A faster way to the CSIDH

Michael Meyer[1,2] and Steffen Reith[1]

[1] Department of Computer Science, University of Applied Sciences Wiesbaden, Germany
[2] Department of Mathematics, University of Würzburg, Germany
{Michael.Meyer, Steffen.Reith}@hs-rm.de

Abstract. Recently Castryck, Lange, Martindale, Panny, and Renes published CSIDH, a new key exchange scheme using supersingular elliptic curve isogenies. Due to its small key sizes, and the possibility of a non-interactive and a static-static key exchange, CSIDH seems very interesting for practical applications. However, the performance is rather slow. Therefore, we employ some techniques to speed up the algorithms, mainly by restructuring the elliptic curve point multiplications and by using twisted Edwards curves in the isogeny image curve computations, yielding a speed-up factor of 1.33 in comparison to the implementation of Castryck et al. Furthermore, we suggest techniques for constant-time implementations.

1 Introduction

Isogeny-based cryptography. Isogeny-based cryptography is one of the current proposals for post-quantum cryptography. Already proposed in a talk (but not published) by Couveignes in 1997 [11] and independently rediscovered by Rostovtsev and Stolbunov in 2004 [22], a Diffie-Hellman-style key exchange based on isogenies between ordinary elliptic curves was designed (called CRS in the following). In 2010, Childs, Jao and Soukharev [8] showed, that this scheme can be attacked by solving an abelian hidden shift problem, for which subexponential quantum algorithms are known to exist.

Due to this, Jao and De Feo [16] considered the use of supersingular elliptic curves, and designed the new key exchange scheme SIDH (supersingular isogeny Diffie-Hellman), based on random walks in isogeny graphs for supersingular elliptic curves defined over fields $\mathbb{F}_{p^2}$. The performance of their scheme was improved by Costello, Longa, and Naehrig [10], yielding an important step towards practical deployment of SIDH, and also causing an increase of attention and research for isogeny-based cryptography. This led to the development of SIKE [15], an isogeny-based key encapsulation scheme, as entry for the NIST post-quantum cryptography competition [23], that aims for the standardization of post-quantum schemes in order to start the transition to the practical use of quantum-resistant primitives. The main advantage of SIKE comes from its key sizes. Among all the submitted key encapsulation schemes, it provides the

smallest public keys. However, the price for this is a rather bad performance. In comparison to the other competition entries, the running time of SIKE is slow[3].

Recently, De Feo, Kieffer, and Smith [14] published some new ideas for the optimization of the CRS scheme. Due to its commutative and non-interactive structure, it is still an interesting alternative to SIDH and SIKE. However, the performance is far from being practical. Therefore, Castryck, Lange, Martindale, Panny, and Renes [7] found that the optimizations, that De Feo, Kieffer, and Smith wanted to employ, work even better when adapting CRS to supersingular elliptic curves, i.e. working with supersingular elliptic curves over $\mathbb{F}_p$ rather than $\mathbb{F}_{p^2}$ like in SIDH. They obtain a non-interactive key exchange scheme with even smaller key sizes than in SIDH, called CSIDH (commutative SIDH, pronounced like "seaside"), that also allows static keys, since public keys can be validated, to detect active attacks. The performance is rather slow in comparison to SIDH and SIKE[4], which explains why it is an interesting and important task to optimize the running time of the scheme. However, we note that the security of the scheme and hence also the choice of parameters is still an open problem, which we will only briefly address in the next section.

Organization. In the following section, we give an introduction to CSIDH, mainly focusing on the implementer's point of view, and recall some aspects about Montgomery and twisted Edwards curves. We then introduce a way to restructure elliptic curve point multiplications in CSIDH, that allows a reduction of the computational effort. Thereafter, we review some methods to compute isogenies, i.e. point evaluations and computations of the image curves. In the first case, we employ an observation of Costello and Hisil [9] for a speed-up to the implementation of [7], whereas in the latter case, we exploit the well-known correspondence between Montgomery and twisted Edwards curves, to compute the image curves more efficiently. We give some implementation results according to our contributions, and give some remarks about constant-time implementations and bounds for their running time.

2 Preliminaries

2.1 CSIDH

Since our aim is to focus on implementations of CSIDH, we only give a very brief description of the mathematical background based on [7]. We recommend the lecture of [7] for a more detailed overview. We refer to [12] for additional information about isogenies and their cryptographic applications.

[3] In [15] it is stated that the best performance for SIKEp503 and SIKEp751 on a 3.4 GHz processor is 10.1 ms and 30.5 ms, respectively.

[4] The implementation results in [7] suggest that SIKE is about 10x faster than CSIDH at NIST security level 1. Note that SIKE uses a protected constant-time implementation, while the numbers for CSIDH are obtained from an unprotected non-constant-time implementation.

Mathematical background. First consider a quadratic number field k and an order $\mathcal{O} \subset k$. The ideal class group of $\mathcal{O}$ is defined as

$$\mathrm{cl}(\mathcal{O}) = I(\mathcal{O})/P(\mathcal{O}),$$

where $I(\mathcal{O})$ denotes the set of invertible fractional ideals and $P(\mathcal{O})$ the set of principal fractional ideals.

In the context of CRS and CSIDH, we use the group action of the ideal class group of an imaginary quadratic order $\mathcal{O}$ on ordinary and supersingular elliptic curves, respectively. In both cases the ideal class group $\mathrm{cl}(\mathcal{O})$ acts on $\mathcal{E}\ell\ell_p(\mathcal{O})$ via isogenies, where $\mathcal{E}\ell\ell_p(\mathcal{O})$ is the set of elliptic curves E defined over $\mathbb{F}_p$ with $\mathrm{End}_p(E) \cong \mathcal{O}$. By $\mathrm{End}_p(E)$ we denote the subring of the endomorphism ring $\mathrm{End}(E)$, that consists of endomorphisms defined over $\mathbb{F}_p$.

In CSIDH, a prime $p = 4 \cdot \ell_1 \cdot ... \cdot \ell_n - 1$ is chosen, where the ℓ_i are small distinct odd primes, and the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$, which is supersingular because $p \equiv 3 \pmod 4$. The supersingularity of E_0 (and of all curves that are isogenous to E_0) now guarantees the existence of elliptic curve points of order ℓ_i for all $i \in \{1, ..., n\}$.

The ideals $\ell_i\mathcal{O}$ split as $\ell_i\mathcal{O} = \mathfrak{l}_i\overline{\mathfrak{l}_i}$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$ and $\overline{\mathfrak{l}_i} = (\ell_i, \pi + 1)$ with the Frobenius endomorphism π. The kernel of the isogeny $\varphi_{\mathfrak{l}_i}$ then is the intersection of the kernels of the point multiplication $[\ell_i]$ and the endomorphism $\pi - 1$, i.e. a subgroup generated by a point P of order ℓ_i defined over $\mathbb{F}_p$. Analogously, the kernel of the isogeny $\varphi_{\overline{\mathfrak{l}_i}}$ is a subgroup generated by an order-ℓ_i point P defined over $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$. Hence, the computation of the action of an ideal class $\prod \mathfrak{l}_i^{e_i}$ by computing the action of the $\mathfrak{l}_i$ resp. $\overline{\mathfrak{l}_i}$ can be done by efficient isogeny formulae: We have to find order-ℓ_i points defined over $\mathbb{F}_p$ resp. $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$ and can apply efficient formulae such as Vélu-isogenies [24] or isogeny formulae for Montgomery curves like in [9] or [21]. By construction such points always exist. Ideal classes $\prod \mathfrak{l}_i^{e_i}$ can simply be represented by vectors $(e_1, ..., e_n)$.

Key exchange. As already observed by Couveignes in [11], the commutativity of the class group action allows for a Diffie-Hellman-style key exchange in the following way:

Alice chooses a secret ideal class $[\mathfrak{a}]$, represented by a vector $(e_1, ..., e_n)$, computes $E_A = [\mathfrak{a}] \cdot E_0$ via isogenies, and sends the result to Bob as her public key in terms of a curve parameter. Bob proceeds in the same way, chooses a secret $[\mathfrak{b}]$ and computes his public key $E_B = [\mathfrak{b}] \cdot E_0$. Then, because of the commutativity, both parties can compute the shared secret $[\mathfrak{a}] \cdot [\mathfrak{b}] \cdot E_0 = [\mathfrak{a}] \cdot E_B = [\mathfrak{b}] \cdot E_A$.

Security of the scheme. As for the CRS scheme, it is clear that the subexponential quantum attack from [8] also applies to CSIDH. However, Castryck

et al. give some estimations for parameter sets for different security levels [7]. More recently, shortly after CSIDH was published, more analysis of this attack has been done [3,4]. Since we are only focusing on efficient implementations throughout this work, we will not discuss these attacks here, and we only note that the appropriate choice of parameters is still an open problem, that requires further analysis. However, our improvements don't rely on a special choice of parameters, and are thus independent of the selected parameters.

2.2 Implementation

We follow the implementation accompanying [7] here[5].

Algorithm 1: Evaluating the class group action.

Input : $A \in \mathbb{F}_p$ and a list of integers $(e_1, ..., e_n)$.
Output: A' such that $[\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}]E_A = E_{A'}$.

1 **while** some $e_i \neq 0$ **do**
2 Sample a random $x \in \mathbb{F}_p$.
3 Set $s \leftarrow +1$ if $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$, else $s \leftarrow -1$.
4 Let $S = \{i \mid sign(e_i) = s\}$.
5 **if** $S = \emptyset$ **then**
6 Go to line 2.
7 $P = (x : 1)$, $k \leftarrow \prod_{i \in S} \ell_i$, $P \leftarrow [(p+1)/k]P$.
8 **foreach** $i \in S$ **do**
9 $K \leftarrow [k/\ell_i]P$.
10 **if** $K \neq \infty$ **then**
11 Compute a degree-ℓ_i isogeny $\varphi : E_A \rightarrow E_{A'}$ with $ker(\varphi) = \langle K \rangle$.
12 $A \leftarrow A'$, $P \leftarrow \varphi(P)$, $k \leftarrow k/\ell_i$, $e_i \leftarrow e_i - s$.

First, we define a prime number $p = 4 \cdot \ell_1 \cdot ... \cdot \ell_n - 1$ as above, where $\ell_1, ..., \ell_n$ are small distinct odd primes. Then we choose a supersingular curve E_0 over $\mathbb{F}_p$. Therefore we have $\#E_0 = p + 1$, which means that there are points of order ℓ_i for $i = 1, ..., n$ on E_0. Note that the factor 4 is needed to ensure that we can use Montgomery curves.

The private key contains n integers sampled from an interval $[-m, m]$, i.e. has the form $(e_1, ..., e_n)$. For each i the absolute value $|e_i|$ determines how many isogenies of degree ℓ_i are to be computed, while the sign of e_i states if we have to use points defined over $\mathbb{F}_p$ or $\mathbb{F}_{p^2} \backslash \mathbb{F}_p$ to generate their kernels.

For the computation of isogenies, we choose a random point P by sampling

[5] We refer to the version from 27.04.2018 throughout this work.

a random $x \in \mathbb{F}_p$, and check in which of the cases above this leads us by checking the minimal field of definition of the corresponding y-coordinate by a square root check. We then eliminate the possible unwanted factors in the order of P by multiplying it by $4 \cdot \prod_{j \notin S} \ell_j$, where S is defined as in algorithm 1.

After that, we iterate over the ℓ_i for $i \in S$, removing all remaining possible factors of the order except for ℓ_i of our point by multiplications, and check whether the resulting point K can be used as kernel generator for computing an ℓ_i-isogeny, i.e. if $K \neq \infty$. If so, we compute the isogeny and push P through. Then we go to the next prime and proceed in the same way. However, we don't have to consider the previous ℓ_i in the multiplication, since the isogeny evaluations of P already eliminate the respective factors from its order, or in the other case, the order of P did not contain the previous ℓ_i as factors in the first place.

We proceed in the same way, and sample new random points, until all of the required isogenies are computed. The resulting curve then forms the public key, or the shared secret, respectively. Note that the computational effort in algorithm 1 highly depends on the private key. Therefore, for the practical usage of CSIDH, it is important to transform this into a constant-time scheme without adding too much computational overhead.

Public keys can also be validated by checking for supersingularity: We can simply sample a random point P on the curve corresponding to the received public key. For each ℓ_i we compute $Q_i = [(p+1)/\ell_i]P$. For all i with $Q_i \neq \infty$, we compute $[\ell_i]Q_i$ and $d = \prod \ell_i$. If any of these $[\ell_i]Q_i \neq \infty$, the curve cannot be supersingular, since $\#E(\mathbb{F}_p) \nmid p+1$. If this is not the case, and $d > 4\sqrt{p}$, the curve must be supersingular, as can be seen from the Hasse interval and Lagrange's theorem (see [7]). Otherwise, the procedure can be repeated with a different point P. Following this approach, it is not possible to wrongly classify an ordinary curve as supersingular. Therefore, we can check if a public key has been honestly generated, and thus can prevent certain kinds of active attacks.

Choice of parameters. The following discussions and implementation results refer to the parameter set proposed in [7] for NIST's post-quantum security category I. They choose $p = 4 \cdot \ell_1 \cdot ... \cdot \ell_{74} - 1$, where $\ell_1, ...\ell_{73}$ are the 73 smallest distinct odd primes and $\ell_{74} = 587$. The elements of the private keys $(e_1, ..., e_{74})$ are chosen from the interval $[-5, 5]$. This parameter set leads to public key lengths of 64 bytes. As mentioned before, the appropriate choice of parameters is still an open problem, so the analysis of the actual security level of this parameter set is left for future work.

2.3 Montgomery curves

Montgomery curves are given by an equation over a field k with $\mathrm{char}(k) > 2$ of the form

$$E_{a,b} : by^2 = x^3 + ax^2 + x,$$

where $a \in k\backslash\{-2, 2\}$ and $b \in k\backslash\{0\}$. To avoid inversions during point additions and doublings, projective coordinates can be used. Furthermore, the efficient arithmetic given by Montgomery in [19] allows for dropping the Y-coordinate and still performing XZ-only point doublings and differential additions, that require the knowledge of the XZ-coordinates of P, Q, and $P - Q$ in order to compute $P + Q$.

In [10], Costello et al. propose to projectivize not only the point coordinates, but also the curve parameters. Instead of a Montgomery curve of the form given above, we work with an equation of the form

$$E_{(A:B:C)} : By^2 = Cx^3 + Ax^2 + Cx,$$

where $(A : B : C) \in \mathbb{P}^2(k)$, such that $a = A/C$ and $b = B/C$ for the corresponding curve $E_{a,b}$. However, in isogeny-based schemes it suffices to work with $(A : C) \in \mathbb{P}^1(k)$ in the projective model, since neither the Montgomery curve arithmetic, nor the isogeny computations require the coefficients b or B, respectively. In general a doubling then costs $4\mathbf{M} + 2\mathbf{S} + 8\mathbf{a}$, while a differential addition costs $4\mathbf{M} + 2\mathbf{S} + 6\mathbf{a}$. As usual, we denote field multiplications by $\mathbf{M}$, field squarings by $\mathbf{S}$, and field additions or subtractions by $\mathbf{a}$.

2.4 Twisted Edwards curves

Introduced by Bernstein et al. in [1], twisted Edwards curves over k with $\mathrm{char}(k) > 2$ are given by equations of the form

$$E_{a,d} : aX^2 + Y^2 = 1 + dX^2Y^2,$$

with $ad \neq 0$, $d \neq 1$, and $a \neq d$. For $a = 1$ the twisted Edwards curve $E_{1,d} = E_d$ is called Edwards curve, originally proposed by Edwards in [13]. As in the Montgomery case, projective coordinates can be used in order to avoid inversions during additions and doublings. Note that in the Edwards case there are different models for doing this, as described in [2].

Similar to the XZ-only Montgomery curve arithmetic, Castryck, Galbraith, and Farashahi introduced a YZ-only doubling formula for twisted Edwards curves in [6] with a cost of $4\mathbf{M} + 5\mathbf{S}$. A formula for YZ-only differential addition of twisted Edwards curve points of odd order is derived in [17], using $6\mathbf{M} + 3\mathbf{S}$ in the projective case. Due to the fact that these operations are in general more expensive than the respective operations on the Montgomery curve, isogeny-based schemes usually use Montgomery curves (see [5,18] for a comparison to twisted Edwards curve point arithmetic in SIDH). However, in the following twisted Edwards curves are shown to be advantageous for the computation of isogenies.

3 Elliptic curve point multiplications

Define $\alpha = \frac{p+1}{4} = \ell_1 \cdot \ell_2 \cdot ... \cdot \ell_n$. For the sake of simplicity, we consider a private key $(e_1, ..., e_n)$, where all $e_i > 0$, or all $e_i < 0$. We will return to the general

case later on. The algorithm used by Castryck et al. then samples a random point P on the current curve E_0, checks if its y-coordinate is defined over the corresponding field $\mathbb{F}_p$ or $\mathbb{F}_{p^2}\backslash\mathbb{F}_p$, and if so, sets $P_0 = [4]P$ in order to remove the possible factor 4 from its order. Then they compute $K_0 = [\frac{\alpha}{\ell_1}]P_0$. If $K_0 = \infty$, the order of P does not contain the factor ℓ_1. We cannot use it to compute an isogeny of degree ℓ_1 and set $P_1 = P_0$ and $E_1 = E_0$. If however $K_0 \neq \infty$, then K_0 must have order ℓ_1 and can be used as generator of the kernel of an isogeny of degree ℓ_1, mapping to a curve E_1. In this case, we pull P_0 through the isogeny and obtain a point $P_1 \in E_1$. Note that this implies, that the order of P_1 does not contain the factor ℓ_1. Therefore, for checking if we can use P_1 to compute an isogeny of degree ℓ_2, it suffices to compute $K_1 = [\frac{\alpha}{\ell 1 \cdot \ell 2}]P_1$ and proceed as before. Following this approach, the required factor for the scalar multiplication of P_j reduces at each step, until only the factor ℓ_n remains at the last step of the loop.

Castryck et al. go through the primes in ascending order in their implementation, starting with small degree isogenies. However, we found it advantageous to change the direction of the loop, i.e. go through the primes in descending order. By doing this, we can eliminate the larger factors of $p+1$ first, and therefore end up with multiplications by significantly smaller factors as we proceed through the loop. Note that as soon as one isogeny degree is done, i.e. $|e_i|$ isogenies of degree ℓ_i were already computed, we include this factor in the first multiplication to compute P_0, making sure that the order of P_0 is not divided by ℓ_i. We can then ignore the factor ℓ_i in the loop, which slightly reduces the advantage of our approach every time this occurs. However, we note that our approach is still faster, as long as at least two factors are left in the loop.

Figure 1 shows the effect of our approach, compared to the implementation of [7]. Note that per bit of the factor of an elliptic curve point multiplication one step in the Montgomery ladder is carried out, i.e. one combined doubling and addition. Therefore, in the first loop, at each multiplication the computational effort is reduced by δ_i times the cost of a ladder step, where δ_i is the difference between the two plots for a given $i < n$, and hence $\delta_i \cdot (8\mathbf{M} + 4\mathbf{S} + 8\mathbf{a})$. As discussed before, the number of saved operations reduces in the following loops.

In the general case, our assumption that all elements of the private key share the same sign obviously does not hold. However, the described effect will translate at a lower scale to both of the somewhat distinct computations for the sets $S_+ = \{\ell_i \mid e_i > 0\}$ and $S_- = \{\ell_i \mid e_i < 0\}$ corresponding to the private key $(e_1, ..., e_n)$. Indeed, when plotting the bitlengths of the factors in the respective first loops in such cases, this leads to a similar result as in Figure 1, only at a lower scale.

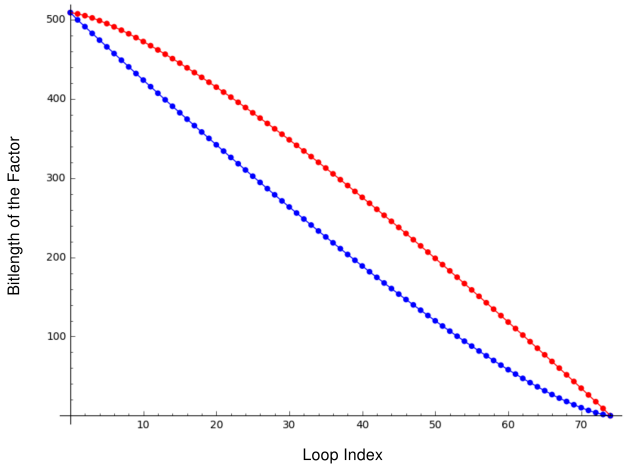

Fig. 1. Bitlengths of factors during the first loop, when all e_i have the same sign. The red line follows the algorithm of [7], the blue line follows our described approach.

4 Isogeny Computations

The algorithm of [7] uses isogeny formulae for Montgomery curves by Costello and Hisil [9] and Renes [21]. We will treat point evaluations and computations of coefficients of image curves separately. First, we will state the isogeny formulae of [9], which can be used for the computation of isogenies in CSIDH.

Let K be a point of order $\ell = 2d+1$ on a Montgomery curve $E : y^2 = x^3+ax^2+x$. Then we can compute the coordinate map of the unique (up to compositions by isomorphisms) ℓ-isogeny $\varphi : E \to E'$ with $ker(\varphi) = \langle K \rangle$ by

$$\varphi : (x, y) \mapsto (f(x), y \cdot f'(x)),$$

where

$$f(x) = x \cdot \prod_{i=1}^{d} \left(\frac{x \cdot x_{[i]K} - 1}{x - x_{[i]K}} \right)^2,$$

and $f'(x)$ is its derivative. The curve parameters a' and b' of E' can be computed by $a' = (6\sigma - 6\tilde{\sigma} + a) \cdot \pi^2$ and $b' = b \cdot \pi^2$, where we define $\sigma = \sum_{i=1}^{d} x_{[i]K}$, $\tilde{\sigma} = \sum_{i=1}^{d} 1/x_{[i]K}$, and $\pi = \prod_{i=1}^{d} x_{[i]K}$.

Note that the representation of $f(x)$ makes use of the fact that $x_{[i]K} = x_{[\ell-i]K}$ for all $k \in \{1, ..., (\ell-1)/2\}$.

4.1 Point evaluations

Since we work with XZ-only projective Montgomery coordinates, we have to represent $f(x)$ projectively. This is done in [9] by writing $(X_i : Z_i) = (x_{[i]K} : 1)$

for $i = 1, ..., d$, $(X : Z) = (x_P : 1)$ for the point P, at which the isogeny should be evaluated, and $(X' : Z')$ for the result. Then

$$X' = X \cdot \left(\prod_{i=1}^{d} (X \cdot X_i - Z_i \cdot Z) \right)^2, \quad \text{and}$$
$$Z' = Z \cdot \left(\prod_{i=1}^{d} (X \cdot Z_i - X_i \cdot Z) \right)^2.$$

In the implementation of [7], this is used directly by going through the $(X_i : Z_i)$ for $i = 1, ..., d$ and computing the pairs $(X \cdot X_i - Z_i \cdot Z)$ and $(X \cdot Z_i - X_i \cdot Z)$ at a cost of $4\mathbf{M} + 2\mathbf{a}$ per step. However, we can also use the observation by Costello and Hisil in [9] to reduce the cost to $2\mathbf{M} + 4\mathbf{a}$ per step by

$$X' = X \cdot \left(\prod_{i=1}^{d} \Big[(X - Z)(X_i + Z_i) + (X + Z)(X_i - Z_i) \Big] \right)^2, \quad \text{and}$$
$$Z' = Z \cdot \left(\prod_{i=1}^{d} \Big[(X - Z)(X_i + Z_i) - (X + Z)(X_i - Z_i) \Big] \right)^2,$$

assuming that $X + Z$ and $X - Z$ are precomputed, and hence save $d \cdot (2\mathbf{M} - 2\mathbf{a})$ per isogeny evaluation.

4.2 Computing the image curve

An efficient computation of the image curve parameters is not as straightforward as for the point evaluations. This is due to the fact that the required parameters σ and $\tilde{\sigma}$ consist of sums of fractions. Therefore, Costello and Hisil give two different approaches to compute the isogenous curve [9].

The first approach uses the fact that the projective parameters $(a' : 1) = (A' : C')$ of the isogenous curve E' can be recovered from the knowledge of the three 2-torsion points of E'. Therefore, it is possible to recover the required curve parameters of E' by computing the 2-torsion points of E and pushing one of these points through the odd-degree isogeny, which preserves its order on the image curve. However, in contrast to SIDH, we only work over the field $\mathbb{F}_p$ instead of $\mathbb{F}_{p^2}$, while the required points of order 2 are not defined over $\mathbb{F}_p$ in the CSIDH setting.

Their second approach uses the fact that the curve parameters can be recovered from the knowledge of the x-coordinates of two points on the curve, and their difference. While these points are typically available in SIDH during the key generation phase, this is not the case for CSIDH, where we only want to compute the isogenous curve and evaluate one point.

In [7], Castryck et al. compute the image curve by defining $c_j \in \mathbb{F}_p$ such that

$$\prod_{i=1}^{\ell-1}(Z_i w + X_i) = \sum_{j=0}^{\ell-1} c_j w^j$$

as polynomials in w. Then they observe that

$$(A' : C') = (\hat{\pi}(a - 3\hat{\sigma}) : 1) = (ac_0c_{\ell-1} - 3(c_0c_{\ell-2} - c_1c_{\ell-1}) : c_{\ell-1}^2),$$

following the formulae and notation from Renes [21], where

$$\hat{\pi} = \prod_{i=0}^{\ell-1} x_{[i]K}, \quad \text{and} \quad \hat{\sigma} = \sum_{i=0}^{\ell-1} \Big(x_{[i]K} - \frac{1}{x_{[i]K}}\Big).$$

In their implementation, this is computed iteratively, going through the $(X_i : Z_i)$ for $i = 2, ..., d$, updating the required values at a cost of $6\mathbf{M} + 2\mathbf{a}$ per step. The final computations after that take further $8\mathbf{M} + 3\mathbf{S} + 6\mathbf{a}$ to compute the curve parameters $(A' : C')$.

Using twisted Edwards curves for the image curve computation. Our idea to speed up this computation exploits the known correspondence between Montgomery and twisted Edwards curves. Given a Montgomery curve $E_{A,B} : Bv^2 = u^3 + Au^2 + u$, we can switch to a birationally equivalent twisted Edwards curve $E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2$, where

$$A = \frac{2(a+d)}{a-d} \quad \text{and} \quad B = \frac{4}{a-d},$$

by the coordinate map

$$(u, v) \mapsto (x, y) = \left(\frac{u}{v}, \frac{u-1}{u+1}\right).$$

and back by its inverse

$$(x, y) \mapsto (u, v) = \left(\frac{1+y}{1-y}, \frac{1+y}{(1-y)x}\right)$$

In [18] it is shown how to switch to and from twisted Edwards curves in the SIDH setting, which also applies to CSIDH, where Montgomery XZ-only coordinates and projective curve parameters $(A : C)$ are used, ignoring the Montgomery parameter b. Following this and [6], a Montgomery point $(X^M : Z^M)$ can be transformed to the corresponding Edwards YZ-coordinates $(Y^E : Z^E)$ by the map

$$(X^M : Z^M) \mapsto (Y^E : Z^E) = (X^M - Z^M : X^M + Z^M),$$

and the Montgomery parameters $(A : C)$ to the corresponding twisted Edwards parameters (a_E, d_E) by

$$a_E = A + 2C \quad \text{and} \quad d_E = A - 2C.$$

As this allows us to switch efficiently between Montgomery and twisted Edwards curves in CSIDH at a cost of $3\mathbf{a}$ for the curve parameters and $2\mathbf{a}$ for point coordinates, we may as well use isogeny formulae for twisted Edwards curves. Therefore, we state the formulae given by Moody and Shumow in [20].

Let K be a point of order $\ell = 2d+1$ on a twisted Edwards curve $E : a_E x^2 + y^2 = 1 + d_E x^2 y^2$. Then we can compute the coordinate map of the unique (up to compositions by isomorphisms) ℓ-isogeny $\varphi : E \to E'$ with $ker(\varphi) = \langle K \rangle$ by

$$\varphi(P) = \left(\prod_{Q \in \langle K \rangle} \frac{x_{P+Q}}{y_Q}, \prod_{Q \in \langle K \rangle} \frac{y_{P+Q}}{y_Q} \right).$$

The curve E' is defined by the parameters $a'_E = a_E^\ell$ and $d'_E = \pi_y^8 d_E^\ell$, where $\pi_y = \prod_{i=1}^{d} y_{[i]K}$.

Since the coordinate map is not as simple to compute as for Montgomery curves, we are only interested in the computation of the image curve parameters. Writing $(Y_i^E : Z_i^E)$ for the projective coordinates of $[i]K$ for $i = 1, ..., d$, we can transform the formulae from above to the projective case by

$$a'_E = a_E^\ell \cdot \pi_Z^8, \quad \text{and} \quad d'_E = d_E^\ell \cdot \pi_Y^8,$$

where $\pi_Y = \prod_{i=1}^{d} Y_i^E$, and $\pi_Z = \prod_{i=1}^{d} Z_i^E$.

We can therefore use these formulae to compute the curve parameters of the image curves in CSIDH, by switching to twisted Edwards coordinates and points, and switch back after the computations by $(A' : C') = (2(a'_E + d'_E) : a'_E - d'_E)$, again at a cost of $3\mathbf{a}$.

Note that the parameters a'_E and d'_E can be computed efficiently: While going through the $(X_i : Z_i)$ on the Montgomery curve for $i = 1, ..., d$, we can compute the corresponding Edwards coordinates $(Y_i^E : Z_i^E)$ at a cost of $2\mathbf{a}$. However, the required sums and differences already occur at the point evaluation part, and hence do not add any computational cost at all. We can then compute π_Y and π_Z iteratively by $1\mathbf{M}$ each per step. Compared to the algorithm of [7], this saves $4\mathbf{M} + 2\mathbf{a}$ per step. Furthermore, we have to compute π_Y^8 and π_Z^8 by three squarings each, and a_E^ℓ and d_E^ℓ, which can be done efficiently, e.g. by a square-and-multiply approach. We further note that the latter computation does not require any values generated during the loop through the $(X_i : Z_i)$. This means that especially hardware architectures that allow for parallel computations would benefit from this, since the computation of a_E^ℓ and d_E^ℓ can be done in parallel to the loop through the $(X_i : Z_i)$.

Figure 2 compares the costs of a combined image curve computation and point evaluation for different prime degrees, where the red line arises from using the Montgomery isogenies from [7], including the optimizations from [9], and the

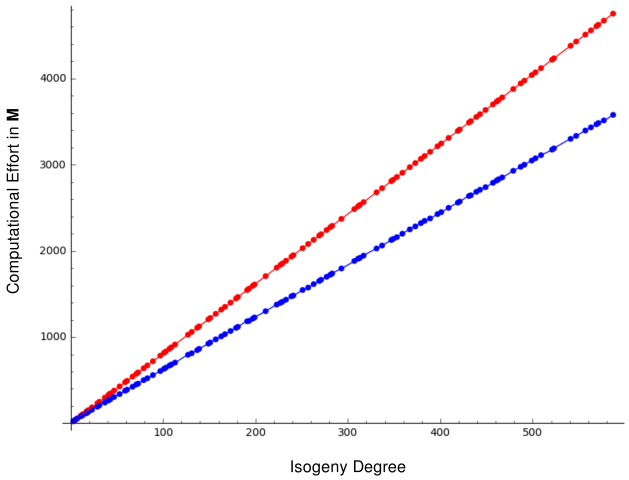

Fig. 2. Cost of different prime-degree isogeny computations. The red line uses Montgomery point evaluations from [9] and image curve computations from [7], while the blue line uses the same Montgomery point evaluations and twisted Edwards image curve computations.

blue line from using our approach utilizing twisted Edwards curves to compute the isogenous curve. The cost is measured in field multiplications, assuming that $\mathbf{S} = 0.8\mathbf{M}$ and $20\mathbf{a} = \mathbf{M}$. In this case, we computationally derive a reduction of the costs by approximately 25% for the largest primes in the current parameter set. We note that different choices for the field operation ratios don't make a big difference, since the main difference between the approaches lies in the number of multiplications. To obtain the cost of the computations of a_E^ℓ and d_E^ℓ, we used a square-and-multiply approach. Since the exponents ℓ_i are small fixed numbers, it is also possible to precompute the optimal addition chains, and therefore save some computational effort compared to square-and-multiply. However, we found that even for the biggest ℓ_i from the current parameter set, this saves at most four multiplications. Hence, the benefit of this is rather small compared to the increased length of the code and the more complicated implementation.

Note that for $\ell \leq 5$, our approach is slightly more expensive than the Montgomery approach. Therefore, in this case, the Montgomery approach can be used. However, the benefit of this is rather small compared to the total computational effort, so it might be better to stick with our approach for all ℓ_i in order to keep the implementation simple.

It is further noted in [7], that for a fixed prime ℓ one could reduce the computational effort by finding an appropriate representative of the isogeny modulo (a factor of) the ℓ-division polynomial ψ_ℓ, as done in [10] for 3- and 4-isogenies.

However, every required isogeny degree would have to be implemented separately, resulting in a much longer code.

5 Implementation results

As a proof of concept and for measuring the efficiency of our work, we took the mentioned implementation of Castryck et al. accompanying [7] as reference, and added our optimizations. The implementation is written in C and uses $\mathbb{F}_p$-arithmetic in assembly. The parameters in use are the ones described in section 2.2. The validation of keys is not included in the following discussion.

The first optimization is the precomputation of the curve parameters $(A + 2C : 4C)$ each time before entering the Montgomery ladder, as also done in SIDH [10]. This only saves a few additions per ladder step, but in total leads to a reduction of the computational effort by approximately 2%.

The other optimizations are as described above: One comes from rearranging the factors in the class group action evaluation algorithm, and the other one from more efficient isogeny computations by using the point evaluation from [9] and our twisted Edwards approach for the image curve computations.

Table 1. Performance comparison of the class group action evaluation in CSIDH with different optimizations applied. All timings are given in 10^6 clock cycles and were measured on an Intel Core i7-6500 Skylake processor running Ubuntu 16.04 LTS, averaged over 10 000 runs.

	Clock Cycles $\times 10^6$	Acceleration Factor
Castryck et al. [7][6]	138.6	-
Precomputation of $(A + 2C : 4C)$	135.7	1.021
Rearranging Factors	126.5	1.096
Isogeny Optimization	118.2	1.173
Combination of all Optimizations	103.9	1.334

Table 1 lists the influence of the different optimizations on the overall performance. In the respective implementations, only the mentioned optimization was used, leaving the rest as in the reference implementation from [7]. For the last line, we combined all the described optimizations and therefore reduced the total computational effort by 25%, yielding a speed-up factor of 1.33. The latter implementation is available at `https://zenon.cs.hs-rm.de/pqcrypto/faster-csidh`.

6 On constant-time implementations

As mentioned in the sections before, the discussed implementations do not include any protection from side-channel attacks. In particular, the running time depends on the private key, which corresponds to the number of isogenies, that have to be computed. Therefore, the first step to prevent simple timing attacks is a constant-time implementation.

One possibility to reach this is to fix the number of isogenies to the maximum for each degree, and use only as many of them as specified in the private key. However, in addition to lots of useless computational effort, this means that after each dummy isogeny, another multiplication by its degree ℓ_i is necessary for the point P, since the algorithm uses the fact, that by pushing P through the ℓ_i-isogeny, the order of the resulting point will not include the factor ℓ_i. The additionally required multiplications would then allow for a new timing attack, since many extra multiplications mean that many dummy isogenies were computed. This could be prevented by using constant-time ladders for the preparation of each kernel point. This however is undesirable, since it would further blow up the running time.

A possible tool for the design of a more optimized constant-time implementation could be specially tailored dummy isogenies, that, instead of computing an ℓ_i-isogeny and pushing the point P through, simply compute $[\ell_i]P$ and leaves the curve parameters unchanged. This is especially easy, since the ℓ_i-isogeny algorithm requires to compute all $[j]K$ for $j \in \{2, ..., (\ell_i - 1)/2\}$. Therefore, by replacing K with P and by two further differential additions, we can compute $[\ell_i]P$, and hence don't have to perform more multiplications to compensate for the dummy isogenies. Furthermore, the dummy isogenies can be designed to have the exact same number of field operations as the real isogeny computations.

Running time. First, we note that this discussion refers to the case, that an implementation that follows algorithm 1 shall be transformed to a constant running-time. If the structure of algorithm 1 is changed, the results may vary accordingly.

It is obvious that the running time of a constant-time implementation must be at least as high as the highest possible running time of the fastest non-constant implementation. At first glance, this seems to be twice the average running time of the non-constant implementation, when we fix all elements of the private key to have the maximum absolute value. For a closer investigation, we consider the parameter set from above, and the private keys $e = (5, -5, 5, -5, ..., 5, -5)$ and $e' = (5, 5, ..., 5)$.

When comparing the performance for the private key e to the average case, our experiments suggest that indeed the running time roughly doubles. In fact,

the computational cost for isogenies doubles, while the factor for point multiplications is slightly higher. One reason for this is that if we have to compute more isogenies, on average there will be more cases in which our randomly chosen points cannot be used to compute isogenies of certain degrees. Therefore, more points have to be chosen and their order checked by multiplications.

Now we want to compare the performance for the keys e and e'. For the sake of simplicity, assume that we can choose full order points, that allow for the computation of isogenies of all required degrees. Consider that we first want to compute one isogeny of each degree. For e', the bitlengths of the factors for the point multiplications are the ones shown in figure 1. After one loop, we have computed one isogeny of each required degree. For the key e, we have to perform one loop each for positive and negative key elements. When doing this and counting the bitlengths of the factors for the point multiplications, after computing one isogeny of each degree, we end up with only 0.54 times the sum of the bitlengths for e', already including the additional required multiplications from line 7 of algorithm 1. Therefore, since we simply perform five such rounds for e and e', the total computational effort for point multiplications for e' is 1.86 times as high as for e. When considering also isogeny computations, our experiments suggest that the total running time for the key e' is 1.49 times the running time for e. Therefore, we conclude that the running time of a constant-time implementation must be at least 2.98 times the running time of average-case measurements of non-constant implementations such as in section 5. In practice, our experiments again suggest a higher factor for the running time for the key e', namely 3.07, for the same reasons as explained above[7].

However, we note that more careful analysis is required for an optimized constant-time implementation, and our proposal of dummy isogenies is merely a tool, that could possibly be used to design such an implementation, which we leave for future work.

7 Conclusion and future work

Although we gained a speed-up factor of 1.33 in our CSIDH implementation, it is still considerably slower than e.g. SIDH. Therefore, further research in that direction is necessary, to make the practical deployment of the scheme more attractive. In particular, side-channel protection, such as constant-time implementations, is required for that aim.

As mentioned before, also the security of CSIDH still requires some more detailed analysis on the implication of new attacks for specific parameter choices.

[7] We measured 318.9 $\times 10^6$ clock cycles for the running time for the key e' in the setting from table 1, using the optimized implementation.

Acknowledgements. This work was partially supported by Elektrobit Automotive, Erlangen, Germany. We thank Fabio Campos, Marc Stöttinger, and the anonymous reviewers for their helpful and valuable comments.

References

1. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards Curves. In: Vaudenay, S. (ed.) Progress in Cryptology - AFRICACRYPT 2008, pp. 389–405. Lecture Notes in Computer Science, 5023, Springer (2008)
2. Bernstein, D.J., Birkner, P., Lange, T., Peters, C.: ECM Using Edwards Curves. Mathematics of Computation **82**(282), 1139–1179 (2013)
3. Biasse, J.F., Jacobson Jr, M.J., Iezzi, A.: A note on the security of CSIDH. arXiv preprint arXiv:1806.03656 (2018), `https://arxiv.org/abs/1806.03656`
4. Bonnetain, X., Schrottenloher, A.: Quantum Security Analysis of CSIDH and Ordinary Isogeny-based Schemes. Cryptology ePrint Archive, Report 2018/537 (2018), `https://eprint.iacr.org/2018/537`
5. Bos, J.W., Friedberger, S.: Arithmetic Considerations for Isogeny Based Cryptography. Cryptology ePrint Archive, Report 2018/376 (2018), `https://eprint.iacr.org/2018/376`
6. Castryck, W., Galbraith, S., Farashahi, R.R.: Efficient arithmetic on elliptic curves using a mixed Edwards-Montgomery representation. Cryptology ePrint Archive, Report 2008/218 (2008), `http://eprint.iacr.org/2008/218`
7. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action (2018), `https://eprint.iacr.org/2018/383.pdf`
8. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology **8**(1), 1–29 (2014)
9. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology ASIACRYPT 2017. Lecture Notes in Computer Science, vol 10624, pp. 303–329. Springer, Cham (2017)
10. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny Diffie-Hellman. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016, pp. 572–601. Lecture Notes in Computer Science, 9814, Springer (2016)
11. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), `https://eprint.iacr.org/2006/291`
12. De Feo, L.: Mathematics of Isogeny Based Cryptography. Notes from a summer school on Mathematics for Post-quantum cryptography (2017), `http://defeo.lu/ema2017/poly.pdf`
13. Edwards, H.M.: A normal form for elliptic curves. Bulletin of the American Mathematical Society **44**, 393–422 (2007)
14. Feo, L.D., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. Cryptology ePrint Archive, Report 2018/485 (2018), `https://eprint.iacr.org/2018/485`
15. Jao, D., et al.: Supersingular isogeny key encapsulation, Round 1 submission, NIST Post-Quantum Cryptography Standardization (2017)
16. Jao, D., De Feo, L., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014)

17. Marin, L.: Differential Elliptic Point Addition in Twisted Edwards Curves. In: 2013 27th International Conference on Advanced Information Networking and Applications Workshops, pp. 1337–1342 (2013)
18. Meyer, M., Reith, S., Campos, F.: On hybrid SIDH schemes using Edwards and Montgomery curve arithmetic. Cryptology ePrint Archive, Report 2017/1213 (2017), https://eprint.iacr.org/2017/1213
19. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation **48**(177), 243–264 (1987)
20. Moody, D., Shumow, D.: Analogues of Vélu's Formulas for Isogenies on Alternate Models of Elliptic Curves. Math. Comput. **85**(300), 1929–1951 (2016)
21. Renes, J.: Computing isogenies between Montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) PQCrypto 2018, 9th International Conference on Post-Quantum Cryptography. pp. 229–247. Springer (2018)
22. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), http://eprint.iacr.org/2006/145
23. The National Institute of Standards and Technology (NIST): Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016)
24. Vélu, J.: Isogénies entre courbes elliptiques. C.R. Acad. Sc. Paris, Série A. **271**, 238–241 (1971)