A Constant-time Algorithm of CSIDH keeping Two Points

Hiroshi Onuki[1] Yusuke Aikawa[2] Tsutomu Yamazaki[3]
Tsuyoshi Takagi[1]

[1]The University of Tokyo

[2]Mitsubishi Electric

[3]Kyushu University

2020/2/21

Table of contents

Table of contents

Overview

- We constructed a constant-time algorithm of an isogeny-based cryptography CSIDH.
- Our algorithm is about 29% faster than a previous work.

Table of contents

Post Quantum Cryptography

- RSA and ECC will be broken if a quantum computer is built.

$\Rightarrow$ **Post Quantum Cryptography (PQC) is important.**

- NIST started PQC standardization process in 2016.
- The candidates include an **isogeny-based cryptography**.
    - SIKE (Supersingular Isogeny Key Encapsulation).

Isogeny-based cryptography (1/3)

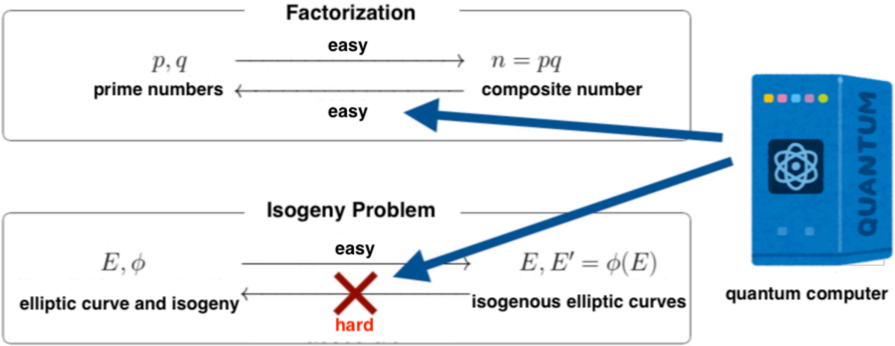
Isogeny-based cryptography is

- a cryptosystem based on **isogeny problem**,
- first proposed by Couvegne and independently by Rostovtsev and Stolbunov.

Isogeny-based cryptography (1/3)

Isogeny-based cryptography is

- a cryptosystem based on **isogeny problem**,
- first proposed by Couvegne and independently by Rostovtsev and Stolbunov.



⇒ **Isogeny-based cryptography is a candidate for PQC.**

Isogeny-based cryptography (2/3)

Rough sketch of isogeny-based key exchange:

$$\begin{array}{ccc} E & \xrightarrow{\varphi_B} & E_B \\ {\scriptstyle \varphi_A}\downarrow & & \downarrow{\scriptstyle \varphi_A} \\ E_A & \xrightarrow[\varphi_B]{} & E_{AB} \end{array}$$

E : public elliptic curve
φ_A : Alice's secret key, E_A : Alice's public key
φ_B : Bob's secret key, E_B : Bob's public key
E_{AB} : shared key

Isogeny-based cryptography (3/3)

Pros

- Short key size
- Various techniques for ECC can be applied
- Many applications (signature, hash, ...)

Cons

- Slow

SIDH & CSIDH

SIDH is

- Supersingular Isogeny Diffie Hellman,
- proposed by Jao and Feo at PQCrypto 2011.

The isogeny-based candidate for NIST PQC is based on SIDH.

SIDH & CSIDH

SIDH is

- Supersingular Isogeny Diffie Hellman,
- proposed by Jao and Feo at PQCrypto 2011.

The isogeny-based candidate for NIST PQC is based on SIDH.

CSIDH is

- Commutative SIDH,
- proposed by Castryck et al. at ASIACRYPT 2018.

Table of contents

CSIDH (1/3)

p : a prime,

$\mathcal{E}\ell\ell = \{E$: supersingular e.c. over $\mathbb{F}_p \mid \mathrm{End}_{\mathbb{F}_p}(E) \cong \mathbb{Z}[\sqrt{-p}]\}/\sim_{\mathbb{F}_p}$,

$\mathcal{C}\ell$: the ideal class group of $\mathbb{Z}[\sqrt{-p}]$.

CSIDH (1/3)

p : a prime,

$\mathcal{E}\ell\ell = \{E : \text{supersingular e.c. over } \mathbb{F}_p \mid \mathrm{End}_{\mathbb{F}_p}(E) \cong \mathbb{Z}[\sqrt{-p}]\}/\sim_{\mathbb{F}_p}$,

$\mathcal{C}\ell$: the ideal class group of $\mathbb{Z}[\sqrt{-p}]$.

Proposition 1

$\mathcal{C}\ell$ acts freely and transitively on $\mathcal{E}\ell\ell$ via isogenies.

$$\begin{array}{ccc} \mathcal{C}\ell \times \mathcal{E}\ell\ell & \to & \mathcal{E}\ell\ell \\ \Downarrow & & \Downarrow \\ (\mathfrak{a}, E) & \mapsto & \mathfrak{a} * E \end{array}$$

CSIDH (1/3)

p : a prime,

$\mathcal{E}\ell\ell = \{E$: supersingular e.c. over $\mathbb{F}_p \mid \mathrm{End}_{\mathbb{F}_p}(E) \cong \mathbb{Z}[\sqrt{-p}]\}/\sim_{\mathbb{F}_p}$,

$\mathcal{C}\ell$: the ideal class group of $\mathbb{Z}[\sqrt{-p}]$.

Proposition 1

$\mathcal{C}\ell$ acts freely and transitively on $\mathcal{E}\ell\ell$ via isogenies.

$$\begin{array}{ccc} \mathcal{C}\ell \times \mathcal{E}\ell\ell & \to & \mathcal{E}\ell\ell \\ \Uparrow & & \Uparrow \\ (\mathfrak{a}, E) & \mapsto & \mathfrak{a} * E \end{array}$$

- $(\mathfrak{a}, E) \mapsto \mathfrak{a} * E$ can be easily computed.
- $(E, \mathfrak{a} * E) \mapsto \mathfrak{a}$ is hard to compute.

CSIDH (2/3)

Rough sketch of CSIDH:

$$\begin{array}{ccc} E & \xrightarrow{\mathfrak{b}} & \mathfrak{b} * E \\ {\scriptstyle \mathfrak{a}}\downarrow & & \downarrow{\scriptstyle \mathfrak{a}} \\ \mathfrak{a} * E & \xrightarrow[\mathfrak{b}]{} & \mathfrak{a}\mathfrak{b} * E \end{array}$$

$E \in \mathcal{E}\ell\ell$: public elliptic curve
$\mathfrak{a} \in \mathcal{C}\ell$: Alice's secret key, $\mathfrak{a} * E$: Alice's public key
$\mathfrak{b} \in \mathcal{C}\ell$: Bob's secret key, $\mathfrak{b} * E$: Bob's public key
$\mathfrak{a}\mathfrak{b} * E$: shared key

CSIDH (3/3)

- CSIDH uses a prime p of form $4\ell_1 \cdots \ell_n - 1$, where $\ell_1, \ldots, \ell_n$ are distinct odd primes.
- In $\mathbb{Z}[\sqrt{-p}]$, a prime ℓ_i splits as $\ell_i = \mathfrak{l}_i \bar{\mathfrak{l}}_i, \quad \mathfrak{l}_i = (\ell_i, \pi - 1),\ \bar{\mathfrak{l}}_i = (\ell_i, \pi + 1)$, where $\pi = \sqrt{-p}$.
- To calculate the action of $\mathfrak{l}_i$ (resp. $\bar{\mathfrak{l}}_i$), one needs a point in $E[\pi - 1]$ (resp. $E[\pi + 1]$) of order ℓ_i.

CSIDH (3/3)

- CSIDH uses a prime p of form $4\ell_1 \cdots \ell_n - 1$, where $\ell_1, \ldots, \ell_n$ are distinct odd primes.
- In $\mathbb{Z}[\sqrt{-p}]$, a prime ℓ_i splits as $\ell_i = \mathfrak{l}_i \bar{\mathfrak{l}}_i, \quad \mathfrak{l}_i = (\ell_i, \pi - 1),\ \bar{\mathfrak{l}}_i = (\ell_i, \pi + 1)$, where $\pi = \sqrt{-p}$.
- To calculate the action of $\mathfrak{l}_i$ (resp. $\bar{\mathfrak{l}}_i$), one needs a point in $E[\pi - 1]$ (resp. $E[\pi + 1]$) of order ℓ_i.

The actions of $\mathfrak{l}_i$ and $\bar{\mathfrak{l}}_i$ can be computed efficiently.
$\Rightarrow$ CSIDH uses ideal of form $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$,
where, $e_1, \ldots, e_n$ are integers in $[-m, m]$.

Secret keys in CSIDH are expressed as $(e_1, \ldots, e_n)$.

Algorithm of CSIDH

Input: $E \in \mathcal{E}\ell\ell$, an integer vector $(e_1, \ldots, e_n)$.
Output: $(\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}) * E$.

while $e_i \neq 0$:
  Sample a random $x_0 \in \mathbb{F}_p$ and set $P \leftarrow (x_0, y_0) \in E$.
  if $P \in E(\mathbb{F}_p)$ **then** $s \leftarrow +1$ **else** $s \leftarrow -1$.
  $S \leftarrow \{i \mid e_i$ and s have the same sign.$\}$, $k \leftarrow \prod_{i \in S} \ell_i$.
  $Q \leftarrow [(p+1)/k]P$.
  for $i \in S$:
    $R \leftarrow [k/\ell_i]Q$.
    if $R \neq \infty$ **then**
      Compute $\varphi : E \to \mathfrak{l}_i^s * E$ by using R.
      $E \leftarrow \mathfrak{l}_i^s * E,\ Q \leftarrow \varphi(Q),\ e_i \leftarrow e_i - s$.
return E.

Algorithm of CSIDH

Input: $E \in \mathcal{E}\ell\ell$, an integer vector $(e_1, \ldots, e_n)$.
Output: $(\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}) * E$.

while $e_i \neq 0$:
    Sample a random $x_0 \in \mathbb{F}_p$ and set $P \leftarrow (x_0, y_0) \in E$.
    if $P \in E(\mathbb{F}_p)$ **then** $s \leftarrow +1$ **else** $s \leftarrow -1$.
    $S \leftarrow \{i \mid e_i$ and s have the same sign.$\}$, $k \leftarrow \prod_{i \in S} \ell_i$.
    $Q \leftarrow [(p+1)/k]P$.   // k-torsion
    for $i \in S$:
        $R \leftarrow [k/\ell_i]Q$.   // $\mathfrak{l}_i^s$-torsion
        if $R \neq \infty$ **then**
            Compute $\varphi : E \to \mathfrak{l}_i^s * E$ by using R. // Isogeny (curve)
            $E \leftarrow \mathfrak{l}_i^s * E$, $Q \leftarrow \varphi(Q)$, $e_i \leftarrow e_i - s$. // Isogeny (point)
return E.   **Not constant-time!**

Constant-time

Constant-time algorithm

No branch depending on secret information.

Constant-time

Constant-time algorithm

No branch depending on secret information.

Meyer, Campos and Reith proposed a contant-time algorithm of CSIDH at PQCrypto 2019.

Constant-time algorithm by Meyer et al.

Meyer el al.

- use dummy isogenies,
- change secret key intervals. $[-m, m] \to [0, 2m]$

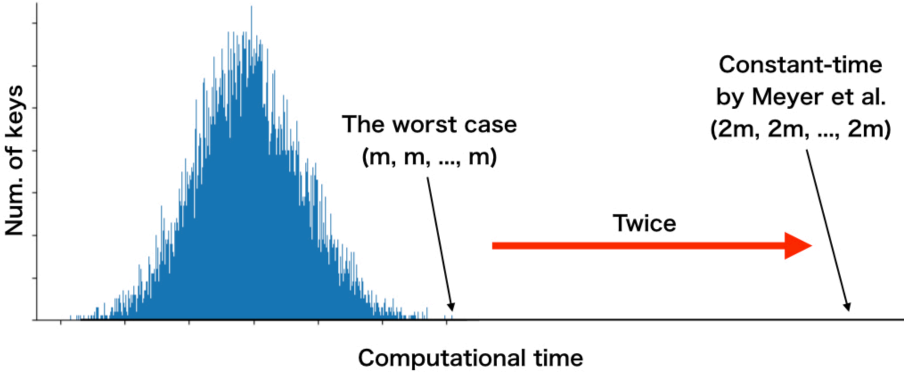
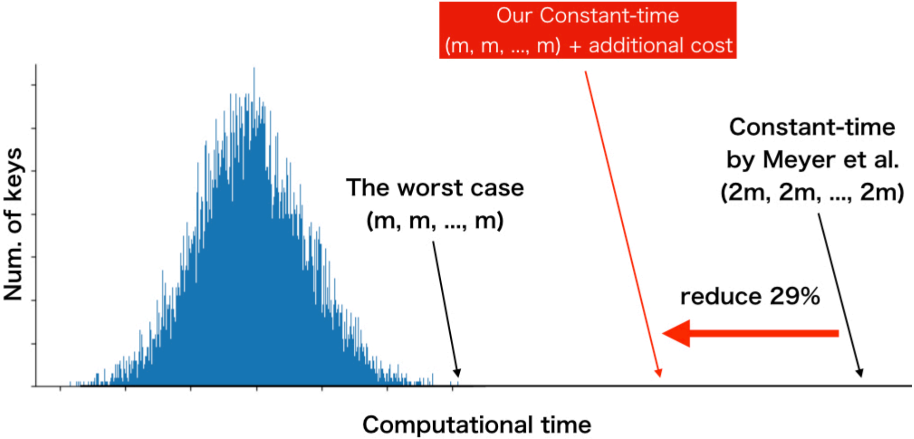


Table of contents

Our contribution

- constant-time algorithm using the interval $[-m, m]$
- keeping two points $P \in E[\pi - 1]$ and $P' \in E[\pi + 1]$
- less cost than Meyer et al.



Algorithm of CSIDH (Redisplay)

Input: $E \in \mathcal{E}\ell\ell$, an integer vector $(e_1, \ldots, e_n)$.
Output: $(\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}) * E$.

```
while e_i ≠ 0 :
    Sample a random x_0 ∈ F_p and set P ← (x_0, y_0) ∈ E.
    if P ∈ E(F_p) then s ← +1 else s ← −1.
    S ← {i | e_i and s have the same sign.}, k ← ∏_{i∈S} ℓ_i.
    Q ← [(p + 1)/k]P.        // k-torsion
    for i ∈ S :
        R ← [k/ℓ_i]Q.        // l_i^s-torsion
        if R ≠ ∞ then
            Compute φ : E → l_i^s * E by using R.  // Isogeny (curve)
            E ← l_i^s * E, Q ← φ(Q), e_i ← e_i − s.  // Isogeny (point)
return E.
```

Comparison

	Meyer et al.	Ours
Initial Point(s)	one point	**two points**
k-torsion	twice as the worst	twice as the worst
$\mathfrak{l}_i^s$-torsion	twice as the worst	**the same as the worst**
Isogeny (curve)	twice as the worst	**the same as the worst**
Isogeny (point)	twice as the worst	twice as the worst

Experimental results

C implementation of CSIDH-512 on an Intel Xeon Gold 6130 Skylake

	Clock cycles $\times 10^6$	Wall clock time
Implementation by Meyer et al.	215.3	102.742ms
Our implementation	**152.8**	**72.913ms**

Our implementation has **29.03%** fewer clock cycles than the implementation by Meyer et al.

Table of contents

Summary

- We constructed an efficient constant-time algorithm of CSIDH.
- Our algorithm uses the same secret key interval as the variable-time algorithm by keeping two points.
- Our algorithm is **29%** faster than the previous work.