Efficient Algorithm to Compute Odd-Degree Isogenies Between Montgomery Curves for CSIDH*

Kenta KODERA†(a), Nonmember, Chen-Mou CHENG††(b), and Atsuko MIYAJI†††(c), Members

SUMMARY Isogeny-based cryptography, such as commutative supersingular isogeny Diffie-Hellman (CSIDH), have been shown to be promising candidates for post-quantum cryptography. However, their speeds have remained unremarkable. This study focuses on computing odd-degree isogeny between Montgomery curves, which is a dominant computation in CSIDH. Our proposed “2-ADD-Skip method” technique reduces the required number of points to be computed during isogeny computation. A novel algorithm for isogeny computation is also proposed to efficiently utilize the 2-ADD-Skip method. Our proposed algorithm with the optimized parameter reduces computational cost by approximately 12% compared with the algorithm proposed by Meyer and Reith. Further, individual experiments for each degree of isogeny show that the proposed algorithm is the fastest for $19 \leq \ell \leq 373$ among previous studies focusing on isogeny computation including the $O(\sqrt{\eta})$ algorithm proposed by Bernstein et al. The experimental results also show that the proposed algorithm achieves the fastest on CSIDH-512. For CSIDH-1024, the proposed algorithm is faster than the algorithm by Meyer and Reith although it is slower than the algorithm by Bernstein et al.

key words: post-quantum cryptography, isogeny, Montgomery curves

1. Introduction

1.1 Overview

Post-quantum cryptography has been intensively studied in response to threats brought about by the rapid development of quantum computing. Isogeny-based cryptography, such as supersingular isogeny Diffie-Hellman (SIDH) and commutative SIDH (CSIDH) [2], have been considered to be promising candidates for post-quantum cryptography. In particular, supersingular isogeny key encapsulation (SIKE) [3], which is an SIDH-based key encapsulation algorithm, has recently emerged as an alternative algorithm in the third round of the post-quantum cryptography standardization process promoted by the National Institute of Standards and Technology (NIST). Among all candidates in the competition, SIKE has the smallest public-key size [4]. In 2018, CSIDH was proposed as a non-interactive key exchange protocol with an even smaller public-key size than that of SIDH. However, its speed performance is not as impressive and leaves considerable room for improvement.

There have been a series of research efforts attempting to tailor the CSIDH algorithm and its parameters to increase its speed [5]–[8]. Researchers have also focused on its two main subroutines, namely scalar multiplication and odd-degree isogeny computation between Montgomery curves. For example, Cervantes-Vázquez et al. sped up the addition chain for scalar multiplication [9], while Meyer and Reith reduced the cost of isogeny computation by using twisted Edwards curves [5]. Specifically, for an elliptic curve $E$ with a finite subgroup $\Phi$, there exists an elliptic curve $E'$ and an isogeny $\phi : E \rightarrow E'$ satisfying $\ker(\phi) = \Phi$. Isogeny computation comprises computing points in the kernel $\Phi$, $E'$, and $\phi(P) \in E'$ for $P \in E$. Meyer and Reith have observed that $E'$ can be more efficiently computed by using the isomorphic curve in the twisted form of the Edwards curves. Additionally, Bernstein et al. have discussed a different sequence to compute points in $\Phi$ [10]. Their approach still requires the computation of the same number of points in $\Phi$. Further, Bernstein et al. achieved isogeny computation with $O(\sqrt{\eta})$ [11]. Both [11] and this study, which are independently conducted, compute less number of points in $\Phi$. As another important issue on CSIDH, constant-time algorithms and their optimizations have also been explored to protect against side-channel attacks [6], [9], [12].

This study investigates the reduction of the cost of isogeny computation to realize compact and efficient post-quantum cryptography. While conventional algorithms precisely compute the points in $\Phi$, we herein focus on reducing the number of points to be computed. Our proposed algorithm is also applicable to these constant-time algorithms.

1.2 Contributions

This study focuses on the isogeny computation with odd degrees $\ell$ between Montgomery curves. The “2-ADD-Skip method” is proposed to reduce the number of points to be computed during isogeny computation. Specifically, a novel efficient algorithm to utilize the 2-ADD-Skip method is presented, and its computational cost is analyzed in terms of the required number of field multiplication, squaring, and addition. Furthermore, the optimized parameter for the proposed algorithm is discussed. The proposed algorithm with the parameter is estimated to reduce the computational cost
Let $E$ be a field with char$(K) \neq 2$. A Montgomery curve [13] over $K$ with coefficients $a, b \in K, b(a^2 - 4) \neq 0$ is given by

$$M_{a,b} : \quad by^2 = x^3 + ax^2 + x.$$  

For a point $P$, the scalar multiplication by $k$ is denoted as $[k]P$.

Alternatively, e.g., following the work done by Costello, Longa, and Naehrig [14], Montgomery curves can be written as

$$M_{A,B,C} : \quad BY^2Z = CX^3 + AX^2Z + CXZ^2,$$

where $(A : B : C), (X : Y : Z) \in \mathbb{P}^2(K), C \neq 0, Z \neq 0, a = {A/C, b = {B/C, x = X/Z and y = Y/Z}. Let

$$\varphi_3 : \mathbb{P}^2(K) \longrightarrow \mathbb{P}^1(K), (X : Y : Z) \longmapsto (X : Z).$$

For the points $(X : Z) \in \mathbb{P}^1(K)$, Montgomery himself introduced efficient addition formulae [13]. Let $P, Q \in M_{a,b}(K), (X_P : Z_P) = \varphi_3(P)$, and $(X_Q : Z_Q) = \varphi_3(Q)$.

- When $P \neq Q$,

$$X_{P+Q} = Z_{P+Q}(X_PX_Q - Z_PZ_Q)^2$$

$$= Z_{P+Q}[(X_PZ_P)(X_Q + Z_Q)$$

$$+ (X_P + Z_P)(X_Q - Z_P)]^2,$$

$$Z_{P+Q} = X_{P+Q}(X_PZ_Q - Z_PX_Q)^2$$

$$= X_{P+Q}[(X_P - Z_P)(X_Q + Z_Q)$$

$$- (X_P + Z_P)(X_Q - Z_P)^2].$$

- When $P = Q$,

$$X_{[2]P} = 4C(X_P + Z_P)^2(X_P - Z_P)^2,$$

$$Z_{[2]P} = (4X_PZ_P)(4C(X_P - Z_P)^2$$

$$+ (A + 2C)(4X_PZ_P)).$$

From these formulae in $XZ$-only coordinates, two functions are defined:

$$ADD : (\varphi_3(P), \varphi_3(Q), \varphi_3(P - Q)) \longmapsto \varphi_3(P + Q),$$

$$DBL : (\varphi_3(P), (A : C)) \longmapsto \varphi_3([2]P).$$

Let $M, S$, and $a$ denote the computational costs of multiplication, squaring, and addition, respectively, in a field $K$. Then, $ADD$ requires $4M + 2S + a$ operations, and $DBL$ requires $4M + 2S + 8a$ operations as in [5, Sect. 3.2].

2.2 Isogenies

Let $E$ and $E'$ be elliptic curves. An isogeny from $E$ to $E'$ is a morphism $\phi : E \longrightarrow E'$ satisfying $\phi(O_E) = O_{E'}$. If $\phi$ is separable, then $\#\ker(\phi) = \deg \phi$. It is denoted as $\ell$-isogeny for $\ell = \deg \phi$. Let $\Phi$ be a finite subgroup of $E$. Then, there exist unique elliptic curve $E'$ up to isomorphism and separable isogeny $\phi : E \longrightarrow E'$ satisfying $\ker(\phi) = \Phi$. For given $E$ and $\Phi$, the Vélu's formula provides explicit equations for $E'$ and $\phi$ [15]. Isogeny computation comprises computing points in the kernel $\Phi$, computing the coefficients of the new curve $E'$ by referring to curve computation, and computing $\phi(P) \in E'$ for $P \in E$ by referring to image computation.

2.3 CSIDH

CSIDH is a non-interactive key-exchange protocol proposed by Castryck, Lange, Martindale, Panny, and Renes in 2018 [2]. Let $p$ be a prime number, and $\mathcal{E}(\mathbb{F}_p(\mathbb{Z}[\sqrt{-p}]))$ be a set of $\mathbb{F}_p$-isomorphism classes of supersingular Montgomery curves defined over $\mathbb{F}_p$ whose $\mathbb{F}_p$-endomorphism ring is isomorphic to $\mathbb{Z}[\sqrt{-p}]$. The ideal class group $\text{Cl}(\mathbb{Z}[\sqrt{-p}])$ acts freely and transitively on the set $\mathcal{E}(\mathbb{F}_p(\mathbb{Z}[\sqrt{-p}]))$. Then, Castryck et al. constructed a Diffie–Hellman-style key-exchange protocol based on the Couveignes–Rostovtsev–Stolbunov scheme [16–18]. Let $E_0 \in \mathcal{E}(\mathbb{F}_p(\mathbb{Z}[\sqrt{-p}]))$. Alice chooses a secret $a \in \text{Cl}(\mathbb{Z}[\sqrt{-p}])$ and generates her public key $E_a = a \cdot E_0$ by computing the action given by $a$. Bob also computes his public key $E_b$ with his secret $b$. Now, Alice and Bob can compute a shared secret $a \cdot b \cdot E_0 = a \cdot E_b = b \cdot E_a$ by the
computation of \( \text{Cl}(\mathbb{Z}[\sqrt{-p}]) \).

The action provided by a class group element \( a \in \text{Cl}(\mathbb{Z}[\sqrt{-p}]) \) is defined by an isogeny \( \phi : E \rightarrow E/[a] \), where \( E[a] = \text{ker}(a) \). In CSIDH, a prime of the form \( p = 4 \prod_{i=1}^{e} \ell_i - 1 \) is used, where \( \ell_i \)'s are odd primes. The principal ideal \( (\ell_i) \) splits into \( \ell_i \) and \( \ell_i = (\ell_i, \pi + 1) \) over \( \mathbb{Z}[\sqrt{-p}] \), where \( \pi \) denotes the Frobenius endomorphism. Hence, the action provided by \( \ell_i \) corresponds to the \( \ell_i \)-isogeny whose kernel is generated by a point over \( \mathbb{F}_p \).

Similarly, the action provided by \( \ell_i \) corresponds to the \( \ell_i \)-isogeny whose kernel is generated by a point over \( \mathbb{F}_p \). We have \( \ell_i = \ell_i^{-1} \in \text{Cl}(\mathbb{Z}[\sqrt{-p}]) \). Subsequently, an element in the class group is sampled by \( a = \prod \ell_i^{e_i} \) for small integers \( e_i \)'s. Hence, a secret key is represented by a vector \( e = (e_1, \ldots, e_n) \). Therefore, given a secret vector \( e \), the main computation in CSIDH is the evaluation of the class group action. This requires \( |e| \) operations to compute each \( \ell_i \)-isogenies for all odd prime factors \( \ell_i \) of \( p + 1 \). Furthermore, the public keys and shared secrets are elements in \( \mathbb{F}_p \) because curves in \( \mathcal{E}(\mathbb{F}_p, \mathbb{Z}[\sqrt{-p}]) \) can be represented by supersingular Montgomery curves of the form \( E_a : y^2 = x^3 + ax^2 + x \) for \( a \in \mathbb{F}_p \).

The high-level concept of the class group action evaluation is provided in Algorithm 1. The scalar multiplications in lines 6 and 8 and the odd-degree isogeny computations in line 10 are the primary subroutines in this algorithm. Furthermore, the computational cost of the evaluation of class group actions depends on the secret vector \( e \), which can be a target in side-channel attacks. To protect against the attacks, Meyer, Campos, and Reith proposed a constant-time algorithm by constructing a dummy computation of \( \ell_i \)-isogenies [6].

In a parameter set CSIDH-512, all \( e_i \)'s are chosen from the interval \([-5, \ldots, 5]\). Further, \( p + 1 \) has 74 odd prime factors: \( \ell_1 = 3, \ell_2 = 5, \ldots, \ell_73 = 373, \) and \( \ell_74 = 587 \). \( \ell_i \) happens to be the \( i \)-th smallest odd prime in all cases except the last \( \ell_74 \). This parameter set is estimated to provide a security level of NIST-1.

### 3. Isogeny Computation on Montgomery Curves

In this section, we summarize algorithms for the isogeny computation on Montgomery curves with an odd degree \( \ell = 2d + 1 \), where an isogeny is given by \( \phi : M_{AB,C} \rightarrow M_{A',B',C} \). These algorithms are constructed based on an explicit formula derived by Costello and Hisil [19]. Given a generator \( (X_1 : Z_1) \), where \( (X_i : Z_i) := \varphi_s(i)[P] \) for \( \ker(\phi) = \langle P \rangle \), all points \( (X_2 : Z_2), \ldots, (X_d : Z_d) \) are precisely computed using addition formulas. Moreover, a technique is described to speed up the curve computation by using the twisted Edwards curves.

Hereinafter, \( (X_i : Z_i) \) is denoted as a point in the \( \ker(\phi) \), as given above. Furthermore, \( (X : Z) \in M_{AB,C} \) and \( (X' : Z') := \phi(X : Z) \in M_{A',B',C} \) are defined for image computation, and \( (A' : C') \) and \( (A : C) \) are denoted as curve coefficients in curve computation.

#### 3.1 Costello–Hisil Formula

Costello and Hisil derived an explicit formula for computing odd-degree isogenies between Montgomery curves [19]. Let \( M_{A,b} : by^2 = x^3 + ax^2 + x \) be Montgomery curve over a field \( K \) with \( \text{char}(K) \neq 2 \). Let \( P \in M_{A,b} \) be a point of order \( \ell = 2d + 1 \). Then, let \( \phi : M_{A,b} \rightarrow M_{A',b'} \) be the \( \ell \)-isogeny whose kernel \( \langle P \rangle \). Costello and Hisil showed that the rational map is given by

\[
\phi : (x, y) \mapsto (f(x), yf'(x)),
\]

where

\[
f(x) = x \prod_{i=1}^{d} \frac{(x - x_i[p] - 1)^2}{(x - x_i[p])},
\]

and \( f'(x) \) is its derivative. Furthermore, the codomain curve \( M_{A',b'} \) is given by

\[
a' = (6b' - 6a + a)\gamma^2 \quad \text{and} \quad b' = b\gamma^2,
\]

where \( \gamma = \sum_{i=1}^{d} x_i[p], \quad \tilde{\sigma} = \sum_{i=1}^{d} 1/x_i[p], \quad \gamma = \prod_{i=1}^{d} x_i[p], \) and \( x_i[p] \) denotes the \( x \)-coordinate of \( [i]P \). Rene provided a different proof and generalized the formula for any separable isogeny whose kernel does not contain \( (0, 0) \) [20].

In a projective space \( P^1(K) \), Eq. (4) leads to Eq. (6) for point computation:

\[
(X' : Z') = (X \cdot (S_X)^2 : Z \cdot (S_Z)^2),
\]

where

\[
S_X = \prod_{i=1}^{d} (XX_i - ZZ_i) \quad \text{and} \quad S_Z = \prod_{i=1}^{d} (XZ_i - XX_i).
\]

Similarly, Eq. (5) leads to Eq. (8) for curve computation:

\[
(A' : C') = (\tau(A - 3\sigma) : C),
\]
where $\tau = \prod_{i=1}^{\ell-1} \frac{X}{Z}$ and $\sigma = \sum_{i=1}^{\ell-1} \left( \frac{X}{Z} - \frac{X}{i} \right)$.

### 3.2 Improvements in Curve Computation

Castryck et al. defined $T_1$ as $\sum_{i=1}^{\ell-1} T_{iw} = \prod_{i=1}^{\ell-1} (Z_w + X_i)$ and computed a new coefficient efficiently by Eq. (9) [2, Sect. 8].

\[(A' : C') = \left( AT_0 T_{r-3} - 3C(T_0 T_{r-2} - T_1 T_{r-1}) : CT_{r-1}^2 \right). \tag{9} \]

Moreover, Meyer and Reith proposed a faster approach to curve computation by using twisted Edwards curves [5].

A twisted Edwards curve [21] over a field $K$ with coefficients $a, d \in K, ad \neq 0$, and $a \neq d$ is given by

\[ tE_{a,d} : av^2 + v^3 = 1 + ad^2 v^2. \]

Converting an elliptic curve between its Montgomery form with $b = 1$ and twisted Edwards form is possible at the cost of only a few field additions:

\[ a_{tE} = A + 2C, \quad d_{tE} = A - 2C, \text{ and} \]

\[ (A : C) = (2(a_{tE} + d_{tE}) : a_{tE} - d_{tE}). \tag{10} \]

In the context of isogeny computation, the efficient conversion is always available due to an isomorphism between $by^2 = x^3 + ax^2 + x$ and $\tilde{y}^2 = \tilde{x}^3 + ax\tilde{x} + \tilde{x}$ given by $y = \tilde{y}/\sqrt{b}$. Furthermore, a point $(X : Z)$ on a Montgomery curve over $\mathbb{P}^1(K)$ can be transformed to a point $(Y_{tE} : Z_{tE})$ in $YZ$-only coordinates on the corresponding twisted Edwards curve as follows:

\[(X : Z) \mapsto (Y_{tE} : Z_{tE}) = (X - Z : X + Z). \tag{11} \]

Moody and Shumow presented an explicit formula for $\ell$-isogeny computation on twisted Edwards curves [22]. Specifically, the curve computation in YZ-only coordinates is given by

\[ d_{tE}' = d_{tE}' \cdot \pi_X, \text{ and } d_{tE}' = d_{tE}' \cdot \pi_Y, \tag{12} \]

where

\[ \pi_X = \prod_{i=1}^{d} Z_{i E,i} \text{ and } \pi_Y = \prod_{i=1}^{d} Y_{i E,i}. \tag{13} \]

### 3.3 Conventional Algorithm for Isogeny Computation on Montgomery Curves

This section shows the conventional algorithm for odd-degree isogeny computation. As an example, Meyer and Reith’s Algorithm [5] is explained, which uses Eqs. (6) and (12). The same approach is applied for curve computation purely on Montgomery curves by Eq. (8).

In the conventional algorithm, intermediate variables such as $S_X, S_Z, \pi_Y, \text{ and } \pi_X$ are computed first. As shown in Eqs. (7) and (13), all these variables are polynomial in $(X_i : Z_i)$ for $i = 1, \ldots, d$; hence, they are computed through iteration. To begin with, all intermediate variables are initialized with $(X_i : Z_i)$ as follows:

\[ S_X = (X - Z)(X_i + Z_i) + (X + Z)(X_i - Z_i), \quad S_Z = (X - Z)(X_i + Z_i) - (X + Z)(X_i - Z_i), \]

\[ \pi_Y = X_i - Z_i, \text{ and } \pi_Z = X_i + Z_i. \]

Subsequently, for all $i = 2, \ldots, d$, a point $(X_i : Z_i)$ is computed by the addition formulae, and intermediate variables are successively updated by the following formulae.

\[
\begin{align*}
(S_X &\leftarrow S_X \cdot ((X - Z) \cdot (X_i + Z_i) + (X + Z) \cdot (X_i - Z_i)), \\
S_Z &\leftarrow S_Z \cdot ((X - Z) \cdot (X_i + Z_i) - (X + Z) \cdot (X_i - Z_i));
\end{align*}
\tag{14}
\]

\[
\begin{align*}
\pi_Y &\leftarrow \pi_Y \cdot (X_i - Z_i), \\
\pi_Z &\leftarrow \pi_Z \cdot (X_i + Z_i).
\end{align*}
\tag{15}
\]

The function combined with Eqs. (14) and (15) is defined:

\[ \text{UPDATE} : (S_X, S_Z, \pi_Y, \pi_Z, (X + Z), (X - Z), (X_i : Z_i)) \]

\[ \mapsto (S_X', S_Z', \pi_Y', \pi_Z'). \]

In the following, the update by this function will be written as $\text{UPDATE}(X_i : Z_i)$ for short. It requires $(4M + 4a) + 2M = 6M + 4a$ field arithmetic operations. Notably, $X_i \pm Z_i$ can be reused over the update formulae, and $X \pm Z$ can also be reused over several updates if they are computed once at the beginning of the algorithm. Finally, $(X' : Z')$ and $(A' : C')$ are computed by Eqs. (6) and (12).

Algorithm 2 summarizes the algorithm for $(\ell = 2d+1)$-isogeny computation. It requires a total of $(10d + \ell - 4)M + (2d + 2\ell + 6)S + (10d + 3)a$ field arithmetic operations, where $\ell$ denotes the bit length of $\ell$, and $(\ell/2)M + 3S$ is assumed to be required to compute the $\ell$-th power.

**Algorithm 2** Isogeny computation with degree $\ell = 2d + 1$

**Input**: $\ell, (X : Z), (X_i : Z_i), \text{ and } (A : C)$

**Output**: $(X' : Z')$ and $(A' : C')$

1. $(\pi_Y, \pi_Z) \leftarrow (X - Z, X + Z)$  // 2a
2. $(t, r') \leftarrow (X + Z, X - Z)$  // 2a
3. $(t_0, t_1) \leftarrow (r - \pi_Z, r + \pi_Y)$  // 2M
4. $(S_X, S_Z) \leftarrow (t_0 + t_1, t_0 - t_1)$  // 2a
5. for $i = 2$ to $(\ell/2) + 2d$ do
6. if $i = 2$ then
7. $(X_i : Z_i) \leftarrow \text{DBL}(X_i : Z_i), (A : C)$  // 4M + 2S + 8a
8. else
9. $(X_i : Z_i) \leftarrow \text{ADD}((X_{i-1} : Z_{i-1}), (X_i : Z_i), (X_{i-2} : Z_{i-2}))$  // 4M + 2S + 6a
10. end if
11. $(S_X, S_Z, \pi_Y, \pi_Z) \leftarrow \text{UPDATE}(X_i : Z_i)$  // 6M + 4a
12. end for
13. $(X' : Z') \leftarrow (X - S_X)^2 : Z \cdot (S_Z)^2$  // 2M + 2S
14. $(d_{tE}', d_{tE}') \leftarrow ((A + 2C)' \cdot \pi_X, (A - 2C)' \cdot \pi_Y)$  // $(2 + \ell)M + (2d + 6a)S + 3a$
15. $(A', C') \leftarrow (2d_{tE}' + d_{tE}', d_{tE}' - d_{tE})$  // 3a
16. return $(X' : Z')$ and $(A' : C')$
of this paper.

3.4 Isogeny Computation with Complexity $O(\sqrt{\ell})$

Toward the end of March 2020, Bernstein et al. proposed a new algorithm to compute $\ell$-isogeny by $O(\sqrt{\ell})$ [11]. A certain form of polynomial occurring in isogeny formula is evaluated by their systematic manner using the resultant computation. As a result, they successfully reduced the computational cost.

Particularly, their biquadratic relation in [16, Example 4.4], is essentially identical to Eq. (18) in Sect. 4, although both studies were conducted separately.

4. Proposed Approach

In this section, a novel technique for updating intermediate variables is proposed. This technique is called the “2-ADD-Skip method”, and we discuss how to apply the method to isogeny computation.

In Algorithm 2, given a point $(X_i : Z_i)$, the points in the kernel $(X_i : Z_i)$ for $i = 1, \ldots, d$ are precisely computed by the addition formulae and used for UPDATE function. However, the coordinates of all $d$ points are not necessarily required as long as the image $(X' : Z')$ and the new coefficient $(A' : C')$ can be computed. The 2-ADD-Skip method enables to perform updates in terms of two points, namely $(X_{m+n} : Z_{m+n})$ and $(X_{m-n} : Z_{m-n})$; this is achieved by using only $(X_m : Z_m)$ and $(X_n : Z_n)$ for $m \neq n$. By this method, two additions required to compute $(X_{m+n} : Z_{m+n})$ and $(X_{m-n} : Z_{m-n})$ are skipped.

To construct an efficient algorithm for isogeny computation with the 2-ADD-Skip method, points to be computed by addition formulae are divided into three sets, namely $M$, $N$, and $R$. The proposed algorithm utilizes 2-ADD-Skip methods for the pairs of points in the Cartesian product of $M \times N$. By considering ordinary updates in terms of points in $M$, $N$, and $R$, the algorithm performs updates in terms of $(X_i : Z_i)$ for $i = 1, \ldots, d$ in total.

4.1 2-ADD-Skip Method

The respective update formulae, Eqs. (14) and (15), are expressed as linear combinations of $X_i$ and $Z_i$. Hence, two times of updates in terms of $(X_i : Z_i)$ and $(X_j : Z_j)$ can be expressed by linear combinations of $X_iX_j$, $Z_iZ_j$, $X_iZ_j$, and $X_jZ_i$. Furthermore, the equations for image computation Eq. (7) and curve computation Eq. (12) show that all intermediate variables such as $S_X$ are symmetrical in terms of index $i$. For example, on expressing $S_X(X_1, X_2, \ldots, X_d, Z_d)$ as a polynomial in $X_1, X_2, \ldots, X_d, Z_d$, the relation $S_X(X_1, X_2, \ldots, X_d, Z_d) = S_X(X_d, X_{d-1}, \ldots, X_2, X_1)$ holds true for any $\sigma \in \mathbb{Z}_d$, where $\mathbb{Z}_d$ is the symmetric group of degree $d$. Therefore, if $X_iX_j$, $Z_iZ_j$, and $X_iZ_j + X_jZ_i$ exist for some indices $i$ and $j$, updates can be performed in terms of $(X_i : Z_i)$ and $(X_j : Z_j)$ according to the new update formulae in Eqs. (16) and (17), respectively:

$\begin{align*}
S_X &\leftarrow S_X \cdot (X_iZ_iZ_jZ_j - (X_iZ_j + X_jZ_i)) \\
&\quad = S_X \cdot ((X^2) \cdot (X_iX_j - (XZ)_i) + (Z^2) \cdot (Z_iZ_j)), \\
S_Z &\leftarrow S_Z \cdot ((X_iZ_i - Z_iX_i) \cdot (X_iZ_j - X_jZ_i)) \\
&\quad = S_Z \cdot ((X^2) \cdot (Z_iZ_j - (XZ)_i) \cdot (X_iZ_j + X_jZ_i)) + (Z^2) \cdot (X_iX_j); \\
\pi_Y &\leftarrow \pi_Y \cdot (X_iX_j + Z_iZ_j - (X_iZ_j + X_jZ_i)) \\
\pi_Z &\leftarrow \pi_Z \cdot (X_iX_j + Z_iZ_j + (X_iZ_j + X_jZ_i)).
\end{align*}$

Equations (16) and (17) require $7M + 4a$ and $2M + 3a$ respectively, as $X^2, XZ$ and $Z^2$ can be reused if they are computed once at the beginning of the algorithm.

For any indices $m$ and $n$ with $m \neq n$, $X_{m+n}X_{m-n}$, $Z_{m+n}Z_{m-n}$ and $M_{m+n}Z_{m-n} + M_{m-n}Z_{m-n}$ can be computed from $(X_m : Z_m)$ and $(X_n : Z_n)$ as follows. As demonstrated in [13, Sect. 10.3.1], let $(x_1, y_1) = (x_1, y_1) + (x_2, y_2)$ and $(x_4, y_4) = (x_1, y_1) - (x_2, y_2)$, where $(x_1, y_1) \neq (x_2, y_2)$, $(x_1, y_1)$ and $(x_2, y_2)$ lies in $\mathbb{F}_p(K)$. By the addition law, $x_3x_4$ and $x_3 + x_4$ can be expressed as

$\begin{align*}
x_3x_4 &= \frac{(1 - x_1x_2)^2}{(x_2 - x_1)^2}, \quad \text{and} \\
x_3 + x_4 &= \frac{2(x_1 + x_2)(x_1x_2 + 1) + 4ax_1x_2}{(x_2 - x_1)^2}. \\
\end{align*}$

They do not contain $y$-coordinates because $x_3x_4$ and $x_3 + x_4$ are uniquely determined when $x_1$ and $x_2$ are fixed. Considering them over $\mathbb{F}_p(K)$, we have

$\begin{align*}
X_3X_4 &= \frac{(x_1X_2 - Z_1Z_2)^2}{Z_3Z_4}, \\
X_3 + X_4 &= \frac{X_3Z_4 + X_4Z_3}{Z_3Z_4} \\
&= \frac{2C(X_1Z_2 + X_2Z_1)(X_1X_2 + Z_1Z_2)}{CZ_3Z_4} + \frac{4AX_1X_2Z_1Z_2}{CZ_3Z_4}.
\end{align*}$

Finally, we have

$\begin{align*}
X_{m+n}X_{m-n} &= C(X_mX_n - Z_mZ_n)^2, \\
Z_{m+n}Z_{m-n} &= C(X_mZ_n - X_nZ_m)^2, \quad \text{and} \\
X_{m+n}Z_{m-n} + X_{m-n}Z_{m+n} &= 2C(X_mZ_n + X_nZ_m)(X_mX_n + Z_mZ_n) + 4AX_nX_mZ_mZ_n.
\end{align*}$

Equation (19) can be computed by $9M + 3S + 7a$ field arithmetic operations using the following transformation:

$4AX_nX_mZ_mZ_n = A \left( (X_nX_m + Z_nZ_m)^2 - (X_nX_m - Z_nZ_m)^2 \right).$

In summary, if we have $(X_m : Z_m)$ and $(X_n : Z_n)$ for some
The function combined Eqs. (16), (17), and (19) is defined called the “2-ADD-Skip method” because two executions without using those points themselves. This technique is, computational cost of updates in terms of two points, $X^2\cdot$ADDSKIP$(X_i; Z_i)$: $(S_X, S_Z, \pi_X, \pi_Z, X^2, XZ, Z^2, (X_m: Z_m), (X_n: Z_n))$ 2ADDSKIP: $(S_X, S_Z, \pi_X, \pi_Z)$ $\rightarrow (S_X', S_Z', \pi_X', \pi_Z')$ In the following, the updates in terms of $(X_{m+n}: Z_{m+n})$ and $(X_{m-n}: Z_{m-n})$ by this function will be written as 2ADDSKIP$((X_m: Z_m), (X_n: Z_n))$ for short. 2ADDSKIP$((X_m: Z_m), (X_n: Z_n))$ for short. Table 1 shows the computational cost of computing updates in terms of two points, $(X_i: Z_i)$ and $(X_j: Z_j)$, with and without the 2-ADD-Skip method. Each time our proposed method is used for isogeny computation on Montgomery curves, $2M + 1S + 6a$ operations can be saved. Therefore, increasing the usage of this method leads to a faster algorithm for isogeny computation. The 2-ADD-Skip method can be applied to curve computation by Eq. (9). In this situation, the 2-ADD-Skip method can reduce more computational cost for updates in isogeny computation purely based on Montgomery curves.

### Efficient Algorithm with 2-ADD-Skip Method

To apply the 2-ADD-Skip method to correct isogeny computations, the following conditions should be satisfied:

- $(X_i: Z_i), (X_j: Z_j),$ and $(X_{i-j}: Z_{i-j})$ should be available before computing the point $(X_{i+j}: Z_{i+j})$;
- $(X_m: Z_m)$ and $(X_n: Z_n)$ should be available before computing updates in terms of $(X_{m+n}: Z_{m+n})$ and $(X_{m-n}: Z_{m-n})$, with 2-ADD-Skip method; and
- the result is equivalent to that when each update is performed exactly once in terms of $(X_i: Z_i)$ for $i = 2, \ldots, d$ without either duplication or omission.

Table 1: Computational cost of updates in terms of two points, $(X_i: Z_i)$ and $(X_j: Z_j)$, with and without the 2-ADD-Skip method.

<table>
<thead>
<tr>
<th>Computing $(X_i: Z_i)$ and $(X_j: Z_j)$ or $X_{i+j}$, $Z_{i+j}$</th>
<th>(ii) 2-ADD-Skip method</th>
<th>Diff (i)-(ii)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8M + 4S + 12a</td>
<td>9M + 3S + 7a</td>
<td>−1M + 1S + 5a</td>
</tr>
<tr>
<td>Updates for image computation</td>
<td>8M + 5a</td>
<td>7M + 4a</td>
</tr>
<tr>
<td>Updates for curve computation</td>
<td>4M</td>
<td>2M + 3a</td>
</tr>
<tr>
<td>Total</td>
<td>20M + 3S + 26a</td>
<td>18M + 3S + 14a</td>
</tr>
</tbody>
</table>

$m \neq n$, then the updates for curve and image computation in terms of $(X_{m+n}: Z_{m+n})$ and $(X_{m-n}: Z_{m-n})$ can be computed without using those points themselves. This technique is called the “2-ADD-Skip method” because two executions of ADD for $(X_{m+n}: Z_{m+n})$ and $(X_{m-n}: Z_{m-n})$ can be skipped. The function combined Eqs. (16), (17), and (19) is defined as follows:

$$2\text{ADDSKIP} : (X_i: Z_i), (X_j: Z_j) \rightarrow (S_X', S_Z', \pi_X', \pi_Z')$$

compute points placed on the lower side. One can imagine that iterating the block of updates can perform updates in terms of $(X_i: Z_i)$ for $i = 2, \ldots, d$ without either duplication or omission. Given the integers $d$ and $n$, integers $q$ and $r$ satisfying $d = q(2n + 1) + r + n$ are uniquely determined for $q \geq 1$ and $0 \leq r < 2n + 1$. Let

$$M = ((X_m: Z_m)) m = i(2n + 1) + r, 1 \leq i \leq q$$

An ordinary update of $(X_{m+i}: Z_{m+i})$, $(X_{n+i}: Z_{n+i})$ is performed. However, the $2qn$-times of point addition can be skipped by using Algorithm 3. As $r \neq 0$ holds for the case of prime $\ell$ and $q \geq 0$, some conditional branches can be omitted.

### 4.3 Analysis of the Proposed Algorithm

To determine the optimal value of a parameter $n$, the computational cost of Algorithm 3 is analyzed. All points to be computed by ADD (or DBL) during the proposed algorithm is in the set $U = (M \cup N \cup R) \setminus \{(X_1: Z_1)\}$. Moreover, the UPDATE in terms of points in $U$ and 2ADDSKIP in terms of pairs of points in the Cartesian product $M \times N$ are performed. Hence, the computational cost of the proposed algorithm is approximately given by

$$\text{Cost(Alg.3)} = (q + n - 1 + r) \cdot (\text{Cost(ADD)} + \text{Cost(UPDATE)}) + nq \cdot \text{Cost(2ADDSKIP)} + C,$$

where $C$ represents the computational cost for initialization and finalization. The computational cost in line 15 is ignored here for simplicity. Similarly, the computational cost of the conventional algorithm is approximately given by

$$\text{Cost(Alg.2)} = (d - 1) \cdot (\text{Cost(ADD)} + \text{Cost(UPDATE)}) + C.$$

With the equation $d = q(2n + 1) + r + n$, the amount of computational cost reduced by the proposed algorithm is given by
Algorithm 3 Isogeny computation with degree $\ell = 2d + 1$ utilizing the 2-ADD-Skip method

**Input:** $\ell \geq 9$, $n$, $(X : Z)$, $(X_1 : Z_1)$, and $(A : C)$

**Output:** $(X' : Z')$ and $(A' : C')$

1: $(x_1, x_2) \leftarrow (X_1 - Z_1, X_1 + Z_1) \quad // 2a$
2: $(x^*, r^*) \leftarrow (X + Z, X - Z) \quad // 2a$
3: $(t_0, t_1) \leftarrow (t^* - r^*x^*, r^*) \quad // 2M$
4: $(S_x : S_z) \leftarrow (t_0 + t_1, t_0 - t_1) \quad // 2a$
5: $(XX, XZ, ZZ) \leftarrow (X^2, XZ, Z^2) \quad // M + 2S$
6: $d \leftarrow (\ell - 1)/2$
7: $(q, r) \leftarrow [(d - n)/(2n + 1), (d - n) \mod (2n + 1)]$
8: $(X : Z) \leftarrow \text{DBL}(X_1 : Z_1, (A : C)) \quad // 4M + 2S + 8a$
9: $(S_x, S_z, x_1, x_2) \leftarrow \text{UPDATE}(X : Z) \quad // 6M + 4a$
10: for $i = 3$ to $n + r$ do
11: $(X_i : Z_i) \leftarrow \text{ADD}(X_{i-1} : Z_{i-1}, X_{i-2} : Z_{i-2}) \quad // 4M + 2S + 6a$
12: $(S_x, S_z, x_1, x_2) \leftarrow \text{UPDATE}(X_i : Z_i) \quad // 6M + 4a$
13: end for
14: if $2n + 1 > n + r$ then
15: $(X_{2n+1} : Z_{2n+1}) \leftarrow \text{ADD}(X_n : Z_n, X_{n+1} : Z_{n+1}, X_1 : Z_1) \quad // 4M + 2S + 8a$
16: end if
17: for $i = 1$ to $q$ do
18: $m_i \leftarrow i(2n + 1) + r$
19: if $i = 1$ then
20: $(X_1 : Z_1) \leftarrow \text{DBL}(X_{n+1} : Z_{n+1}, (A : C)) \quad // 4M + 2S + 8a$
21: else
22: $(X_{n+1} : Z_{n+1}) \leftarrow \text{ADD}(X_{n+1} : Z_{n+1}, X_{n+2} : Z_{n+2}, X_1 : Z_1) \quad // 4M + 2S + 6a$
23: $(X_{n+2} : Z_{n+2}) \leftarrow \text{UPDATE}(X_{n+1} : Z_{n+1}) \quad // 6M + 4a$
24: end if
25: end for
26: $(X_{n+1} : Z_{n+1}) \leftarrow \text{ADD}(X_{n+1} : Z_{n+1}, X_{2n+1} : Z_{2n+1}, X_{n+2} : Z_{n+2}) \quad // 4M + 2S + 6a$
27: end if
28: $(S_x, S_z, x_1, x_2) \leftarrow \text{UPDATE}(X_n : Z_n) \quad // 6M + 4a$
29: for $j = 1$ to $n$ do
30: $(S_x, S_z, x_1, x_2) \leftarrow \text{ADDSKIP}((X_j : Z_j, (A : C)) \quad // 18M + 3S + 14a$
31: end for
32: end for
33: $(X' : Z') \leftarrow (X - S_x : Z - S_z²) \quad // 2M + 2S$
34: $(a_{1E}^{M}, a_{1E}^{M}) \leftarrow ((A + 2C)^2, \pi^8 \left( A - 2C^2 \right) - \pi^8) \quad // 2 + \ell M + (2\ell + 6)S + 3a$
35: $(A' : C') \leftarrow (2a_{1E}^{M} + d_{1E}^{M}, a_{1E}^{M} - d_{1E}^{M}) \quad // 3a$
36: return $(X' : Z')$ and $(A' : C')$

Cost(Alg.2) - Cost(Alg.3) = $nq \cdot (2M + 1S + 6a)$.

Therefore, given a degree of isogeny $\ell = 2d + 1$, the parameter $n$ should be chosen so that $nq$, which is the number of 2ADDSKIP performed, is maximized.

For the sake of simplicity, it is assumed that $q > \frac{n - 1}{2d + 1}$, instead of taking floor function. Subsequently, $f(n) = nq = n\cdot \frac{n - 1}{2d + 1}$ with $n > 0$ and $d > 3$ has the unique global maximum at $q = nM$. Hence, $f(nM) = (\ell + 1 - 2\sqrt{\ell})/4$ is given with $q = nM$.

Furthermore, assuming $S = M$ and $a = 0M$, the maximum reduction rate can be estimated by

$$\text{Cost(Alg.2) - Cost(Alg.3)} \approx \frac{f(nM) \cdot 3M}{(d - 1) \cdot 12M + C} \approx 3(\ell + 1 - 2\sqrt{\ell}) \approx \frac{24\ell - 1 + 12 \log \ell}{d}$$

Therefore, our proposed algorithm can reduce the computational cost by approximately 12% for a large degree $\ell$.

4.4 Algorithm with More Use of the 2-ADD-Skip Method

Defining the sets of points $N$ and $M$ by referring the index system [11, Sect. 4.5], it is possible to construct an algorithm that performs almost all of the updates by the 2-ADD-Skip method. As points to be computed having the order $\ell$ and $\varphi(\mathcal{P}) = \varphi(-\mathcal{P})$ is satisfied, computing $(X_{l-1} : Z_{l-1})$ instead of $(X_l : Z_l)$ still works accurately. Hence, one can consider computing $d$ points in the kernel such that $S = \{(X_{l+1} : Z_{l+1}) \mid 0 \leq i < d\}$. Let $M = \{(X_n : Z_n) \mid m_i = 2b(2i + 1), 0 \leq i < b\}$ and $N = \{X_{2n+1} : Z_{2n+1} \mid 0 \leq i < b\}$, where $2b = \lfloor \sqrt{\ell - 1}/2 \rfloor$ and $b' = \lfloor (\ell - 1)/4b \rfloor$. Then, performing the 2-ADD-Skip method for the pairs of points in $M \times N$ covers almost all updates in terms of points in $S$.

Although this algorithm performs more of the 2-ADD-Skip method than Algorithm 3, its computational cost is worse. This is because the 2-ADD-Skip method takes more computational cost than ordinary updates when points to be updated are already obtained as in Table 1. Therefore, the points in $M$ and $N$ should be updated by UPDATE function, and it shows that Algorithm 3 is reasonably constructed.

4.5 Constructing Dummy Isogenies for Constant-Time CSIDH

As mentioned in Sect. 2.3, CSIDH implementations can be protected against side-channel attacks by always computing a constant number of $\ell$-isogenies for any secret $e$ by choosing appropriately between dummy and real isogeny computations. Because $\ell$-isogeny inherently removes a factor of $\ell$ from the order of the image point $\phi(P)$, the dummy isogeny computation must perform scalar multiplication $[\ell/P]$. Algorithm 3 can perform a dummy isogeny computation with two extra ADD operations, resulting in similar overheads as demonstrated in [6]. Specifically, $(X_{d-3n+1} : Z_{d-3n+1})$ and $(X_{d-n} : Z_{d-n})$ are contained in a set $M$ for any given $d$. Then,

$(X_{d+1} : Z_{d+1}) \leftarrow \text{ADD}(X_{d-3n+1} : Z_{d-3n+1}, X_{d-n} : Z_{d-n})$,
$(X_{d-n} : Z_{d-n}) = (X_{2n+1} : Z_{2n+1})$, and
$(X_{2d+1} : Z_{2d+1}) \leftarrow \text{UPDATE}(X_{d+n+1} : Z_{d+n+1})$.

Thus, the proposed approach can increase the speed of constant-time CSIDH implementations.

5. Experimental Results

To evaluate the efficiency of the proposed algorithm, experiments were conducted to demonstrate the computational costs for $\ell$-isogeny for several $\ell$ and the class group
evaluation in CSIDH. Our proposed algorithm is implemented in C by referring the source code by Bernstein et al. [11]. The code is available at https://github.com/KKodera/CSIDH_2ADDSKIP. The experiments compare the proposed algorithm with Meyer and Reith [5] and Bernstein et al. [11] in terms of required clock cycles. The following results were obtained on a computer with an Intel Core i7-8569U Coffee Lake processor running Debian 10.4 with Turbo Boost disabled.

The parameter \( n \) in the proposed algorithm is optimized for each \( \ell \) through experiments before comparisons. Each \( n \) is determined around \( n_{\text{M}} \) in Sect. 4.3.

5.1 Results for \( \ell \)-Isogenies Computation

Figure 2 shows the computational costs for \( \ell \)-isogeny computation for several \( \ell \) by following [11, Fig. 3]. The horizontal axis corresponds to degree \( \ell \) for 74 primes used in CSIDH-512, which comprises the first 73 smallest odd primes and 587. The vertical axis corresponds to the required clock cycles, which is obtained by a median across 15 experiments, divided by \( \ell + 2 \). The figure is displayed on a logarithmic scale. Moreover, the green, red, and blue dots represent the proposed algorithm, Bernstein et al.’s algorithm, and Meyer and Reith’s algorithm, respectively. Our proposed algorithm has the lowest number of clock cycles for \( 19 \leq \ell \leq 373 \). Furthermore, our proposed algorithm can reduce the computational cost by 12% compared to Meyer and Reith’s algorithm asymptotically as shown in Sect. 4.3.

5.2 Results for the Class Group Evaluation in CSIDH

Figures 3 and 4 show the computational costs for the class group evaluation in CSIDH-512 and CSIDH-1024, respectively. CSIDH-1024 involves \( \ell \)-isogenies for \( (\ell_1, \ldots, \ell_{130}) = (3, \ldots, 983) \), the first 129 smallest odd primes and 983. For both CSIDH-512 and CSIDH-1024, 15 experiments were conducted each for 35 types of secret keys. In these figures, the horizontal axis corresponds to secret keys sorted in the increasing order of median cost, and the vertical axis corresponds to clock cycles. The color of dots represents the same algorithm as above. None of these implementations are performed in constant time.

For CSIDH-512, the class group evaluation with our proposed algorithm has the lowest number of clock cycles. Our proposed algorithm and Bernstein et al.’s algorithm can reduce the computational cost by approximately 5.2% and 1.0%, respectively. For CSIDH-1024, our proposed algorithm is better than Meyer and Reith’s, but worse than Bernstein et al.’s. Our proposed algorithm and Bernstein et al.’s algorithm can reduce the computational cost by approximately 5.2% and 9.0%, respectively.

6. Conclusion

In this study, efficient algorithms for isogeny computation with odd degree \( \ell \) were investigated to speed up isogeny-based cryptography. We proposed the “2-ADD-Skip method” to reduce the number of points to be computed by addition formulae during isogeny computation. By exploring algorithms for isogeny computation utilizing the 2-ADD-Skip method, a reasonable algorithm was proposed. Furthermore, the computational cost of the proposed algorithm was analyzed in terms of the number of field arithmetic operations. As a result, our proposed algorithm performs well when it computes around \( \sqrt{\ell} \) points in the kernel. With those optimized parameters, our proposed algorithm can reduce the cost by approximately 12% for large \( \ell \) compared with Meyer and Reith’s algorithm [5]. Moreover, this study showed that our proposed algorithm can compute dummy isogeny with two extra additions; hence, it can also increase the speed of constant-time CSIDH implementations.

Our proposed algorithm was implemented in C, and the experiments were conducted to compare its performance with Meyer and Reith’s algorithm and Bernstein et al.’s algorithm [11]. Although our proposed algorithm is not asymptotically faster than Bernstein et al.’s algorithm, the results showed that it still achieved the lowest number of clock cycles for \( 19 \leq \ell \leq 373 \). Moreover, in terms of the class group evaluation in CSIDH-512 and CSIDH-1024, The results showed that the proposed algorithm could speed up the implementation with Meyer and Reith’s algorithm by approximately 5.2% in both cases. Moreover, the proposed algorithm had the best performance for isogeny computation for CSIDH-512 among all.
Acknowledgments

This study was supported by JSPS KAKENHI Grant Number JP19H10400, enPiT (Education Network for Practical Information Technologies) at MEXT, and Innovation Platform for Society 5.0 at MEXT.

References


Kenta Kodera received his B.S. and M.S. in Electrical, Electronic, and Information Engineering from Osaka University in 2017 and 2018, respectively. His current research activities focus on cryptography, including security analysis of elliptic curve cryptography and investigating efficient algorithms for isogeny-based cryptography.

Chen-Mou Cheng received his B.S. and M.S. in Electrical Engineering from National Taiwan University in 1996 and 1998, respectively, and his Ph.D. in Computer Science from Harvard University in 2007. He is currently an associate professor at the Graduate School of Natural Science and Technology, Kanazawa University. His main research area is in cryptographic hardware and embedded systems (CHES). Currently, his main research activities focus on the design and analysis of efficient algorithms to solve several important problems arising from cryptography as well as the development and implementation of these algorithms on massively parallel computers. These problems include solving systems of polynomial equations over finite fields, integer factorization, elliptic-curve discrete logarithm, and lattice reduction.
Atsuko Miyaji received the B.Sc., the M.Sc., and the Dr. Sci. degrees in mathematics from Osaka University in 1988, 1990, and 1997 respectively. She joined Panasonic Co., LTD from 1990 to 1998 and engaged in research and development for secure communication. She was an associate professor at the Japan Advanced Institute of Science and Technology (JAIST) in 1998. She joined the computer science department of the University of California, Davis from 2002 to 2003. She has been a professor at Japan Advanced Institute of Science and Technology (JAIST) since 2007. She has been a professor at Graduate School of Engineering, Osaka University since 2015. Her research interests include the application of number theory into cryptography and information security. She received Young Paper Award of SCIS'93 in 1993, Notable Invention Award of the Science and Technology Agency in 1997, the IPSJ Sakai Special Researcher Award in 2002, the Standardization Contribution Award in 2003, the AWARD for the contribution to CULTURE of SECURITY in 2007, the Director-General of Industrial Science and Technology Policy and Environment Bureau Award in 2007, DoCoMo Mobile Science Awards in 2008, Advanced Data Mining and Applications (ADMA 2010) Best Paper Award, Prizes for Science and Technology, the Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology, International Conference on Applications and Technologies in Information Security (ATIS 2016) Best Paper Award, the 16th IEEE Trustcom 2017 Best Paper Award, IEICE milestone certification in 2017, the 14th Asia Joint Conference on Information Security (AsiaJCIS 2019) Best Paper Award, and Information Security Applications - 20th International Conference (WISA 2020) Best Paper Gold Award. She is a member of the International Association for Cryptologic Research, the Institute of Electrical and Electronics Engineers, the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, and the Mathematical Society of Japan.