A Faster Constant-time Algorithm of CSIDH keeping Two Torsion Points

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

[1] Department of Mathematical Informatics, University of Tokyo, Japan
{onuki,takagi}@mist.i.u-tokyo.ac.jp
[2] Department of Mathematics, Hokkaido University, Japan
yusuke@math.sci.hokudai.ac.jp
[3] Graduate School of Mathematics, Kyushu University, Japan
yamazaki.tsutomu.890@s.kyushu-u.ac.jp

Abstract. 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 in 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 28.23% compared with the implementation by Mayer 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 172.4 million clock cycles, which is about 27.35% faster than that of Meyer et al. and confirms the above reduction ratio in our cost evaluation.

Keywords: 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 [22]. Candidates for the NIST PQC standardization includes supersingular isogeny key encapsulation (SIKE) [18], which is a cryptography based on isogenies between elliptic curves. SIKE is an implementation of supersingular isogeny Diffie-Hellman (SIDH), which was proposed by Jao and De Feo [16] 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 [14] and Petit [23].

Isogeny-based cryptography was first proposed by Couveignes [8] in 1997 and independently rediscovered by Rostovtsev and Stolbunov [24, 26]. It 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 [9] and Galbraith and Vercauteren [15] as nice introductions to isogeny-based cryptography. In 2018, Castryck, Lange, Martindale, Panny and Renes [3] 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 [11] and its speedup was studied by Decru, Panny and Vercauteren [12].

However, the computational time of a public key 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 [19] 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 valuable-time CSIDH.

In this paper, we propose a new constant-time implementation, which is faster than the constant-time implementation by Meyer et al. Our implementation is "constant-time" in the same sense as that of Meyer et al. 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 Meyer et al. 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 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 by Meyer et al. 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 by Meyer et al. for our implementation. Our cost model shows that after this speedup, our implementation reduces the cost by 28.23% compared with the implementation by Meyer et al. We implemented our algorithm in C and compared its cycle count and running time with those of the implementation by Meyer et al. Our experiment shows that the cycle count of our implementation is 27.35% less than that of the implementation by Meyer et al. This confirms our cost model.

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 by Meyer et al. and briefly introduces constant-time implementations based on anther definition. We give the details of our new constant-time implementation of CSIDH in Section 4. In Section 5, we propose a cost model for CSIDH, evaluate the costs of several implementations of CSIDH by this model, present experimental results, and discuss security of a speedup technique for CSIDH. We conclude our work in Section 6.

2 CSIDH

In this section, we overview the protocol of CSIDH and its mathematical backgrounds. For more details, see Castryck et al. [3].

2.1 Protocol of CSIDH

For describing the protocol of CSIDH, we define the following notations. Let p be a prime number, $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ 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

$$\mathcal{CL}(\mathbb{Z}[\sqrt{-p}]) \times \mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}]) \to \mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}]),\ (\mathfrak{a}, E) \mapsto \mathfrak{a} * E.$$

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 $\mathfrak{a} \in \mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ as her secret key and sends the curve $\mathfrak{a} * E$ to Bob as her public key. Bob proceeds in the same way by choosing a secret key $\mathfrak{b} \in \mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$. Then, both parties can compute the shared secret $\mathfrak{ab} * E = \mathfrak{ba} * E$. Note that $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ 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 $\ell_1, \ldots, \ell_n$ 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 [13] 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.

We define a set

$$\mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}]) = \frac{\{E/\mathbb{F}_p : \text{a supersingular elliptic curve} \mid \mathrm{End}_{\mathbb{F}_p}(E) \cong \mathbb{Z}[\sqrt{-p}]\}}{\mathbb{F}_p\text{-isomorphism}},$$

where $\mathrm{End}_{\mathbb{F}_p}(E)$ is the $\mathbb{F}_p$-endomorphism ring of E. This is not an empty set, and for all classes $\mathcal{E} \in \mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$, 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 $\mathcal{E}$ ([3, Theorem 8]).

The ideal class group $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ of $\mathbb{Z}[\sqrt{-p}]$ acts $\mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ 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 $\mathfrak{a} \in \mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$, there are an elliptic curve $E' \in \mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$ and an isogeny $\varphi : E \to E'$, 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 $(\mathfrak{a}, E) \mapsto E'$ does not depend on the choices of the representatives E and $\mathfrak{a}$ or of the isogeny φ. This map defines an action of $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ on $\mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$. We denote the curve E' described above by $\mathfrak{a} * E$. The action $(\mathfrak{a}, E) \mapsto \mathfrak{a} * E$ is free and transitive ([3, Theorem 7]). According to the Brauer-Siegel theorem [25], the cardinality of $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ is asymptotically

$$\#\mathcal{CL}(\mathbb{Z}[\sqrt{-p}]) \approx \sqrt{p}.$$

This is the size of the key space of CSIDH.

To compute the action of an ideal $\mathfrak{a} \in \mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ 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

Algorithm 1 Evaluating the class group action in CSIDH

Input: $A \in \mathbb{F}_p$, $m \in \mathbb{N}$, a list of integers $(e_1, \ldots, e_n)$ s.t. $-m \leq e_i \leq m$ for $i = 1, \ldots, n$, and distinct odd primes $\ell_1, \ldots, \ell_n$ 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 $i = 1, \ldots, n$, and π is the p-th power Frobenius endomorphism of E_A.

1: **while** some $e_i \neq 0$:
2: Sample a random $x \in \mathbb{F}_p \backslash \{0\}$.
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 $\varphi : E_A \to E_B$ 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.

$\bar{\mathfrak{l}}_i = (\ell_i, \pi + 1)$. For $E \in \mathcal{ELL}_{\mathbb{F}_p}(\mathbb{Z}[\sqrt{-p}])$, the torsion subgroups of these ideals can be written as

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

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 defined by the equation $y^2 = x^3 + Ax^2 + x$, the x-coordinate of a point of $E[\bar{\mathfrak{l}}_i]$ is in $\mathbb{F}_p$. Therefore, 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 $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$, $e_1, \ldots, e_n \in \mathbb{Z}$ by the composition of the actions of $\mathfrak{l}_i$ and $\bar{\mathfrak{l}}_i$. Castryck et al. [3] 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 $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$, where $m \in \mathbb{N}$ such that $(2m+1)^n \geq \#\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$. We denote the exponents (e_i) by secret exponents.

2.3 Computing the class group action $\mathfrak{a} * E$

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 $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ by secret exponents $(e_1, \ldots, e_n)$

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 [21] 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 makes 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 makes 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 $E[\overline{\mathfrak{l}}_i]$ if $s = -1$. If Q is not the point at infinity, we compute the isogeny φ with kernel $\langle Q \rangle$ and update a curve coefficient, the k-torsion 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 Q and in the kernel of the isogeny φ. Updating k reduces the scalar of the next scalar multiplication in the inner loop.

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. [19] and briefly introduce related works.

3.1 Constant-time implementation

As already mentioned by Castryck et al. [3], Algorithm 1 is not side-channel resistant because the computational time for a public key depends on the associated secret key. To solve this problem, Meyer et al. proposed a constant-time implementation of CSIDH. According to them [19], "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 in 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. In this algorithm, one samples a point in an elliptic curve by using Elligator [1] for CSIDH, which was proposed by Bernstein, Lange, Martindale and Panny [2]. Elligator enables us to generate x-coordinates of P by computing only one Legendre symbol. For the details, see Bernstein et al. [2].

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. Meyer and Reith [20] 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 additions on an elliptic curve. For more details, see Meyer et al. [20, 19].

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 reason the sizes of the scalar multiplications vary is that the integer k in Algorithm 1 depends on the signs of secret exponents. To remove this effect, Meyer et al. proposed changing the intervals from $[-m, m]$ to $[0, 2m]$. In other words, they used only non-negative secret exponents.

Algorithm 2 Constant-time evaluation of the class group action in CSIDH [19]

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

1: Set $e'_i = 2m - e_i$ for $i = 1, \ldots, n$.
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 $\varphi : E_A \to E_B$ 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.

3.2 Constant-time implementations based on another definition

As we stated above, Meyer et al. 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. [2] 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 [17] proposed a constant-time implementation for classical computers, which also has no branches. The computational time of their implementation is determined by the worst case of choices of the torsion point Q excluding negligible probability. Therefore, the implementation by Jalali et al. is slower than that of Meyer et al. In this paper, we propose a constant-time implementation based on the definition by Meyer et al. and leave further discussion about constant-time as a future work.

3.3 Speedup techniques for CSIDH

Meyer et al. [19] 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 (Splitting isogeny computations into multiple batches). SIMBA 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. 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. The same as Meyer et al. [19], 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 Meyer et al.), 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

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

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. We call this technique weighted secret exponents. We discuss its effect on the security of CSIDH in Section 5.4.

4 Our constant-time implementation

In this section, we propose a new constant-time implementation that is faster than that of Meyer et al.

The constant-time implementation by Meyer et al. requires the cost to be the same as that of calculating the action of the ideal class corresponding to secret exponents $(2m, \ldots, 2m)$. This cost is twice the cost corresponding to secret exponents $(m, \ldots, m)$, 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 Meyer et al. [20, 19], 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. The essential part of our algorithm is as follows:

1. Generate (x-coordinates of) two points $P_0 \in E[\pi - 1]$ and $P_1 \in E[\pi + 1]$.
2. Make k-torsions: $P_0 \leftarrow [(p+1)/k]P_0,\ P_1 \leftarrow [(p+1)/k]P_1$.
3. For $i = 1, \ldots, n$, repeat the following.
4. Set the indicator s for a sign: Set s the sign bit of e_i so that $s = 0$ 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.
5. If $e_i \neq 0$, then do the following:
   (a) Make a ℓ_i-torsion point: $Q \leftarrow [k/\ell_i]P_s$.
   (b) Drop the ℓ_i-torsion part of P_{1-s}: $P_{1-s} \leftarrow [\ell_i]P_{1-s}$.
   (c) If $Q = \infty$, skip this i.
   (d) Compute an isogeny $\varphi : E_A \to E_B$ with kernel $\langle Q \rangle$ ($\langle Q \rangle = E[\mathfrak{l}_i]$ or $E[\bar{\mathfrak{l}}_i]$ according to the sign of e_i).

(e) Update $A \leftarrow B$, $P_0 \leftarrow \varphi(P_0)$, $P_1 \leftarrow \varphi(P_1)$, $k \leftarrow k/\ell_i$, and $e_i \leftarrow e_i - 1 + 2s$.

6. If $e_i = 0$ and the number of isogeny computations does not reach the maximum number, then compute a dummy isogeny.

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 $(e_1, \ldots, e_n)$ s.t. $-m \leq e_i \leq m$ for $i = 1, \ldots, n$, and distinct odd primes $\ell_1, \ldots, \ell_n$ 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 $i = 1, \ldots, n$, and π is the p-th power Frobenius endomorphism of E_A.

1: Set $e_i' = m - |e_i|$ for $i = 1, \ldots, n$.
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 $\varphi : E_A \to E_B$ 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.

Remark 1. The same as in the implementation by Meyer et al., we use Elligator for CSIDH. It enables us to generate x-coordinates of P_0 and P_1 in line 1 in the above table by computing only one Legendre symbol. For the details, see Bernstein et al. [2].

Remark 2. 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.

We need a scalar multiplication on P_{1-s} by ℓ_i in line 5b in the above table 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 contract, 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 is the parameter set for CSIDH proposed by Castryck et al. [3].

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 by Meyer et al.

Meyer et al. claimed that their implementation is constant-time in the sense that it can prevent the two leakage scenarios they consider [19, §3]: timing leakage and power analysis. Timing leakage is leaking information on a secret key by the computational time. Power analysis 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 by Meyer et al. 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 by Meyer et al. 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 by Meyer et al. is in line 10 in Algorithm 2). 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. Therefore, this does not affect whether our implementation is constant-time. As a result, we conclude both implementations have equivalent security.

5 Evaluation of our implementation

In this section, we discuss the computational cost of constant-time implementations of CSIDH. We focus on CSIDH-512 [3], 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 $i = 1, \ldots, 73$ and $\ell_{74} = 587$.

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 [7] and isogenies by the formula of Costello and Hisil [5] and Meyer and Reith [20]. 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 [5]. 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 [20, §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.

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

Function	**M**	**S**	**a**
`xDBLADD` [6]	8	4	8
`xADD` [6]	4	2	6
`xDBL` [6]	4	2	4
`LADDER` [7]	$8t-4$	$4t-2$	$8t-6$
`Kernel_Points` [5]	$4(d-1)$	$2(d-1)$	$2(3d-4)$
`OddIsogeny_Points` [5]	$4d$	2	$2(d+1)$
`OddIsogeny_Curve` [20]	$2d+t'$	$2t'+6$	6

As we stated in Section 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 Section 3.3. The best

parameters we found are SIMBA-3-8 and weighted secret exponents $-m_i \leq e_i \leq m_i$ for $i = 1, \ldots, 74$, where

$$\begin{aligned}(m_i) = (&5,6,7,7,7,7,7,8,8,8,9,10,10,10,10,9,9,9,8,7,7,7,7,7,7,7,7,7,\\ &7,7,7,6,6,6,6,6,5,5,5,5,5,5,5,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,\\ &3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,1).\end{aligned}$$

On the other hand, the parameters Meyer et al. proposed are SIMBA-5-11 and the secret exponents $0 \leq e_i \leq m_i$ for $i = 1, \ldots, 74$, where

$$\begin{aligned}(m_i) = (&13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,\\ &13,13,13,13,13,13,13,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,\\ &11,11,11,8,8,8,8,8,8,8,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,5).\end{aligned}$$

Note that our (m_i) do not line up in descending order for the following reason. The reason is as follows. One fails to obtain a generator for the kernel of an isogeny of small degree with a higher probability than that of a higher degree. Therefore, small degree isogenies are left to compute when large degree isogenies are finished. The number of the outer while loops in the algorithm is determined by an isogeny left at the end. We can decrease this number by taking small intervals for small degree isogenies.

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 in 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 by Meyer et al. and ourselves. We used speedup parameters proposed by Meyer et al. [19, §6] for the implementation by Meyer et al. In the following tables, the notation $\overline{\mathbf{M}}$ means the cost measured in the number of multiplications, assuming $\mathbf{S} = 0.8\mathbf{M}$ and $\mathbf{a} = 0.05\mathbf{M}$ the same as Meyer and Reith [20]. Table 2 shows the numbers of operations on $\mathbb{F}_p$ in the implementation by Meyer et al. with the speedup techniques in Section 3.3, and Table 3 shows that of our implementation with the parameters in Section 5.2. The cost of our implementation is 962,022 $\overline{\mathbf{M}}$, which is about 28.23% less than that of the implementation by Meyer et al.

We implemented our algorithm with the speedup techniques in C. Our code is based on the code by Meyer et al. [4] (originally from Castryck et al. [3]). Table 4 shows the cycle counts and running times for out implementation and that by Meyer et al. For the implementation by Meyer et al., we used the code on which our code is based. We ran both codes on an Intel Xeon Gold 6130 Skylake

[4] The code by Meyer et al. is available for download at `https://zenon.cs.hs-rm.de/pqcrypto/constant-csidh-c-implementation`. The commit ID of the version we used is 4d223c65, the latest version on 15 Jan, 2019.

Table 2. The numbers of operations on $\mathbb{F}_p$ in the implementation by Meyer et al. with the speedup techniques in Section 3.3.

	Outer SM (line 6 in Algorithm 2)	Inner SM (line 8 in Algorithm 2)	Isogenies (lines 11–12, 14–15 in Algorithm 2)	Total number
M	220,178	227,810	560,961	1,008,950
S	110,089	113,905	127,467	351,462
a	220,050	226,238	562,459	1,008,749
$\overline{\mathbf{M}}$	319,252	330,247	691,058	1,340,558

Table 3. The numbers of operations on $\mathbb{F}_p$ in our implementation with the parameters in Section 5.2.

	Outer SM (line 6 in Algorithm 3)	Inner SM (lines 9–10 in Algorithm 3)	Isogenies (lines 13–14, 16–17 in Algorithm 3)	Total number
M	176,891	187,422	368,651	732,966
S	88,445	93,711	61,681	243,838
a	176,784	185,770	318,246	680,801
$\overline{\mathbf{M}}$	256,487	271,680	433,908	**962,077**

processor running Ubuntu 16.04.5 LTS. Our implementation has 27.35% fewer clock cycles than the implementation by Meyer et al., which is almost the same as the reduction ratio expected by the evaluation of our cost model.

Table 4. Performance comparison, averaged over 10,000 runs.

	Cost evaluation	Clock cycles $\times 10^6$	Wall clock time
Implementation by Meyer et al.	1,340,558$\overline{\mathbf{M}}$	237.3	113.237ms
Our implementation	**962,077**$\overline{\mathbf{M}}$	**172.4**	**82.272ms**

5.4 Security of weighted secret exponents

We discuss the security of using weighted secret exponents. According to the same discussion by Castryck et al. [3, §7.1], we can expect ideals of the form $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$ with weighted secret exponents to represent uniformly "almost" all the ideal classes in $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$.

By a heuristic of Cohen and Lenstra [4], we assume that $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ is "almost cyclic," i.e., $\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$ has a cyclic subgroup of order N such that $N \approx \#\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$. We define

$$\rho : \mathcal{CL}(\mathbb{Z}[\sqrt{-p}]) \to \mathbb{Z}/N\mathbb{Z}$$

by a projection to the large subgroup and $\alpha_i = \rho(\mathfrak{l}_i)$. The number of generators of $\mathbb{Z}/N\mathbb{Z}$ is $\phi(N)$, where ϕ is Euler's totient function. The probability that a random element of $\mathbb{Z}/N\mathbb{Z}$ is a generator is $\phi(N)/N \geq 1/2$. Therefore, at least one α_i generates $\mathbb{Z}/N\mathbb{Z}$ with high probability. We may assume $\alpha_1 = 1$. We show that for any $M \in \mathbb{Z}$, the congruence

$$e_1 + e_2\alpha_2 + \cdots + e_n\alpha_n \equiv M \pmod{N}$$

has solutions (e_i) such that $-m_i \leq e_i \leq m_i$, $i = 1, \ldots, n$ and the number of the solutions does not depend on M. This means that the restriction of ρ to the set of the ideal classes of the form $\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$ is surjective and uniform. Since $N \approx \#\mathcal{CL}(\mathbb{Z}[\sqrt{-p}])$, this is what we should show. We define a lattice $\mathcal{L}$ in $\mathbb{R}^n$ spanned by the rows of the matrix

$$L = \begin{pmatrix} N & 0 & 0 & \cdots & 0 \\ -\alpha_2 & 1 & 0 & \cdots & 0 \\ -\alpha_3 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ -\alpha_n & 0 & \cdots & 0 & 1 \end{pmatrix},$$

a set

$$B = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid -m_i - 1/2 < x_i < m_i + 1/2 \text{ for } i = 1, \ldots, n\}$$

and a set $B_M = B + (M, 0, \ldots, 0)$. The number of the solutions for the above congruence in the intervals $[-m_i, m_i]$ is the same as the number of elements in $\mathcal{L} \cap B_M$. The Gaussian heuristic claims

$$\#(\mathcal{L} \cap B_M) \approx \mathrm{vol}(B_M)/\det(L) = \prod_i (2m_i + 1)/N.$$

Therefore, if $\prod_i (2m_i + 1) \geq N$, the above congruence has a solution in the intervals $[-m_i, m_i]$ and the number of the solutions does not depends on M. As stated in Section 2.2, we have $\#\mathcal{CL}(\mathbb{Z}[\sqrt{-p}]) \approx \sqrt{p}$, so we may choose (m_i), which satisfies $\prod_i (2m_i + 1) \geq \sqrt{p}$.

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 by Meyer et al. Whereas Meyer et al. 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 in an elliptic curve. The additional cost for calculation associated with this point is less than the additional cost of Meyer et al. to achieve constant-time. Consequently, our implementation is faster than the implementation by Meyer et al.

We evaluated these costs by counting the number of operations on $\mathbb{F}_p$. This evaluation showed that our implementation reduces the cost by 28.23% compared with the implementation by Meyer et al. We tested this reduction ratio by implementing our algorithm in C and measuring its clock cycles. The reduction ratio measured by clock cycles is 27.35%. 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.

Our implementation allows variance the computational time with randomness that does not relate to secret information. Applying our method to an implementation based on a stricter definition of constant-time is a future work.

References

1. D. J. Bernstein, M. Hamburg, A. Krasnova, T. Lange.: Elligator: Elliptic-curve points indistinguishable from uniform random strings. Proceedings of the 2013 ACM Conference on Computer & Communications Security, 967–980 (2013).
2. D. J. Bernstein, T. Lange, C. Martindale, L. Panny.: Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. IACR Cryuptography ePrint Archive 2018/1059; `https://eprint.iacr.org/2018/1059` (to appear at Eurocrypt 2019).
3. W. Castryck, T. Lange, C. Martundale, L. Panny, J. Renes.: CSIDH: An efficient post-quantum commutative group action. ASIACRYPT 2018, LNCS 11274, 395–427 (2018).
4. H. Cohen and H. W. Lenstra, Jr.: Heuristics on class groups of number fields. Number Theory, Noordwijkerhout 1983, 33–62 (1984).
5. C. Costello, H. Hisil.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. ASIACRYPT 2017, LNCS 10625, 303–329 (2017).
6. C. Costello, P.Longa, M. Naehrig.: Efficient algorithms for supersingular isogeny Diffie-Hellman. CRYPTO 2016, LNCS 9814, 572–601 (2016).
7. C. Costello, B. Smith.: Montgomery curves and their arithmetic. Journal of Cryptographic Engineering **8**(3), 227–240 (2018).
8. J-M. Couveigne.: Hard homogeneous spaces. IACR Cryptology ePrint Archive 2006/291; `https://eprint.iacr.org/2006/291`.
9. L. De Feo.: Mathematics of isogeny based cryptography. arXiv:1711.04062 (2017).
10. L. De Feo, J. Kieffer, B. Smith.: Towards practical key exchange from ordinary isogeny graphs. ASIACRYPT 2018, LNCS 11274, 365–394 (2018).
11. L. De Feo, S. D. Galbraith.: SeaSign: Compact isogeny signatures from class group actions. IACR Cryptology ePrint Archive 2018/824; `https://eprint.iacr.org/2018/824` (to appear at Eurocrypt 2019).

12. T. Decru, L. Panny, F. Vercauteren.: Faster SeaSign signatures through improved rejection sampling. Cryptology ePrint Archive, Report 2018/1109; `https://eprint.iacr.org/2018/1109` (to appear at PQCrypto 2019).
13. C. Delfs, S. D. Galbraith, Computing isogenies between supersingulrar elliptic curves over $\mathbb{F}_p$. Designs, Codes and Cryptography **78**(2), 425–440 (2016).
14. S. D. Galbraith, C. Petit, B. Shani, Y. B. Ti.: On the security of supersingular isogeny cryptosystems. ASIACRYPT 2016, LNCS 10031, 63–91 (2016).
15. S. D. Galbraith, F. Vercauteren.: Computational problems in supersingular elliptic curve isogenies. Quantum Information Processing **17**(10), 265 (2018).
16. D. Jao, L. De Feo.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. PQCrypto 2011, LNCS 7071, 19–34 (2011).
17. A. Jalali, R. Azarderakhsh, M. M. Kermani, D. Jao.: Towards optimized and constant-time CSIDH on embedded devices. IACR Cryptology ePrint Archive 2019/297; `https://eprint.iacr.org/2019/297`. (to apper at COSADE 2019).
18. D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, V. Soukharev, D, Urbanik.: Supersingular isogeny key encapsulation. Submission to the NIST Post-Quantum Cryptography Standardization project; `https://sike.org`.
19. M. Meyer, F. Campos, S. Reith.: On Lions and Elligators: An efficient constatn-time implementation of CSIDH. IACR Cryptology ePrint Archive 2018/1198; `https://eprint.iacr.org/2018/1198` (to appear at PQCrypto 2019).
20. M. Meyer, S. Reith.: A faster way to the CSIDH. INDOCRYPT 2018, LNCS 11356, 137–152 (2018).
21. P. L. Montgomery.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**(177), 24–264 (1987).
22. National Institute of Standards and Technology (NIST): NIST Post-Quantum Cryptography Standardization; `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`, (2016).
23. C. Petit.: Faster algorithms for isogeny problems using torsion point images. ASIACRYPT 2017, LNCS 10625, 330-353 (2017).
24. A. Rostovtsev, A. Stolbunov.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive 2006/145; `https://eprint.iacr.org/2006/145`.
25. C. Siegel.: Über die Classenzahl quadratischer Zahlkörper. Acta Arithmetica **1**(1) 83–86 (1935).
26. A. Stolbunov.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves, Advances in Mathematics of Communications **4**(2), 215–235 (2010).