  - December: Meyer, Campos, and Reith [7];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];
- Constant-time implementations:
  - August: Jalali *et al.* [6];
  - October: Bernstein, Lange, Martindale, and Panny [2];
  - December: Meyer, Campos, and Reith [7];
  - April: Onuki, Aikawa, Yamazaki, and Takagi [9];

Timeline of CSIDH

CSIDH (supersingular curves):

- April: Castryck, Lange, Martindale, Panny, and Renes proposed CSIDH [3];
- August: Meyer and Reith [8];
- Constant-time implementations:
  - August: Jalali *et al.* [6];
  - October: Bernstein, Lange, Martindale, and Panny [2];
  - December: Meyer, Campos, and Reith [7];
  - April: Onuki, Aikawa, Yamazaki, and Takagi [9];
  - July: This work.

CSIDH implementations

- Castryck et al. [3]: The original CSIDH works on Montgomery curves;
- Jalali *et al.* [6] keep using Montgomery curves;
- Meyer and Reith [8]: Propose an hybrid CSIDH by using isogeny construction formulas but on Twisted Edwards curves, and then mapping into Montgomery form;
- Meyer–Campos–Reith [7], and Onuki et al. [9]: They keep using the hybrid CSIDH as in [8];

Our contributions

1) A fully Twisted Edwards version of CSIDH;
2) An efficient projective elligator;
3) The use of Shortest Differential Addition Chains (SDACs) in the CSIDH algorithm, which are cheaper than Classical Montgomery Ladders.
4) A stronger constant-time CSIDH algorithm without dummy operations.

CSIDH overview

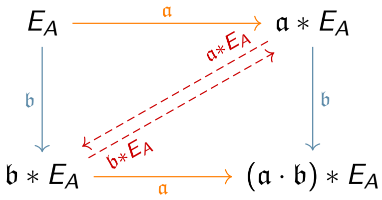
CSIDH framework [3]:

- Small odd primes numbers ℓ_i such that $p = 4\prod_{i=1}^{n} \ell_i - 1$ is prime number;
- Supersingular elliptic curves in Montgomery form $E_A/\mathbb{F}_p \colon y^2 = x^3 + Ax^2 + x$ with $\#E(\mathbb{F}_p) = p + 1$; and
- Positive integer m.

General description CSIDH:

The shared secret key is $(\mathfrak{a} \cdot \mathfrak{b}) * E_A$.

The security is given by the hardness of computing $\mathfrak{a}$ (or $\mathfrak{b}$) given the data colored in red ink.



CSIDH overview

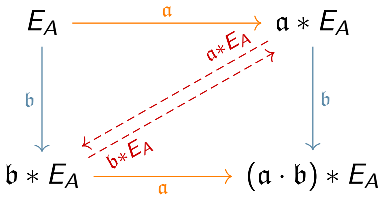
CSIDH framework [3]:

- Small odd primes numbers ℓ_i such that $p = 4\prod_{i=1}^{n} \ell_i - 1$ is prime number;
- Supersingular elliptic curves in Montgomery form $E_A/\mathbb{F}_p \colon y^2 = x^3 + Ax^2 + x$ with $\#E(\mathbb{F}_p) = p + 1$; and
- Positive integer m.

General description CSIDH:

The shared secret key is $(\mathfrak{a} \cdot \mathfrak{b}) * E_A$.

The security is given by the hardness of computing $\mathfrak{a}$ (or $\mathfrak{b}$) given the data colored in red ink.



Each ℓ_i is required e_i times for evaluating the *action* $\mathfrak{a} * E_A$ (similarly for $\mathfrak{b} * E_A$). Formally, this is written as $\mathfrak{a} = \mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$.

CSIDH overview

The action $\mathfrak{a} * E_A$ defines a path on the isogeny graph over $\mathbb{F}_p$, and is determined by an integer vector $(e_1, \ldots, e_n) \in [\![-m, m]\!]^n$:

1) Nodes are supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form;
2) Edges are degree-ℓ_i isogenies.

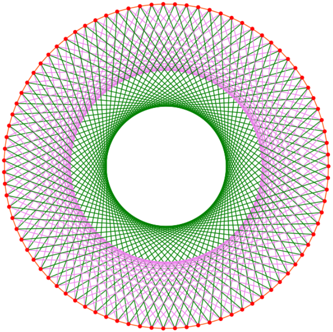


Figure 1: Isogeny graph over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Nodes are supersingular curves and edges marked with orange, green , and violet inks denote isogenies of degree 5, 13 and 61, respectively.

CSIDH overview

The action $\mathfrak{a} * E_A$ defines a path on the isogeny graph over $\mathbb{F}_p$, and is determined by an integer vector $(e_1, \ldots, e_n) \in [\![-m, m]\!]^n$:

1) Nodes are supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form;
2) Edges are degree-ℓ_i isogenies. Two types of edges: isogeny with kernel generated by
   - 2.a) $(x, y) \in E_A[\ell_i, \pi - 1]$, or
   - 2.b) $(x, iy) \in E_A[\ell_i, \pi + 1]$.

Here, $x, y \in \mathbb{F}_p$, $\pi\colon (X, Y) \mapsto (X^p, Y^p)$ is the Frobenius map, $i = \sqrt{-1}$ and thus $i^p = -i$.

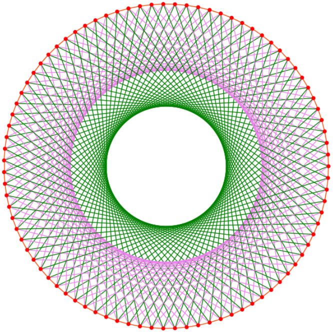


Figure 1: Isogeny graph over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Nodes are supersingular curves and edges marked with orange, green , and violet inks denote isogenies of degree 5, 13 and 61, respectively.

CSIDH overview

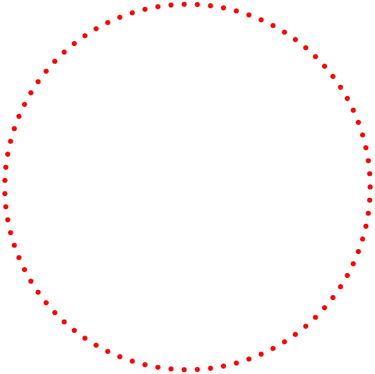


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

E_0

CSIDH overview

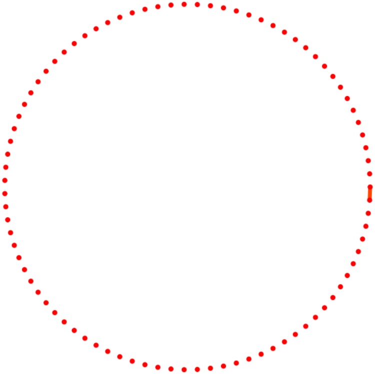


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \rightarrow E_{0x3A7D}$$

CSIDH overview

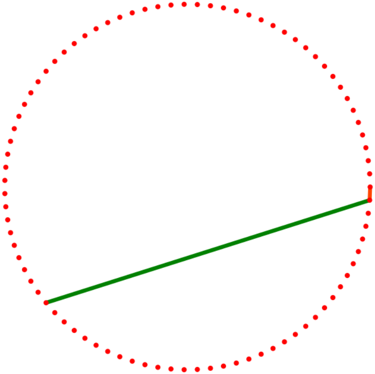


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \rightarrow E_{0x3A7D} \rightarrow E_{0x2BF7}$$

CSIDH overview

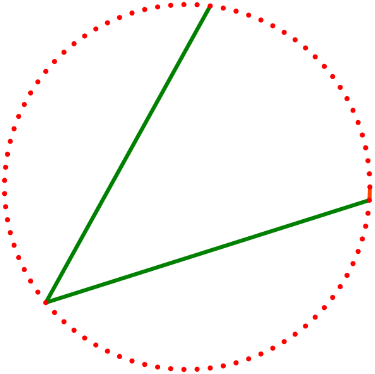


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \rightarrow E_{0x3A7D} \rightarrow E_{0x2BF7} \rightarrow E_{0x1404}$$

CSIDH overview

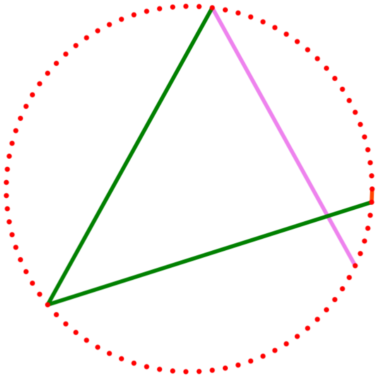


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \rightarrow E_{0x3A7D} \rightarrow E_{0x2BF7} \rightarrow E_{0x1404} \rightarrow E_{0x5EB}$$

CSIDH overview

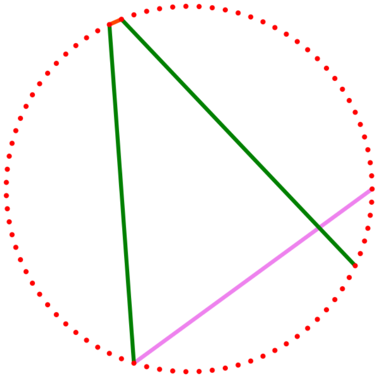


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. In general, the atcion evaluation is *commutative*. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \rightarrow E_{0x7A0} \rightarrow E_{0x8EC} \rightarrow E_{0x25B3} \rightarrow E_{0x5EB}$$

CSIDH overview

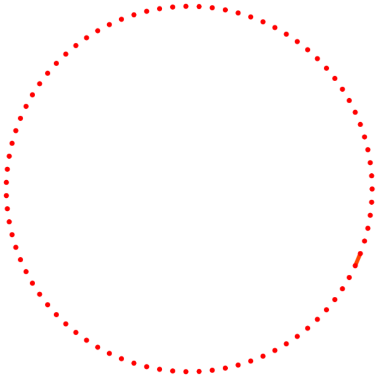


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(1, -2, -1) \in [\![-2, 2]\!]^3$ has *inverse* $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_{0x5EB} \rightarrow E_{0x1D50} \rightarrow E_{0x8EC} \rightarrow E_{0x56D} \rightarrow E_0$$

CSIDH overview

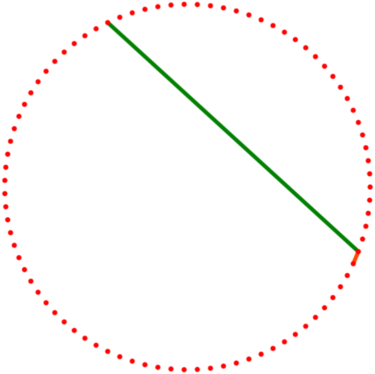


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(1, -2, -1) \in [\![-2, 2]\!]^3$ has *inverse* $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_{0x5EB} \rightarrow E_{0x1D50} \rightarrow E_{0x8EC} \rightarrow E_{0x56D} \rightarrow E_0$$

CSIDH overview

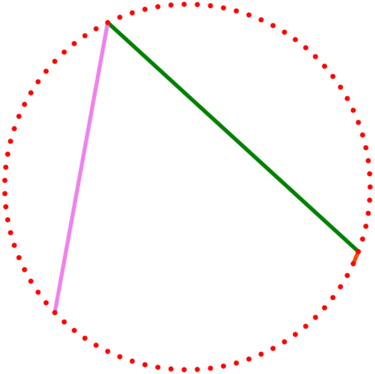


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(1, -2, -1) \in [\![-2, 2]\!]^3$ has *inverse* $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_{0x5EB} \rightarrow E_{0x1D50} \rightarrow E_{0x8EC} \rightarrow E_{0x56D} \rightarrow E_0$$

CSIDH overview

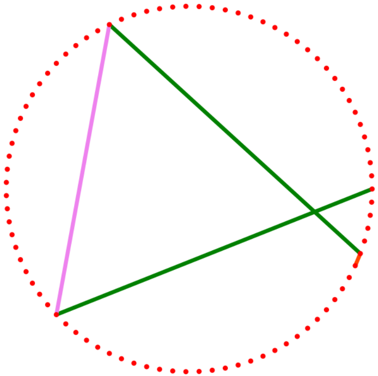


Figure 2: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(1, -2, -1) \in [\![-2, 2]\!]^3$ has *inverse* $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_{0x5EB} \rightarrow E_{0x1D50} \rightarrow E_{0x8EC} \rightarrow E_{0x56D} \rightarrow E_0$$

Outline

Constant-time CSIDH algorithm [7, 9]

In both the original CSIDH and the Onuki *et al.* variants $e_i \in [\![-m_i, m_i]\!]$, while in Meyer-Campos-Reith variant $e_i \in [\![0, m_i]\!]$. However, in constant-time implementations of CSIDH, the exponents e_i are implicitly interpreted as

$$|e_i| = \underbrace{1 + 1 + \cdots + 1}_{e_i \text{ times}} + \underbrace{0 + 0 + \cdots}_{m_i - e_i \text{ times}},$$

and then it starts by constructing isogenies with kernel generated by $P \in E_A[\ell_i, \pi - \mathtt{sign}(e_i)]$ for e_i iterations, then performs dummy isogeny computations for $(m_i - e_i) = 2k_i$ iterations.

Constant-time CSIDH algorithm [7, 9]

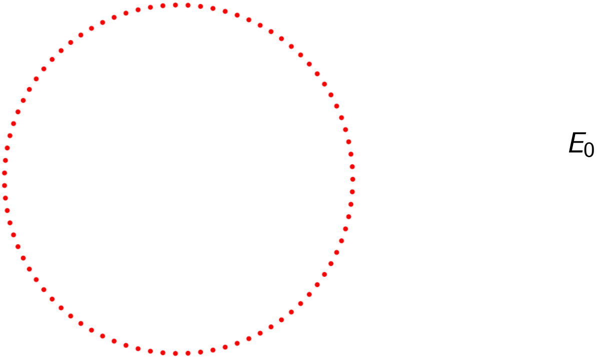
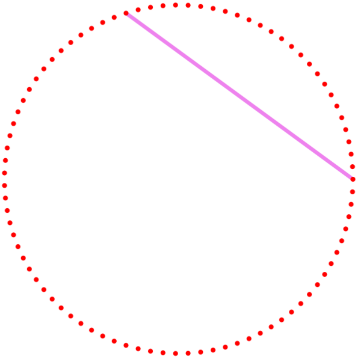


Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]

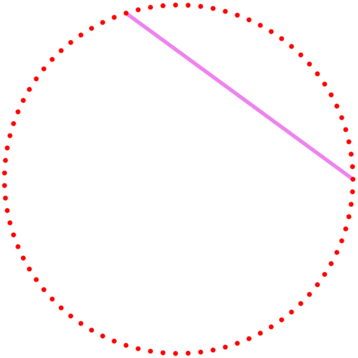


$E_0 \rightarrow$

$E_{0\text{x}3653}$

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]

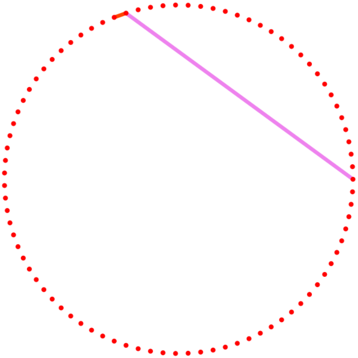


$E_0 \rightarrow$

$E_{0\times3653}$, dummy

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



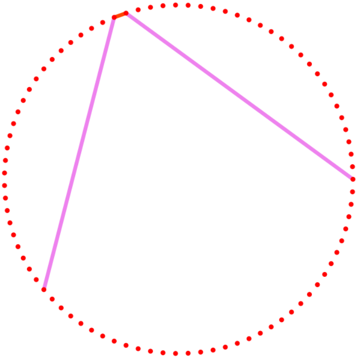
$E_0 \rightarrow$

$E_{0\text{x}3653}$, dummy $\rightarrow$

$E_{0\text{x}25\text{B}3}$

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

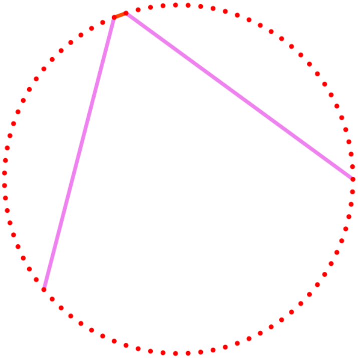
E_{0x3653}, dummy $\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

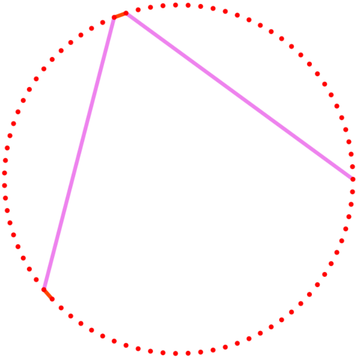
E_{0x3653}, dummy $\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

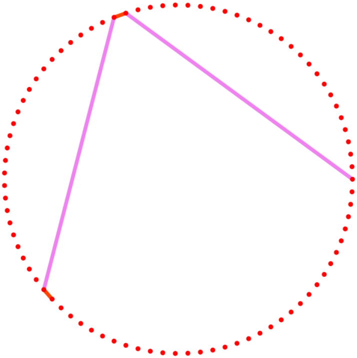
$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

E_{0x3653}, dummy$\rightarrow$

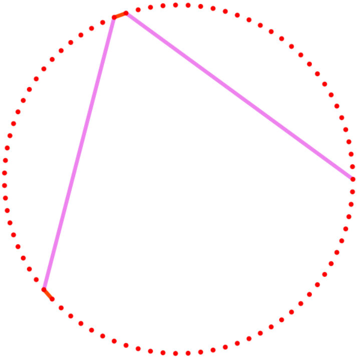
$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy$\rightarrow$

E_{0x56D}, dummy

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

E_{0x3653}, dummy$\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy$\rightarrow$

E_{0x56D}, dummy, dummy

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]

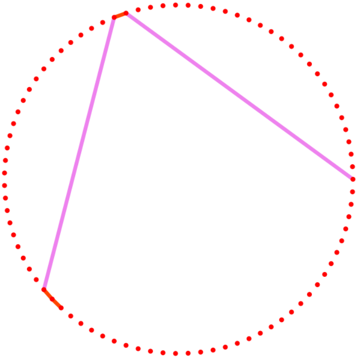


Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

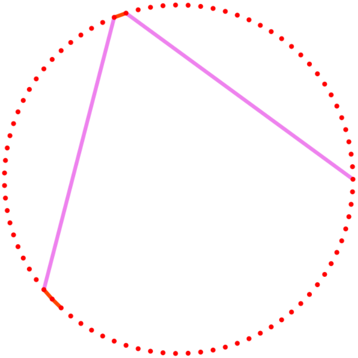
$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}, dummy, dummy $\rightarrow$

E_{0x24D5}

Constant-time CSIDH algorithm [7, 9]



$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}, dummy, dummy $\rightarrow$

E_{0x24D5}, dummy

Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [7, 9]

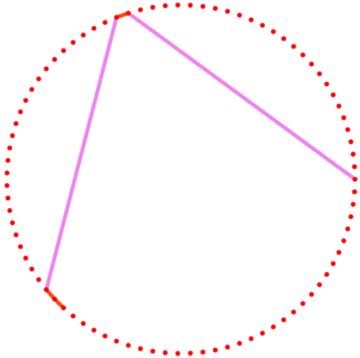


Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}, dummy, dummy $\rightarrow$

E_{0x24D5}, dummy, dummy

Constant-time CSIDH algorithm [7, 9]

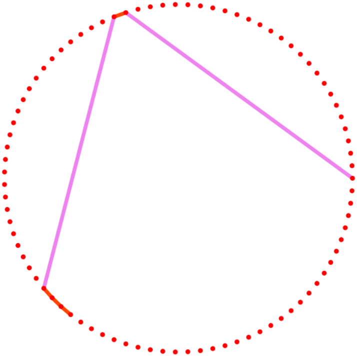


Figure 3: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}, dummy, dummy $\rightarrow$

E_{0x24D5}, dummy, dummy $\rightarrow$

E_{0x280E}

Outline

Outline

Issue with random point selection

In practice, one uses *Elligator*, which is an algorithm to efficiently sample points on a curve and its twist. However, elligator requires a random element $u \in \left[\!\left[2, \frac{p-1}{2}\right]\!\right]$ and also the inverse of $(u^2 - 1)$.

Issue with random point selection

In practice, one uses *Elligator*, which is an algorithm to efficiently sample points on a curve and its twist. However, elligator requires a random element $u \in \left[\!\left[2, \frac{p-1}{2}\right]\!\right]$ and also the inverse of $(u^2 - 1)$.

- To avoid a costly inversion of $u^2 - 1$: Meyer, Campos and Reith, and Onuki *et al.* follow [2] and precompute a set of ten pairs $(u, (u^2 - 1)^{-1})$;
- No randomness for u: elligator's output only depends on the A-coefficient of the current secret curve, which itself depends on the secret key.
- Running time of the algorithm varies and it is necessarily correlated to A and thus to the secret key.

Fixing random point selection

To avoid field inversions, we write $V = (A : u^2 - 1)$, and we determine whether V is the abscissa of a projective point on E_A. Plugging V into the homogeneous equation

$$E_A : Y^2Z^2 = X^3Z + AX^2Z^2 + XZ^3$$

gives

$$Y^2(u^2-1)^2 = \big((A^2u^2 + (u^2-1)^2\big)A(u^2-1).$$

We can test the existence of a solution for Y by computing the Legendre symbol of the right hand side: if it is a square, the points with projective XZ-coordinates

$$T_+ = (A : u^2 - 1), \qquad T_- = (-Au^2 : u^2 - 1)$$

are in $E_A[\pi - 1]$ and $E_A[\pi + 1]$ respectively, otherwise their roles are swapped.

Fixing random point selection

To avoid field inversions, we write $V = (A : u^2 - 1)$, and we determine whether V is the abscissa of a projective point on E_A. Plugging V into the homogeneous equation

$$E_A : Y^2Z^2 = X^3Z + AX^2Z^2 + XZ^3$$

gives

$$Y^2(u^2-1)^2 = \big((A^2u^2 + (u^2-1)^2\big)A(u^2-1).$$

We can test the existence of a solution for Y by computing the Legendre symbol of the right hand side: if it is a square, the points with projective XZ-coordinates

$$T_+ = (A : u^2 - 1), \qquad T_- = (-Au^2 : u^2 - 1)$$

are in $E_A[\pi - 1]$ and $E_A[\pi + 1]$ respectively, otherwise their roles are swapped. Consequently, u can be randomly chosen from $\left[\!\left[2, \frac{p-1}{2}\right]\!\right]$, and elligator's output only depends on randomness.

Outline

Twisted Edwards or Montgomery curves?

From [1], we see that the Twisted Edwards curve

$$E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2$$

is equivalent to the Montgomery curve

$$E_{(A:C)} : y^2 = x^3 + (A/C)x^2 + x$$

with constants

$$A_{24p} := A + 2C = a\,, \quad A_{24m} := A - 2C = d\,, \quad C_{24} := 4C = a - d\,.$$

In particular,

$$\psi : (X : Z) \longmapsto (Y : T) = (X - Z : X + Z)$$

ψ maps Montgomery XZ-coordinate points into Twisted Edwards YT-coordinate points, and

$$\psi^{-1} : (Y : T) \longmapsto (X : Z) = (T + Y : T - Y).$$

Twisted Edwards or Montgomery curves?

Using previous formulas, one can re-write the following Montgomery XZ-projective formulas in terms of Twisted Edwards YT-coordinates:

- Montgomery XZ-coordinates doubling
- Montgomery XZ-coordinates differential addition
- Montgomery XZ-coordinates degree-$(2k+1)$ isogeny evaluation.

Twisted Edwards or Montgomery curves?

Using previous formulas, one can re-write the following Montgomery XZ-projective formulas in terms of Twisted Edwards YT-coordinates:

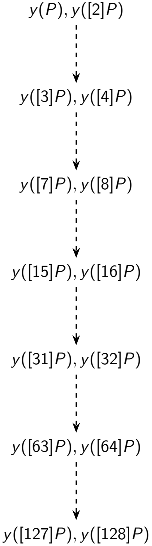
- Montgomery XZ-coordinates doubling
- Montgomery XZ-coordinates differential addition
- Montgomery XZ-coordinates degree-$(2k+1)$ isogeny evaluation.

In particular, the computational costs of doubling and differential addition in YT-coordinates are $4\mathbf{M} + 2\mathbf{S} + 4\mathbf{A}$, and $4\mathbf{M} + 2\mathbf{S} + 6\mathbf{A}$ (the same as XZ-coordinates).

Additionally, degree-$(2k+1)$ isogeny evaluation in XZ-coordinates costs $4k\mathbf{M} + 2\mathbf{S} + 6k\mathbf{A}$, whereas our YT-coordinate formula costs $4k\mathbf{M} + 2\mathbf{S} + (2k+4)\mathbf{A}$, thus saving $4k - 4$ field additions.

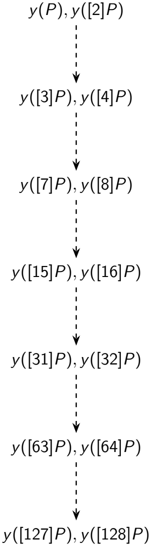
Outline

Classical Montgomery ladders



Example: given $y(P)$, $y([127]P)$ can be computed with 13 differential point operations.

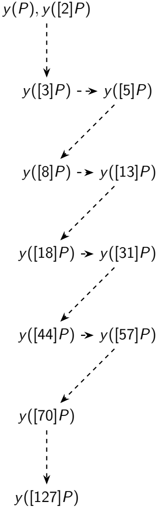
Classical Montgomery ladders



Example: given $y(P)$, $y([127]P)$ can be computed with 13 differential point operations.

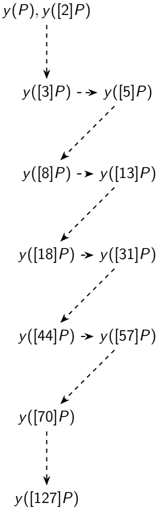
- Compute $y([\ell]P)$ requires $2\times\lceil\log_2 \ell\rceil - 1$ differential point operations.

Shortest differential addition chains (SDACs)



Example: given $y(P)$, $y([127]P)$ can be computed with 11 differential point operations.

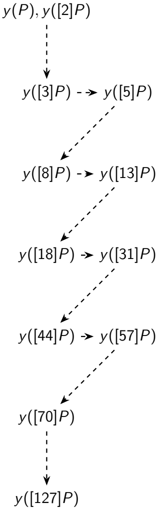
Shortest differential addition chains (SDACs)



Example: given $y(P)$, $y([127]P)$ can be computed with 11 differential point operations.

- Compute $y([\ell]P)$ requires $\approx 1.5 \times \lceil \log_2 \ell \rceil$ differential point operations,
- SDACs yields a saving of $\approx 25\%$ compared with the cost of the classical Montgomery ladder,

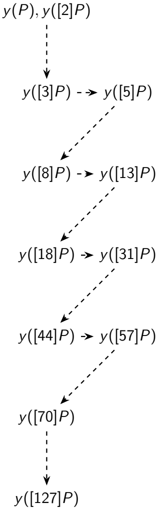
Shortest differential addition chains (SDACs)



Example: given $y(P)$, $y([127]P)$ can be computed with 11 differential point operations.

- Compute $y([\ell]P)$ requires $\approx 1.5 \times \lceil \log_2 \ell \rceil$ differential point operations,
- SDACs yields a saving of $\approx 25\%$ compared with the cost of the classical Montgomery ladder,
- SDACs are not constant-time,

Shortest differential addition chains (SDACs)



Example: given $y(P)$, $y([127]P)$ can be computed with 11 differential point operations.

- Compute $y([\ell]P)$ requires $\approx 1.5 \times \lceil \log_2 \ell \rceil$ differential point operations,
- SDACs yields a saving of $\approx 25\%$ compared with the cost of the classical Montgomery ladder,
- SDACs are not constant-time,
- But each scalar ℓ is public thus it's okay to use SDACs!

Outline

CSIDH with dummy operations

To mitigate power consumption analysis attacks, the constant-time algorithms proposed in [7] and [9] always compute the maximal amount of isogenies allowed by the exponent, using dummy isogeny computations if needed.

This implies that an attacker can obtain information on the secret key by injecting faults into variables during the computation. If the final result is correct, then she knows that the fault was injected in a dummy operation; if it is incorrect, then the operation was real.

Removing dummy operations

For our new approach, the exponents e_i are uniformly sampled from sets

$$\mathcal{S}(m_i) = \{e \mid e = m_i \bmod 2 \text{ and } |e| \leq m_i\},$$

i.e., centered intervals containing only even or only odd integers.

Removing dummy operations

For our new approach, the exponents e_i are uniformly sampled from sets

$$\mathcal{S}(m_i) = \{e \mid e = m_i \bmod 2 \text{ and } |e| \leq m_i\},$$

i.e., centered intervals containing only even or only odd integers. Consequently, the exponents e_i can implicitly interpreted as

$$|e_i| = \underbrace{1 + 1 + \cdots + 1}_{e_i \text{ times}} + \underbrace{(1-1) - (1-1) + (1-1) - \cdots}_{m_i - e_i \text{ times}},$$

and then our approach starts by constructing isogenies with kernel generated by $P \in E_A[\ell_i, \pi - \mathtt{sign}(e_i)]$ for e_i iterations, then alternates between isogenies with kernel generated by $P \in E_A[\ell_i, \pi - 1]$ and $P \in E_A[\ell_i, \pi + 1]$ for $(m_i - e_i) = 2k_i$ iterations.

Removing dummy operations

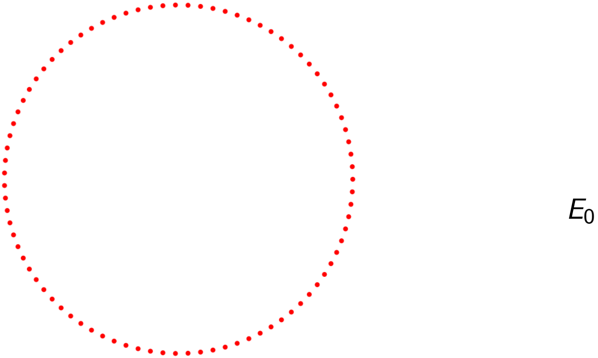
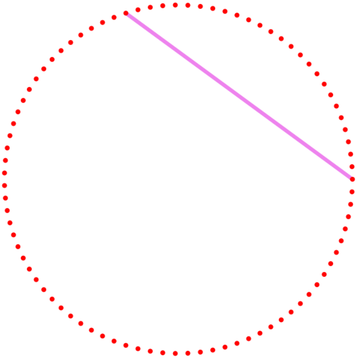


Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

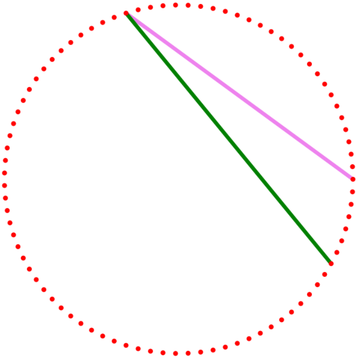
Removing dummy operations



$E_0 \rightarrow E_{0\times 3653}$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

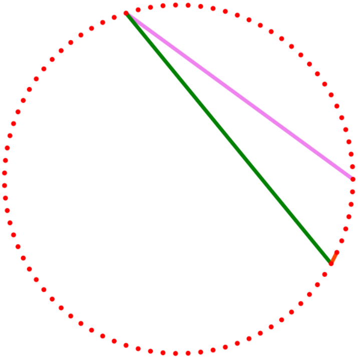
Removing dummy operations



$$E_0 \rightarrow E_{0\text{x}3653} \rightarrow E_{0\text{x}3C4A}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

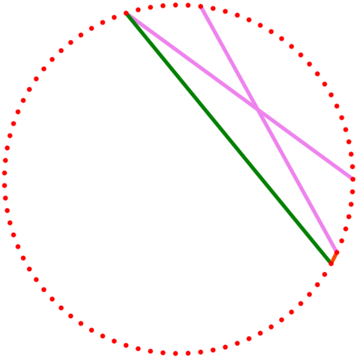
Removing dummy operations



$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

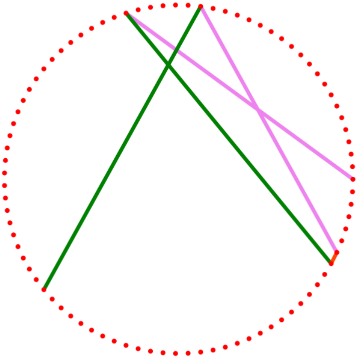
Removing dummy operations



$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB}$
$\to E_{0\text{x}1404}$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

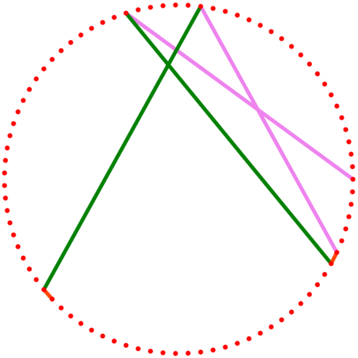
Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB} \rightarrow E_{0x1404} \rightarrow E_{0x2BF7}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

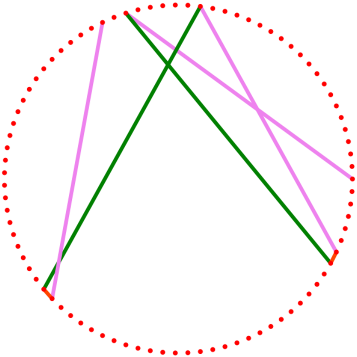
Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$$
$$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

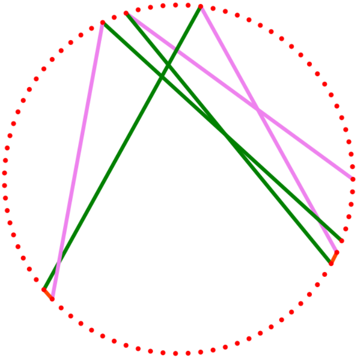
Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$$
$$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$$
$$\rightarrow E_{0x8EC}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

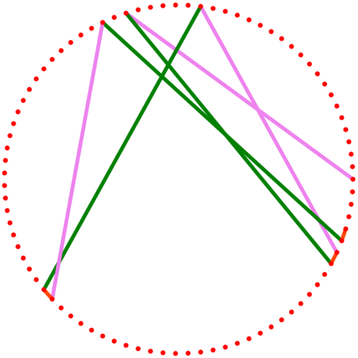
Removing dummy operations



$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$
$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$
$\rightarrow E_{0x8EC} \rightarrow E_{0x1D50}$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

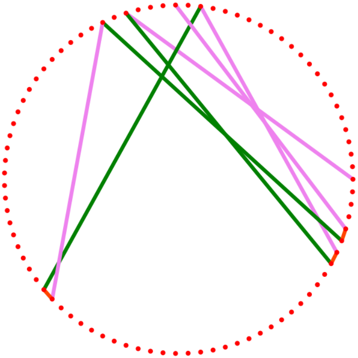
Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$$
$$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$$
$$\rightarrow E_{0x8EC} \rightarrow E_{0x1D50} \rightarrow E_{0x13F5}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

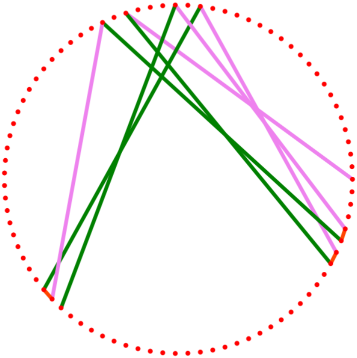
Removing dummy operations



$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB}$
$\to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D}$
$\to E_{0x8EC} \to E_{0x1D50} \to E_{0x13F5}$
$\to E_{0x1CDD}$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

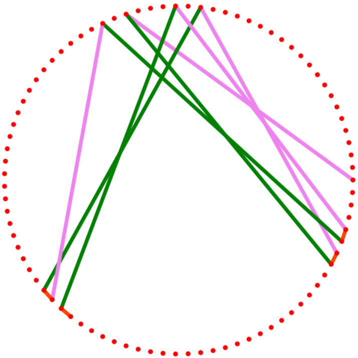
Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$$
$$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$$
$$\rightarrow E_{0x8EC} \rightarrow E_{0x1D50} \rightarrow E_{0x13F5}$$
$$\rightarrow E_{0x1CDD} \rightarrow E_{0x24D5}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Removing dummy operations



$$E_0 \rightarrow E_{0x3653} \rightarrow E_{0x3C4A} \rightarrow E_{0x5EB}$$
$$\rightarrow E_{0x1404} \rightarrow E_{0x2BF7} \rightarrow E_{0x56D}$$
$$\rightarrow E_{0x8EC} \rightarrow E_{0x1D50} \rightarrow E_{0x13F5}$$
$$\rightarrow E_{0x1CDD} \rightarrow E_{0x24D5} \rightarrow E_{0x280E}$$

Figure 4: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Outline

Running-time: field operations

Table 1: Field operation counts for constant-time CSIDH. Counts are given in millions of operations, averaged over 1024 random experiments. The performance ratio uses [7] as a baseline, considers only multiplication and squaring operations, and assumes $M = S$.

Implementation	CSIDH Algorithm	M	S	A	Ratio
Castryck et al. [3]	unprotected, unmodified	0.252	0.130	0.348	0.26
Meyer–Campos–Reith [7]	unmodified	1.054	0.410	1.053	1.00
Onuki et al. [9]	unmodified	0.733	0.244	0.681	0.67
This work	MCR-style	0.901	0.309	0.965	0.83
	OAYT-style	0.657	0.210	0.691	0.59
	No-dummy	1.319	0.423	1.389	1.19

Running-time: measured clock cycles

Table 2: Clock cycle counts for constant-time CSIDH implementations, averaged over 1024 experiments. The ratio is computed using [7] as baseline implementation.

Implementation	CSIDH algorithm	Mcycles	Ratio
Castryck et al. [3]	unprotected, unmodified	155	0.39
Meyer–Campos–Reith [7]	unmodified	395	1.00
This work	MCR-style	337	0.85
	OAYT-style	239	0.61
	No-dummy	481	1.22

Outline

1. CSIDH overview
2. Constant-time CSIDH algorithm
3. Improvements to constant-time CSIDH algorithm
    - Fixing random point selection
    - Twisted Edwards or Montgomery curves?
    - Addition chains for a faster scalar multiplication
    - Removing dummy operations
4. Experimental results
5. Conclusions

Conclusions

1) Previous implementations failed at being constant time because of a subtle mistake (Elligator was being used in an insecure way).
2) We fixed the problem, and proposed new improvements, to achieve the most efficient version of CSIDH protected against timing and simple power analysis attacks to date.
3) We proposed a protection against some fault-injection and timing attacks that only comes at a cost of a twofold slowdown.
4) We also sketched an alternative version of CSIDH "for the paranoid", with much stronger security guarantees, however at the moment this version seems too costly for the security benefits.

Further work

In SIDH one uses strategies for an efficient isogeny construction.
Thus, one could ask:

- Are strategies à la SIDH applicable to CSIDH?

Further work

In SIDH one uses strategies for an efficient isogeny construction.
Thus, one could ask:

- Are strategies à la SIDH applicable to CSIDH? Yes, they are!!!

Further work

In SIDH one uses strategies for an efficient isogeny construction.
Thus, one could ask:

- Are strategies à la SIDH applicable to CSIDH? Yes, they are!!!
- Do strategies à la SIDH help to improve CSIDH?

Further work

In SIDH one uses strategies for an efficient isogeny construction. Thus, one could ask:

- Are strategies à la SIDH applicable to CSIDH? Yes, they are!!!
- Do strategies à la SIDH help to improve CSIDH? We will know in a couple of days!!!

Thank you for your attention

I look forward to your comments and questions.

e-mail: jjchi@computacion.cs.cinvestav.mx

Our software library is freely available from

https://github.com/JJChiDguez/csidh.

We thank Prof. Onuki for his comments about an incorrect claim in an earlier version of this work.

References I

- Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters.

  Twisted Edwards curves.

  In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2008.

- Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny.

  Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies.

  In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, pages 409–441, 2019.

References II

- Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes.

  CSIDH: an efficient post-quantum commutative group action.

  In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, pages 395–427, 2018.

- Jean Marc Couveignes.

  Hard homogeneous spaces.

  Cryptology ePrint Archive, Report 2006/291, 2006.

References III

- Luca De Feo, Jean Kieffer, and Benjamin Smith.

  Towards practical key exchange from ordinary isogeny graphs.

  In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, pages 365–394, 2018.

- Amir Jalali, Reza Azarderakhsh, Mehran Mozaffari Kermani, and David Jao.

  Towards optimized and constant-time CSIDH on embedded devices.

  In *Constructive Side-Channel Analysis and Secure Design*, pages 215–231. Springer International Publishing, 2019.

References IV

- Michael Meyer, Fabio Campos, and Steffen Reith.

  On lions and elligators: An efficient constant-time implementation of CSIDH.

  In *Post-Quantum Cryptography - 10th International Workshop, PQCrypto 2019*, 2019.

- Michael Meyer and Steffen Reith.

  A faster way to the CSIDH.

  In *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, pages 137–152, 2018.

- Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi.

  A faster constant-time algorithm of CSIDH keeping two torsion points.

  To appear in IWSEC 2019 – The 14th International Workshop on Security, 2019.

References V

- Alexander Rostovtsev and Anton Stolbunov.
  Public-key cryptosystem based on isogenies.
  Cryptology ePrint Archive, Report 2006/145, 2006.
- Anton Stolbunov.
  Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves.
  Advances in Mathematics of Communication, 4(2), 2010.