A Constant-Time Algorithm of CSIDH Keeping Two Points*

Hiroshi ONUKI^{†a)}, Yusuke AIKAWA^{††b)}, Tsutomu YAMAZAKI^{†††c)}, *Nonmembers,*
and Tsuyoshi TAKAGI^{†d)}, *Member*

SUMMARY At ASIACRYPT 2018, Castryck, Lange, Martindale, Panny and Renes proposed CSIDH, which is a key-exchange protocol based on isogenies between elliptic curves, and a candidate for post-quantum cryptography. However, the implementation by Castryck et al. is not constant-time. Specifically, a part of the secret key could be recovered by the side-channel attacks. Recently, Meyer, Campos, and Reith proposed a constant-time implementation of CSIDH by introducing dummy isogenies and taking secret exponents only from intervals of non-negative integers. Their non-negative intervals make the calculation cost of their implementation of CSIDH twice that of the worst case of the standard (variable-time) implementation of CSIDH. In this paper, we propose a more efficient constant-time algorithm that takes secret exponents from intervals symmetric with respect to the zero. For using these intervals, we need to keep two torsion points on an elliptic curve and calculation for these points. We evaluate the costs of our implementation and that of Meyer et al. in terms of the number of operations on a finite prime field. Our evaluation shows that our constant-time implementation of CSIDH reduces the calculation cost by 28% compared with the implementation by Meyer et al. We also implemented our algorithm by extending the implementation in C of Meyer et al. (originally from Castryck et al.). Then our implementation achieved 152 million clock cycles, which is about 29% faster than that of Meyer et al. and confirms the above reduction ratio in our cost evaluation.

key words: CSIDH, post-quantum cryptography, isogeny-based cryptography, constant-time implementation, supersingular elliptic curve isogenies

1. Introduction

RSA and elliptic curve cryptosystems will no longer be secure once a large-scale quantum computer is built. Due to this, the importance of post-quantum cryptography (PQC) has increased. In 2017, the National Institute of Standards and Technology (NIST) started the process of PQC standardization [2]. Candidates for the NIST PQC standardization include supersingular isogeny key encapsulation (SIKE) [3], which is a scheme based on isogenies between elliptic curves. SIKE is a variant of supersingular isogeny Diffie-Hellman (SIDH), which was proposed by Jao and De

Feo [4] in 2011. SIDH uses isogenies between supersingular elliptic curves over a finite field. SIDH achieves an efficient key-exchange but needs to send torsion points of an elliptic curve as supplementary information. Attacks using this information are discussed in by Galbraith, Petit, Shani, and Ti [5] and Petit [6].

Isogeny-based cryptography was first proposed by Couveignes [7] in 1997 and independently rediscovered by Rostovtsev and Stolbunov [8], [9]. Their proposed scheme is a Diffie-Hellman-style key-exchange based on isogenies between ordinary elliptic curves over a finite field and typically called CRS. CRS does not need to send any point of elliptic curves, therefore the attacks to SIDH, which is based on information of points of elliptic curves, cannot be applied to CRS. However, even after optimizations by De Feo, Kieffer, and Smith [10], CRS is much slower than SIDH. We recommend De Feo [11] and Galbraith and Vercauteren [12] as nice introductions to isogeny-based cryptography. In 2018, Castryck, Lange, Martindale, Panny, and Renes [13] proposed commutative SIDH (CSIDH), which adopts supersingular elliptic curves to the CRS scheme. They used supersingular elliptic curves over a finite prime field $\mathbb{F}_p$ and their endomorphism rings over $\mathbb{F}_p$. Since the number of $\mathbb{F}_p$ -rational points on a supersingular elliptic curve E over $\mathbb{F}_p$ is $p + 1$, one can choose p such that $\#E(\mathbb{F}_p)$ has many small prime factors. This allows CSIDH to compute isogenies faster than CRS. Furthermore, a signature scheme using CSIDH was proposed by De Feo and Galbraith [14] and its speedup was studied by Decru, Panny, and Vercauteren [15]. Recently, a more efficient CSIDH-based signature scheme was constructed by Beullens et al. [16], though it performs only for specified parameter CSIDH-512.

However, the computational time in the proof-of-concept implementation by Castryck et al. depends on the associated secret key, so their implementation of CSIDH is not side-channel resistant. Recently, Meyer, Campos, and Reith [17] proposed a constant-time implementation of CSIDH and several speedup techniques for their implementation. They achieved the constant-time implementation by using dummy isogenies and by changing intervals of key elements from $[-m, m]$ to $[0, 2m]$, where $m \in \mathbb{N}$. Consequently, their constant-time implementation needs to calculate each degree isogeny $2m$ times, while the worst case of the variable-time CSIDH needs only m times. Therefore, the computational cost of their constant-time implementation is twice as that of the worst case of the variable-time

Manuscript received October 20, 2019.

Manuscript revised February 17, 2020.

[†]The authors are with the Department of Mathematical Informatics, The University of Tokyo, Tokyo, 113-8656 Japan.

^{††}The author is with Information Technology R&D Center, the Mitsubishi Electric Corporation, Kamakura-shi, 247-8501 Japan.

^{†††}The author is with the Graduate School of Mathematics, Kyushu University, Fukuoka-shi, 819-0395 Japan.

*This paper is a full version of [1] presented at IWSEC 2019.

a) E-mail: onuki@mist.i.u-tokyo.ac.jp

b) E-mail: Aikawa.Yusuke@bc.MitsubishiElectric.co.jp

c) E-mail: yamazaki.tsutomu.890@s.kyushu-u.ac.jp

d) E-mail: takagi@mist.i.u-tokyo.ac.jp

DOI: 10.1587/transfun.2019DMP0008

CSIDH. The constant-time implementation in [17] allows variance of the computational time of their implementation with randomness that does not relate to secret information. On the other hand, implementations which do not allow such variance are proposed by Bernstein, Lange, Martindale, and Panny [18] and Jalali, Azarderakhsh, Kermani, and Jao [19]. The implementation in [18] is for evaluating the performance of quantum attacks for CSIDH. It must not have branches in order to compute in superposition on quantum computers. The implementation in [19] is for classical computers, but it has no branches. As a result, it is slower than the implementation in [17]. We discuss the differences in these implementations in Sect. 3.2.

Our contribution.

In this paper, we propose a new constant-time implementation, which is faster than the constant-time implementation by Meyer et al. [17]. Our implementation is “constant-time” in the same sense as that of [17]. In other words, the computational time and the order of scalar multiplications and isogenies in our implementation do not depend on a secret key. We use the dummy isogenies proposed by [17], but do not change the key intervals of CSIDH, i.e., we use the interval $[-m, m]$. To achieve a constant-time implementation without changing the key intervals, we need to keep two torsion points of both $E[\pi - 1]$ and $E[\pi + 1]$ and calculation associated with these points, where π is the Frobenius endomorphism of an elliptic curve E . As a result, our implementation needs almost twice as many scalar multiplications on elliptic curves and twice as many calculations of images of points under isogenies as the worst case of the variable-time CSIDH. However, the number of calculations of the images of curves is the same as in the worst case of the variable-time CSIDH, and scalars in a part of additional scalar multiplications on elliptic curves are smaller. Therefore, our implementation is faster than the implementation in [17]. Furthermore, we propose a cost model of CSIDH that evaluates the cost by counting the number of operations on $\mathbb{F}_p$. On the basis of this cost model, we propose a parameter set of the speedup techniques of [17] for our implementation. Our cost model shows that after this speedup, our implementation reduces the cost by 28% compared with the implementation in [17]. We implemented our algorithm in C and compared its cycle count and running time with those of the implementation in [17]. Our experiment shows that the cycle count of our implementation is 29% less than that of the implementation in [17]. This confirms our cost model.

Related work.

Recently, Cervantes-Vázquez, Chenu, Chi-Domínguez, De Feo, Rodríguez-Henríquez, and Smith [20] proposed a new constant-time CSIDH resistant against a fault injection attack, which uses our method that keeps two torsion points. Furthermore, they proposed speedup techniques for CSIDH and reported that the cost of our algorithm can be reduced by about 12% by using their speedup techniques.

Organization.

The rest of this paper is organized as follows. The following section describes CSIDH. Section 3 explains a constant-time implementation and some speedup techniques for it of [17], and briefly introduces constant-time implementations based on another definition. We give the details of our new constant-time implementation of CSIDH in Sect. 4. In Sect. 5, we propose a cost model for CSIDH, evaluate the costs of several implementations of CSIDH by this model, present experimental results. We conclude our work in Sect. 6.

2. CSIDH

In this section, we overview the protocol of CSIDH and its mathematical backgrounds. For more details, see Castryck et al. [13].

2.1 Protocol of CSIDH

For describing the protocol of CSIDH, we define the following notations. Let p be a prime number, $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ the ideal class group of $\mathbb{Z}[\sqrt{-p}]$ and $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ a set of $\mathbb{F}_p$ -isomorphism classes of supersingular elliptic curves whose endomorphism ring is isomorphic to $\mathbb{Z}[\sqrt{-p}]$. Then we can define an action

$$\begin{aligned} \text{cl}(\mathbb{Z}[\sqrt{-p}]) \times \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}]) &\rightarrow \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}]) \\ (\alpha, E) &\mapsto \alpha * E. \end{aligned}$$

We call this action the class group action. The details of these notations and the action are described in the next subsection. CSIDH is a Diffie-Hellman style key exchange as follows:

Alice and Bob share an elliptic curve $E_0 \in \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ as a public parameter. Alice chooses an ideal $\alpha \in \text{cl}(\mathbb{Z}[\sqrt{-p}])$ as her secret key and sends the curve $\alpha * E_0$ to Bob as her public key. Bob proceeds in the same way by choosing a secret key $\beta \in \text{cl}(\mathbb{Z}[\sqrt{-p}])$. Then, both parties can compute the shared secret $\alpha\beta * E_0 = \beta\alpha * E_0$. Note that $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ is commutative.

2.2 Supersingular Elliptic Curves Over $\mathbb{F}_p$

Let p be a large prime of the form $4\ell_1 \cdots \ell_n - 1$, where $\ell_1, \dots, \ell_n$ are small distinct odd primes. For a supersingular elliptic curve E defined over $\mathbb{F}_p$, the p -th power Frobenius endomorphism π satisfies a characteristic equation

$$\pi^2 + p = 0,$$

and the $\mathbb{F}_p$ -endomorphism ring of E is isomorphic to an order $\mathbb{Z}[\sqrt{-p}]$ or $\mathbb{Z}[\frac{1+\sqrt{-p}}{2}]$ (see [21] for details). By the characteristic equation, π corresponds to $\sqrt{-p}$ or $-\sqrt{-p}$ in the order. We use the same symbol for an element of the order and a $\mathbb{F}_p$ -endomorphism.

The set $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ is not an empty set, and for all

classes $\mathcal{E}$ in this set, there exists one and only one $A \in \mathbb{F}_p$ such that the curve $E_A : y^2 = x^3 + Ax^2 + x$ belongs to the class $\mathcal{E}$ ([13, Theorem 8]). In other words, a class in $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ contains a unique Montgomery curve.

The ideal class group $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ of $\mathbb{Z}[\sqrt{-p}]$ acts on $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ in the following way. For simplicity, we use the same symbol for a $\mathbb{F}_p$ -isomorphism class and its representative curve and for an ideal class and its representative ideal. Furthermore, we always take an integral ideal as a representative of an ideal class. For $E \in \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ and $\mathfrak{a} \in \text{cl}(\mathbb{Z}[\sqrt{-p}])$, there are an elliptic curve $E' \in \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ and an isogeny $\varphi : E \rightarrow E'$, with the kernel $E[\mathfrak{a}] = \{P \in E \mid [\alpha]P = \infty, \forall \alpha \in \mathfrak{a}\}$. The isogeny φ and its codomain E' are unique up to $\mathbb{F}_p$ -isomorphism. The map $(\alpha, E) \mapsto E'$ does not depend on the choices of the representatives E and α or of the isogeny φ . This map defines an action of $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ on $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$. We denote the curve E' described above by $\alpha * E$. The action $(\alpha, E) \mapsto \alpha * E$ is free and transitive ([13, Theorem 7]). The Brauer-Siegel theorem [22] claims that

$$\log(\#\text{cl}(\mathbb{Z}[\sqrt{-p}])) \approx \log(\sqrt{p}).$$

Therefore, one can assume that the cardinality of $\#\text{cl}(\mathbb{Z}[\sqrt{-p}])$ is approximately $\sqrt{p}$ for a large p . This is the size of the key space of CSIDH.

To compute the action of an ideal $\mathfrak{a} \in \text{cl}(\mathbb{Z}[\sqrt{-p}])$ on an elliptic curve $E \in \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$, we express $\mathfrak{a}$ by a product of some small prime ideals whose action can be computed efficiently.

Since the prime p is of the form $4 \prod_i \ell_i - 1$ and the elliptic curve E is supersingular, the primes ℓ_i split in $\mathbb{Z}[\sqrt{-p}]$ as $(\ell_i) = \mathfrak{l}_i \bar{\mathfrak{l}}_i$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$ and $\bar{\mathfrak{l}}_i = (\ell_i, \pi + 1)$. For $E \in \mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$, the torsion subgroups of these ideals can be written as

$$\begin{aligned} E[\mathfrak{l}_i] &= E[\ell_i] \cap E[\pi - 1] = E[\ell_i] \cap E(\mathbb{F}_p), \\ E[\bar{\mathfrak{l}}_i] &= E[\ell_i] \cap E[\pi + 1] = E[\ell_i] \cap \{Q \in E \mid \pi(Q) = -Q\}. \end{aligned}$$

The second equation means that $E[\bar{\mathfrak{l}}_i] \not\subset E(\mathbb{F}_p)$ but $E[\bar{\mathfrak{l}}_i] \subset E(\mathbb{F}_{p^2})$, and if E is a Montgomery curve, the x -coordinate of a point of $E[\bar{\mathfrak{l}}_i]$ is in $\mathbb{F}_p$. Since x -coordinate only formulae for scalar multiplications [23] and odd degree isogenies [24], [25] are known, the actions of $\mathfrak{l}_i$ and $\bar{\mathfrak{l}}_i$ can be computed efficiently. In the ideal class group, $\bar{\mathfrak{l}}_i$ is the inverse of $\mathfrak{l}_i$, so we can compute the action of an ideal of the form $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$, $e_1, \dots, e_n \in \mathbb{Z}$ by the composition of the actions of $\mathfrak{l}_i$ and $\bar{\mathfrak{l}}_i$. Castryck et al. [13] showed that under some heuristics, $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$, $-m \leq e_i \leq m$ represent uniformly “almost” all the ideal classes in $\text{cl}(\mathbb{Z}[\sqrt{-p}])$, where $m \in \mathbb{N}$ such that $(2m + 1)^n \geq \#\text{cl}(\mathbb{Z}[\sqrt{-p}])$. Now, it is known that the map $[-m, m]^n \rightarrow \text{cl}(\mathbb{Z}[\sqrt{-p}])$ is not exactly uniform. Beulens, Kleinjung, and Vercauteren [16] computed the structure of $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ for 511 bit prime p . Their result shows that $(2m + 1)^n < \#\text{cl}(\mathbb{Z}[\sqrt{-p}])$ for this prime and m proposed in [13]. Therefore, the map $[-m, m]^n \rightarrow \text{cl}(\mathbb{Z}[\sqrt{-p}])$ is not surjective in CSIDH-512. Furthermore, for all sufficiently large p , it is shown that the map is not injective by Castryck, Panny, and Vercauteren [26] and Onuki and Takagi

Algorithm 1 Evaluating the class group action in CSIDH

Input: $A \in \mathbb{F}_p$, $m \in \mathbb{N}$, a list of integers $(e_1, \dots, e_n)$ s.t. $-m \leq e_i \leq m$ for $i = 1, \dots, n$, and distinct odd primes $\ell_1, \dots, \ell_n$ s.t. $p = 4 \prod_i \ell_i - 1$.

Output: $B \in \mathbb{F}_p$ s.t. $E_B = (\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}) * E_A$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$ for $i = 1, \dots, n$, and π is the p -th power Frobenius endomorphism of E_A .

```

1: while some  $e_i \neq 0$ :
2:   Sample a random  $x \in \mathbb{F}_p \setminus \{0\}$ .
3:   if  $x^3 + Ax^2 + x$  is a square in  $\mathbb{F}_p$ :
4:      $s \leftarrow +1$ .
5:   else
6:      $s \leftarrow -1$ .
7:   end if
8:   Let  $S = \{i \mid e_i s > 0\}$ .
9:   if  $S = \emptyset$ :
10:    Go to line 2.
11:   end if
12:   Set  $P = (x : 1)$  and  $k = \prod_{i \in S} \ell_i$ .
13:   Let  $P \leftarrow [(p + 1)/k]P$ .
14:   for  $i \in S$ :
15:      $Q \leftarrow [k/\ell_i]P$ .
16:     if  $Q \neq \infty$ :
17:       Compute an isogeny  $\varphi : E_A \rightarrow E_B$  with  $\ker \varphi = \langle Q \rangle$ .
18:       Let  $A \leftarrow B$ ,  $P \leftarrow \varphi(P)$ ,  $k \leftarrow k/\ell_i$ , and  $e_i \leftarrow e_i - s$ .
19:     end if
20:   end for
21: end while
22: return A.
```

[27]. However, any attacks using these gaps of bijectivity is not known. In this paper, we agree with the heuristics by Castryck et al. We denote the exponents (e_i) by secret exponents.

2.3 Computing the Class Group Action $\alpha * E$

As stated above, we can express a supersingular elliptic curve in $\mathcal{Ell}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ by $A \in \mathbb{F}_p$ and an ideal class in $\text{cl}(\mathbb{Z}[\sqrt{-p}])$ by secret exponents $(e_1, \dots, e_n)$ in the intervals $[-m, m]^n$. Then the class group action can be computed as described in Algorithm 1. In this algorithm, we use the XZ-only Montgomery curve arithmetic [23] in the computation for the arithmetic of elliptic curves. This allows us to compute an action of an ideal class on an elliptic curve only by operations on $\mathbb{F}_p$. Algorithm 1 consists of three main parts: (1) scalar multiplications in the outer loop (line 13), (2) scalar multiplications in the inner loop (line 15), and (3) isogenies (lines 17–18). We denote (1) by the outer SM[†] and (2) by the inner SM. The outer SM generates a k -torsion point, where k is the product of all primes whose exponents have the same sign as s . By using this point, the inner SM generates a ℓ_i -torsion point Q . If Q is not the point at infinity, Q is a generator of $E[\mathfrak{l}_i]$ if $s = 1$, or $E[\bar{\mathfrak{l}}_i]$ if $s = -1$, i.e., a generator the kernel of an isogeny that we should compute. In this case, we compute the isogeny φ with kernel $\langle Q \rangle$ and update the curve coefficient, the point P , the product of primes k and an exponent e_i . Note that we can update k to k/ℓ_i because the ℓ_i -torsion part of P is in the kernel of the isogeny φ . Updating k reduces the scalar of the next scalar multiplication in the inner loop.

[†]The acronym SM stands for “scalar multiplication.”

3. Previous Works for Constant-Time Implementation of CSIDH

In this section, we explain a constant-time implementation of CSIDH and its speedup techniques proposed by Meyer et al. [17] and briefly describe related works.

3.1 Constant-Time Implementation

As already mentioned by Castryck et al. [13], Algorithm 1 is not side-channel resistant because the computational time for a public key and a shared secret depends on the associated secret key. To solve this problem, Meyer et al. [17] proposed a constant-time implementation of CSIDH. According to [17], “a constant-time implementation” means an implementation whose computational time and order of scalar multiplications of each size and isogenies of each degree do not depend on a secret key. Their constant-time implementation is described in Algorithm 2. Note that they allowed the computational time of their implementation to vary with random choices of a point P on an elliptic curve in line 5 in Algorithm 2, which do not relate to secret information. These choices decide the conditional branch **if** $Q \neq \infty$ in line 9 and affect the computational time.

To achieve a constant-time implementation, they used dummy isogenies and changed the intervals of the integer key elements from $[-m, m]$ to $[0, 2m]$. We explain these techniques below.

Dummy isogenies.

It seems that one should compute a constant number of isogenies of each degree ℓ_i and only use the ones required by the secret key. However, to do this, one should compute additional scalar multiplications on elliptic curves in line 18 in Algorithm 1, because one needs to drop the ℓ_i -torsion part of a point P . By this scalar multiplication, one can update k to k/ℓ in line 18 in Algorithm 1. This reduces the cost of a scalar multiplications in line 15 in Algorithm 1. Meyer and Reith [25] proposed a technique that uses the kernel generation in the isogeny computation to compute the scalar multiplication $[\ell_i]P$. By using this technique, one achieves dummy isogenies with two extra differential additions on an elliptic curve. For more details, see [17], [25].

Changing the key intervals.

By using dummy isogenies, the number of isogeny computations is fixed. However, this is not sufficient to achieve a constant-time implementation, since the sizes of the scalar multiplications in lines 13 and 15 in Algorithm 1 vary in accordance with the signs of secret exponents. The sizes of the scalar multiplications vary, because the integer k in Algorithm 1 depends on the signs of secret exponents. To remove this effect, Meyer et al. [17] proposed changing the intervals from $[-m, m]$ to $[0, 2m]$.

For efficiency, Meyer et al. [17] uses Elligator [28] for

Algorithm 2 Constant-time evaluation of the class group action in CSIDH [17]

Input: $A \in \mathbb{F}_p$, $m \in \mathbb{N}$, a list of integers $(e_1, \dots, e_n)$ s.t. $0 \leq e_i \leq 2m$ for $i = 1, \dots, n$, and distinct odd primes $\ell_1, \dots, \ell_n$ s.t. $p = 4 \prod_i \ell_i - 1$.
Output: $B \in \mathbb{F}_p$ s.t. $E_B = (\ell_1^{e_1} \dots \ell_n^{e_n}) * E_A$, where $\ell_i = (\ell_i, \pi - 1)$ for $i = 1, \dots, n$, and π is the p -th power Frobenius endomorphism of E_A .

- 1: Set $e'_i = 2m - e_i$ for $i = 1, \dots, n$.
- 2: **while** some $e_i \neq 0$ or $e'_i \neq 0$:
- 3: Set $S = \{i \mid e_i \neq 0 \text{ or } e'_i \neq 0\}$.
- 4: Set $k = \prod_{i \in S} \ell_i$.
- 5: Generate a point $P \in E_A[\pi - 1]$ by Elligator.
- 6: Let $P \leftarrow [(p+1)/k]P$.
- 7: **for** $i \in S$:
- 8: Set $Q = [k/\ell_i]P$.
- 9: **if** $Q \neq \infty$: /* branch not involving secret information */
- 10: **if** $e_i \neq 0$: /* branch involving secret information */
- 11: Compute an isogeny $\varphi : E_A \rightarrow E_B$ with $\ker \varphi = \langle Q \rangle$.
- 12: Let $A \leftarrow B$, $P \leftarrow \varphi(P)$, and $e_i \leftarrow e_i - 1$.
- 13: **else**
- 14: Dummy computation.
- 15: Let $A \leftarrow A$, $P \leftarrow [\ell_i]P$, and $e'_i \leftarrow e'_i - 1$.
- 16: **end if**
- 17: **end if**
- 18: Let $k \leftarrow k/\ell_i$.
- 19: **end for**
- 20: **end while**
- 21: **return** A .

CSIDH, which was proposed by Bernstein, Lange, Martinale, and Panny [18]. Elligator enables us to generate x -coordinates of points in $E[\pi - 1]$ and $E[\pi + 1]$ simultaneously by computing only one Legendre symbol. Note that we cannot control the orders of points generated by Elligator. The orders are randomly determined and the orders of two points generated simultaneously could be different. For the details, see [17], [18]. Typically, the cost of Elligator is negligible. Because this consists of the cost of Legendre symbol ($O(\log(p))$ multiplications on $\mathbb{F}_p$) and few operations on $\mathbb{F}_p$, and the number of calling Elligator is the range of the interval $2m$ plus a few times (this depends on randomness in Algorithm 2). In CSIDH-512, which is a parameter set proposed by [13], the cost of Elligators in Algorithm 2 is less than one thousand multiplications on $\mathbb{F}_p$, on the other hand, the total cost of this algorithm is more than one million multiplications on $\mathbb{F}_p$. See Table 2 in Sect. 5.1 for the total cost.

Remark 1: Cervantes-Vázquez et al. [20] pointed out that the implementation of Elligator in [17] is not correct in the sense of constant-time, since it does not use randomness. However, this can be easily fixed. See Sect. 3 in [20] for the details.

3.2 Constant-Time Implementations Based on Another Definition

As we stated above, Meyer et al. [17] allow variance of the computational time of their implementation with randomness that does not relate to secret information (caused by the branch **if** $Q \neq \infty$ in line 9 in Algorithm 2). On the other hand, constant-time implementations that do not

allow this variance are known. Bernstein et al. [18] constructed a constant-time implementation of CSIDH for evaluating the performance of quantum attacks. For calculating the class group actions in superposition on a quantum computer, a completely constant-time implementation is required. Therefore, their constant-time implementation has no branches (such as **if** branch). Jalali, Azarderakhsh, Kermani, and Jao [19] proposed a constant-time implementation for classical computers, which also has no branches. As a result of removing all branches, these implementations are slower than that of [17]. We propose a constant-time implementation based on the definition in [17], i.e., our implementation allows branches which do not depend on secret information.

3.3 Speedup Techniques for CSIDH

Meyer et al. [17] proposed several techniques to speedup their constant-time implementation of CSIDH. These can be also applied to our algorithm. We briefly explain two of them here.

SIMBA.

SIMBA (Splitting isogeny computations into multiple batches) splits the set S in Algorithm 2 into small sets. This decreases the value of $k = \prod_{i \in S} \ell_i$. Therefore, this reduces the cost of a scalar multiplication in line 8 in Algorithm 2, while this increases the cost of a scalar multiplication in line 6 in Algorithm 2. The number of the latter scalar multiplications in one execution of the algorithm is much smaller than that of the former, so SIMBA reduces the total cost of the algorithm. Furthermore, Meyer et al. [17] proposed merging the splitting sets after a certain number of steps of the while loop in Algorithm 1. This is because after more than $2m$ steps of the loop, SIMBA could backfire, see [17] for details. The same as [17], we denote the technique that splits S into ν small sets and merges after μ steps by SIMBA- ν - μ .

Sampling secret exponents from different intervals.

Instead of sampling all secret exponents from the same interval $[-m, m]$ (or $[0, 2m]$ for the implementation by [17].), one can choose the key elements from different intervals for each isogeny degree. This means that one changes the set of the secret keys to

$$\{(e_1, \dots, e_n) \in \mathbb{Z}^n \mid -m_i \leq e_i \leq m_i, \text{ for } i = 1, \dots, n\},$$

where $m_i \in \mathbb{N}$ are new bound for e_i . One can reduce the cost of computing the isogenies by using smaller m_i for high degree isogenies and larger m_i for low degree isogenies. The same technique for CRS using ordinary curves is proposed by De Feo, Kieffer, and Smith [10]. We call this technique weighted secret exponents.

4. Our Constant-Time Implementation

In this section, we propose a new constant-time implementation that is faster than that of [17].

The constant-time implementation in [17] requires the cost to be the same as that of calculating the action of the ideal class corresponding to secret exponents $(2m, \dots, 2m)$. This cost is twice the cost corresponding to secret exponents $(m, \dots, m)$, which is the worst case in the variable-time CSIDH. We mitigate the cost for achieving constant-time by using positive and negative secret exponents.

4.1 Basic Idea

To achieve a constant-time implementation without fixing the signs of secret exponents, we compute isogenies corresponding to positive and negative secret exponents in the same round in the while loop in Algorithm 1. This requires keeping two points of both $E[\pi - 1]$ and $E[\pi + 1]$ and computing scalar multiplications and images under isogenies for both points. This means that our new method needs almost twice as many scalar multiplications and twice as many computations of images of points per isogeny calculation (the reason we need “almost” twice as many scalar multiplications is explained later). However, it needs only one computation for an isogenous curve coefficient. Therefore, the cost of our method is less than twice of the worst case of the variable-time CSIDH. Combining this method and dummy isogenies of [17], [25], we achieve a more efficient constant-time implementation.

4.2 Proposed Algorithm

Our constant-time implementation for computing the class group action is described in Algorithm 3.

In Algorithm 3, the points P_0 and P_1 are k -torsion of $E[\pi - 1]$ and $E[\pi + 1]$, respectively. The indicator s is the sign bit of a secret exponent e_i (line 8), i.e., $s = 0$ if $e_i \geq 0$ and $s = 1$ if $e_i < 0$. This can be computed by bit operations. For example, $s = e_i \gg 7$ if e_i is stored as a signed 8-bit integer. The point Q is ℓ_i -torsion of $E[\pi - 1]$ if $e_i \geq 0$ or of $E[\pi + 1]$ is $e_i < 0$ (line 9). Therefore, the algorithm computes the isogeny corresponding to the sign of e_i in line 13–17. Note that we need a scalar multiplication on P_{1-s} by ℓ_i in line 10 because the ℓ_i -torsion parts of P_0 and P_1 should drop in order to update k to k/ℓ_i . The ℓ_i -torsion part of P_s is Q and drops by the isogeny φ , since Q is in the kernel of φ . In contrast, the ℓ_i -torsion part of P_{1-s} does not drop by φ . We also note that we need to calculate this scalar multiplication even when $Q = \infty$, i.e., one fails to obtain a generator of the kernel of an isogeny. The equation $Q = \infty$ means the ℓ_i -torsion part of P_s has already vanished but does not mean the ℓ_i -torsion part of P_{1-s} has vanished. Therefore, for updating k to k/ℓ_i , we need the scalar multiplication on P_{1-s} by ℓ_i . In contrast, in the variable-time CSIDH algorithm, one calculates nothing when $Q = \infty$. This is why we said “we need “almost” twice as many scalar multiplications” in the previous subsection. However, the number of these additional scalar multiplications is much smaller than the total number of scalar multiplications. For example, it is about 2% of the total number of scalar multiplications in CSIDH-512, which

Algorithm 3 Our constant-time evaluation of the class group action in CSIDH

Input: $A \in \mathbb{F}_p$, $m \in \mathbb{N}$, a list of integers $(e_1, \dots, e_n)$ s.t. $-m \leq e_i \leq m$ for $i = 1, \dots, n$, and distinct odd primes $\ell_1, \dots, \ell_n$ s.t. $p = 4 \prod_i \ell_i - 1$.

Output: $B \in \mathbb{F}_p$ s.t. $E_B = (\ell_1^{e_1} \dots \ell_n^{e_n}) * E_A$, where $\ell_i = (\ell_i, \pi - 1)$ for $i = 1, \dots, n$, and π is the p -th power Frobenius endomorphism of E_A .

- 1: Set $e'_i = m - |e_i|$ for $i = 1, \dots, n$.
- 2: **while** some $e_i \neq 0$ or $e'_i \neq 0$:
- 3: Set $S = \{i \mid e_i \neq 0 \text{ or } e'_i \neq 0\}$.
- 4: Set $k = \prod_{i \in S} \ell_i$.
- 5: Generate points $P_0 \in E_A[\pi - 1]$ and $P_1 \in E_A[\pi + 1]$ by Elligator.
- 6: Let $P_0 \leftarrow [(p + 1)/k]P_0$ and $P_1 \leftarrow [(p + 1)/k]P_1$.
- 7: **for** $i \in S$:
- 8: Set s the sign bit of e_i .
- 9: Set $Q = [k/\ell_i]P_s$.
- 10: Let $P_{1-s} \leftarrow [\ell_i]P_{1-s}$.
- 11: **if** $Q \neq \infty$: /* branch not involving secret information */
- 12: **if** $e_i \neq 0$: /* branch involving secret information */
- 13: Compute an isogeny $\varphi : E_A \rightarrow E_B$ with $\ker \varphi = \langle Q \rangle$.
- 14: Let $A \leftarrow B$, $P_0 \leftarrow \varphi(P_0)$, $P_1 \leftarrow \varphi(P_1)$,
and $e_i \leftarrow e_i - 1 + 2s$.
- 15: **else**
- 16: Dummy computation.
- 17: Let $A \leftarrow A$, $P_s \leftarrow [\ell_i]P_s$, and $e'_i \leftarrow e'_i - 1$.
- 18: **end if**
- 19: **end if**
- 20: Let $k \leftarrow k/\ell_i$.
- 21: **end for**
- 22: **end while**
- 23: **return** A .

is the parameter set for CSIDH proposed by Castryck et al. [13].

Remark 2: The same as in the implementation in [17], we use Elligator for CSIDH. It enables us to generate x -coordinates of P_0 and P_1 in line 5 in Algorithm 3 by computing only one Legendre symbol.

Remark 3: Our dummy isogeny includes a dummy calculation corresponding to evaluations of P_1 under φ not only of P_0 so that the calculation costs of lines 13–14 and lines 16–17 in Algorithm 3 are the same.

Remark 4: One reviewer suggested that we should consider side-channel attacks detecting calculations on the point at infinity, like as in [29]. Fortunately, our algorithm is resistant against attacks of this type, since the timing at which calculations on the point at infinity occur depends only on the randomness of generating point in line 5 in Algorithm 3 and not on secret information. Note that the only point that could be the point at infinity in the isogeny computation (line 13–14) and the dummy computation (line 16–17) is P_{1-s} , and the evaluation of $\varphi(P_{1-s})$, which is the only calculation involving P_{1-s} , exists in both the isogeny computation and the dummy computation.

4.3 Security Comparison with the Implementation by Meyer et al.

We claim that the security of our implementation against

side-channel attacks is equivalent to that of the implementation in [17]. Although Algorithm 3 contains a conditional branch on secret information, one can replace the branch by conditional swaps and implement it without conditional branches and memory accesses which depend on secret information.

Meyer et al. [17] claimed that their implementation is constant-time in the sense that it can prevent the two leakage scenarios they consider [17, §3]: timing leakage and simple power analysis (SPA)[†]. Timing leakage is leaking information on a secret key by the computational time. SPA measures the power consumption of the algorithm and determines blocks that represent the two main primitives in CSIDH, scalar multiplications, and isogeny computation. Their implementation prevents these leakage scenarios because the computational time and the order of scalar multiplications of each size and isogenies of each degree in their implementation do not depend on a secret key.

Our implementation also prevents the above two leakage scenarios. Its computational time does not depend on information on a secret key because of dummy isogenies. By calculating isogenies whose exponents have different signs in the same loop, the order of scalar multiplications of each size and isogenies of each degree do not depend on information on a secret key. Furthermore, our implementation has two branches, the same as the implementation in [17]. The first is **if** $Q \neq \infty$ in line 11 in Algorithm 3, which does not involve secret information and affects the computational time (the corresponding branch in the implementation in [17] is in line 9 in Algorithm 2). The second is **if** $e_i \neq 0$, line 12 in Algorithm 3, which involves secret information and does not affect the computational time (the corresponding branch in the implementation in [17] is in line 10 in Algorithm 2). This branch can be removed by using conditional swaps and implemented securely. See the code of [17], that is available at <https://zenon.cs.hs-rm.de/pqcrypto/constant-csidh-c-implementation>. We note that our implementation switches calculation for isogenies associated to positive and negative secret exponents by the indicator s in line 8 in Algorithm 3, which can be computed by bit operations. There are memory accesses which depend on the secret bit s in line 9–10 in Algorithm 3. But one can implement it securely by using a conditional swap to swap the values of P_0 and P_1 . As a result, we conclude that our implementation is constant-time as that of [17].

5. Evaluation of Our Implementation

In this section, we discuss the computational cost of constant-time implementations of CSIDH. We focus on CSIDH-512 [13], which uses the characteristic of the definition field $p = 4 \prod_{i=1}^{74} \ell_i - 1$, where ℓ_i is the i -th odd prime number for $i = 1, \dots, 73$ and $\ell_{74} = 587$.

[†]We does not consider multi-trace attacks (e.g. differential power analysis (DPA), correlation power analysis (CPA)).

Table 1 The number of operations on $\mathbb{F}_p$ in functions for CSIDH.

Function	M	S	a
LADDER [30]	$8t - 4$	$4t - 2$	$8t - 6$
Kernel.Points [24]	$4(d - 1)$	$2(d - 1)$	$2(3d - 4)$
OddIsogeny.Points [24]	$4d$	2	$2(d + 1)$
OddIsogeny.Curve [25]	$2d + t'$	$2t' + 6$	6

5.1 Cost Model

First, we explain our cost model for CSIDH that evaluates the cost as the number of operations on $\mathbb{F}_p$. Our model computes the arithmetic of elliptic curves by a Montgomery ladder [30] and isogenies by the formula of Costello and Hisil [24] and Meyer and Reith [25]. Table 1 shows the cost of functions we use. We use LADDER for scalar multiplications on elliptic curves and Kernel.Points, OddIsogeny.Points and OddIsogeny.Curve for isogenies. OddIsogeny.Points is a function that outputs the image of a point under an isogeny and is called OddIsogeny by Costello and Hisil [24]. We use this name in order to distinguish this function from OddIsogeny.Curve. The function OddIsogeny.Curve outputs the image curve under an isogeny by using the method described by Meyer and Reith [25, §4.2]. In the table, **M**, **S**, and **a** mean the numbers of multiplications, squarings, and additions on $\mathbb{F}_p$ respectively. t is the bit size of a for computing scalar multiplication $[a]P$, and for computing an isogeny of degree ℓ , $d = (\ell - 1)/2$ and t' is the bit size of ℓ .

As we stated in Sect. 2.3, the algorithm of CSIDH consists of three main parts. In Algorithm 3, the outer SM is in line 6, the inner SM is in lines 9–10, and the isogenies are in lines 13–14 and 16–17. The number of operations on $\mathbb{F}_p$ in these parts accounts for more than 99.9% of all operations on $\mathbb{F}_p$, so we regard this cost as the total cost of the class group action in CSIDH.

5.2 Our Proposed Parameters

By using our cost model for CSIDH, we evaluated the cost of various choices of parameters for the speedup techniques we described in Sect. 3.3. The best parameters we found by heuristic experiments are SIMBA-3-8 and weighted secret exponents $-m_i \leq e_i \leq m_i$ for $i = 1, \dots, 74$, where

$$(m_i) = (5, 6, 7, 7, 7, 7, 7, 8, 8, 8, 9, 10, 10, 10, 10, 9, 9, 9, 8, \\ 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 5, 5, 5, 5, 5, \\ 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, \\ 3, 3, 3, 3, 2, 2, 2, 2, 2, 1).$$

A systematic way to determine the parameters for SIMBA and weighted secret exponents is proposed by Hutchinson, LeGrow, Koziel, and Azarderakhsh [31].

For CSIDH-512, the structure of the class group is known by [16]. However, as far as we know, there is no result to use this information for optimizing the constant-time CSIDH. We leave it as an open problem.

Table 2 The numbers of operations on $\mathbb{F}_p$ in the implementation by Meyer et al. and in our implementation (1,000 operations).

	M	S	a	$\bar{M}$
[17]	1,009	351	1,009	1,341
This work	733	244	681	962

Table 3 Performance comparison in 10,000 runs. Average, maximum, minimum and standard deviation of clock cycles, and average of wall clock time.

Implementation	Clock cycles (Mcycles)				Wall clock time
	Ave.	Max.	Min.	Std.	Ave.
[17]	216	408	207	6.3	103ms
This work	152	273	149	6.3	73ms

5.3 Comparison with the Implementation by Meyer et al.

We construct a program that counts the numbers of operations on $\mathbb{F}_p$ in the class group action of CSIDH-512. Because the costs vary with random choices of points on elliptic curves, we ran this program 10,000 times for each implementation and took the average of the numbers of each operation. We give the numbers of each operation on $\mathbb{F}_p$ in the constant-time implementations in [17] and by ourselves. We used speedup parameters proposed by [17, §6] for the implementation in [17]. In the following tables, the notation $\bar{M}$ means the cost measured in the number of multiplications, assuming **S** = 0.8**M** and **a** = 0.05**M** the same as [25]. Table 2 shows the numbers of operations on $\mathbb{F}_p$ in the implementation in [17] and in our implementation. The cost of our implementation is 962 thousand $\bar{M}$, which is about 28% less than that of the implementation in [17].

We implemented our algorithm with the speedup techniques in C. Our code is based on the code by Meyer et al. [17] (originally from Castryck et al. [13])[†].

Table 3 shows the cycle counts and running times for our implementation and that in [17]. For the implementation in [17], we used the code on which our code is based. We ran both codes on an Intel Xeon Gold 6130 Skylake processor running Ubuntu 16.04.5 LTS. Our implementation has 29% fewer clock cycles than the implementation in [17], which is almost the same as the reduction ratio expected by the evaluation of our cost model.

6. Conclusion

We improved a constant-time implementation of commutative supersingular isogeny Diffie-Hellman (CSIDH), which is isogeny-based Diffie-Hellman-style key exchange and a candidate for post-quantum cryptography. Our implementation is based on the constant-time implementation in Meyer

[†]Our code is available for download at <https://github.com/hiros-hi-onuki/constant-time-csidh> and the code by Meyer et al. is available at <https://zenon.cs.hs-rm.de/pqcrypto/constant-csidh-c-implementation>. We used the version of commit ID 7fc2abdd for the code by Meyer et al.

et al. [17]. Whereas they used only non-negative key intervals, we used key intervals symmetric with respect to zero. To achieve a constant-time implementation using these intervals, we constructed a new algorithm that keeps two torsion points on an elliptic curve. The additional cost for calculation associated with this point is less than the additional cost in [17] to achieve constant-time. Consequently, our implementation is faster than the implementation in [17].

We evaluated these costs by counting the number of operations on $\mathbb{F}_p$. This evaluation showed that our implementation reduces the cost by 28% compared with the implementation in [17]. We tested this reduction ratio by implementing our algorithm in C and measuring its clock cycles. The reduction ratio measured by clock cycles is 29%. This confirms the evaluation results by our cost model. Furthermore, we considered a representation of ideal classes that are used in CSIDH. We showed that our new parameter for the representations is at least as secure as that of the original CSIDH.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1406, Japan.

References

- [1] H. Onuki, Y. Aikawa, T. Yamazaki, and T. Takagi, "(Short paper) a faster constant-time algorithm of csidh keeping two points," *Advances in Information and Computer Security*, N. Attrapadung and T. Yagi, eds., pp.23–33, Springer International Publishing, Cham, 2019.
- [2] National Institute of Standards and Technology (NIST) "NIST Post-Quantum Cryptography Standardization," <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>
- [3] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, G. Pereira, J. Renes, V. Soukharev, and D. Urbanik, "SIKE - Supersingular isogeny key encapsulation," Submission to the NIST Post-Quantum Cryptography Standardization project; <https://sike.org>
- [4] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," *Post-Quantum Cryptography*, B.Y. Yang, ed., pp.19–34, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [5] S.D. Galbraith, C. Petit, B. Shani, and Y.B. Ti, "On the security of supersingular isogeny cryptosystems," *Advances in Cryptology – ASIACRYPT 2016*, J.H. Cheon and T. Takagi, eds., pp.63–91, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
- [6] C. Petit, "Faster algorithms for isogeny problems using torsion point images," *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, eds., pp.330–353, Springer International Publishing, Cham, 2017.
- [7] J.M. Couveignes, "Hard homogeneous spaces," *IACR Cryptology ePrint Archive 2006/291*; <https://eprint.iacr.org/2006/291>
- [8] A. Rostovtsev and A. Stolbunov, "Public-key cryptosystem based on isogenies," *IACR Cryptology ePrint Archive 2006/145*; <https://eprint.iacr.org/2006/145>
- [9] A. Stolbunov, "Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves," *Advances in Mathematics of Communications*, vol.4, no.2, pp.215–235, 2010.
- [10] L. De Feo, J. Kieffer, and B. Smith, "Towards practical key exchange from ordinary isogeny graphs," *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, eds., pp.365–394, Springer International Publishing, Cham, 2018.
- [11] J. De Feo, "Mathematics of isogeny based cryptography," *arXiv:1711.04062 [cs.CR]*, 2017.
- [12] S.D. Galbraith and F. Vercauteren, "Computational problems in supersingular elliptic curve isogenies," *Quantum Information Processing*, vol.17, no.10, p.265, 2018.
- [13] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes, "CSIDH: An efficient post-quantum commutative group action," *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, eds., pp.395–427, Springer International Publishing, Cham, 2018.
- [14] L. De Feo and S.D. Galbraith, "SeaSign: Compact isogeny signatures from class group actions," *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, eds., pp.759–789, Springer International Publishing, Cham, 2019.
- [15] T. Decru, L. Panny, and F. Vercauteren, "Faster seaisign signatures through improved rejection sampling," *Post-Quantum Cryptography*, J. Ding and R. Steinwandt, eds., pp.271–285, Springer International Publishing, Cham, 2019.
- [16] W. Beullens, T. Kleinjung, and F. Vercauteren, "CSI-FiSh: Efficient isogeny based signatures through class group computations," *Advances in Cryptology – ASIACRYPT 2019*, S.D. Galbraith and S. Moriai, eds., pp.227–247, Springer International Publishing, Cham, 2019.
- [17] M. Meyer, F. Campos, and S. Reith, "On lions and elligators: An efficient constant-time implementation of CSIDH," *Post-Quantum Cryptography*, J. Ding and R. Steinwandt, eds., pp.307–325, Springer International Publishing, Cham, 2019.
- [18] D.J. Bernstein, T. Lange, C. Martindale, and L. Panny, "Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies," *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, eds., pp.409–441, Springer International Publishing, Cham, 2019.
- [19] A. Jalali, R. Azarderakhsh, M.M. Kermani, and D. Jao, "Towards optimized and constant-time CSIDH on embedded devices," *Constructive Side-Channel Analysis and Secure Design*, I. Polian and M. Stöttinger, eds., pp.215–231, Springer International Publishing, Cham, 2019.
- [20] D. Cervantes-Vázquez, M. Chenu, J.J. Chi-Domínguez, L. De Feo, F. Rodríguez-Henríquez, and B. Smith, "Stronger and faster side-channel protections for csidh," *Progress in Cryptology – LATINCRYPT 2019*, P. Schwabe and N. Thériault, eds., pp.173–193, Springer International Publishing, Cham, 2019.
- [21] C. Delfs and S.D. Galbraith, "Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$," *Des. Codes Cryptogr.*, vol.78, no.2, pp.425–440, 2016.
- [22] C. Siegel, "Über die classenzahl quadratischer zahlkörper," *Acta Arith.*, vol.1, no.1, pp.83–86, 1935.
- [23] P.L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Math. Comput.*, vol.48, no.177, pp.243–264, 1987.
- [24] C. Costello and H. Hisil, "A simple and compact algorithm for SIDH with arbitrary degree isogenies," *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, eds., pp.303–329, Springer International Publishing, Cham, 2017.
- [25] M. Meyer and S. Reith, "A faster way to the CSIDH," *Progress in Cryptology – INDOCRYPT 2018*, D. Chakraborty and T. Iwata, eds., pp.137–152, Springer International Publishing, Cham, 2018.
- [26] W. Castryck, L. Panny, and F. Vercauteren, "Rational isogenies from irrational endomorphisms," *IACR Cryptology ePrint Archive 2019/1202*; <https://eprint.iacr.org/2019/1202>
- [27] H. Onuki and T. Takagi, "On collisions related to an ideal class of order 3 in csidh," *IACR Cryptology ePrint Archive 2019/1209*; <https://eprint.iacr.org/2019/1209>
- [28] D.J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange, "Elligator: Elliptic-curve points indistinguishable from uniform random strings," *Proc. 2013 ACM SIGSAC Conference on Computer &*

- Communications Security, pp.967–980, 2013.
- [29] D. Genkin, L. Valenta, and Y. Yarom, “May the fourth be with you: A microarchitectural side channel attack on several real-world applications of curve25519,” Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.845–858, Association for Computing Machinery New York, Oct. 2017.
 - [30] C. Costello and B. Smith, “Montgomery curves and their arithmetic,” J. Cryptogr. Eng., vol.8, no.3, pp.227–240, 2018.
 - [31] A. Hutchinson, J. LeGrow, B. Koziel, and R. Azarderakhsh, “Further optimizations of CSIDH: A systematic approach to efficient strategies, permutations, and bound vectors,” IACR Cryptology ePrint Archive 2019/1209; <https://eprint.iacr.org/2019/1121>

Tsuyoshi Takagi received the B.Sc. and M.Sc. degrees in mathematics from Nagoya University in 1993 and 1995, respectively. He was engaged in research on network security at NTT Laboratories from 1995 to 2001. He received the Ph.D. from the Technical University of Darmstadt in 2001. He was an Assistant Professor in the Department of Science at Technical University of Darmstadt until 2005. He is currently a Professor in the Graduate School of Information Science and Technology at University of Tokyo. His current research interests are information security and cryptography. He received DOCOMO Mobile Science Award in 2013, IEICE Achievement Award in 2013, and JSPS Prize in 2014. Dr. Takagi was a Program Chair of the 7th International Conference on Post-Quantum Cryptography, PQCrypto 2016.

Hiroshi Onuki received the Ph.D. degree from Tokyo Metropolitan University, Japan, in 2017. Currently, he is a research fellow at Graduate School of Information Science and Technology, the University of Tokyo, Japan. His research interests are mainly in algorithmic number theory and mathematical cryptography.

Yusuke Aikawa received the Ph.D. degree in Science from Hokkaido University, Japan, in 2019. During Oct. 2018–Mar. 2019, he studied at Graduate School of Information Science and Technology, the University of Tokyo, Japan. He is currently working as a researcher at Mitsubishi Electric Corporation. His research interests are mainly in mathematical cryptography and algorithmic number theory.

Tsutomu Yamazaki graduated from the School of Science, Kyushu University in 2017. He has received the Master of Mathematics from the Graduate School of Mathematics, Kyushu University in 2019. He is currently a teacher at Kasumigaoka Senior High School.