# Fully Projective Radical Isogenies in Constant-Time

Jesús-Javier Chi-Domínguez[1] and Krijn Reijnders[2(✉)]

[1] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi,
United Arab Emirates
`jesus.dominguez@tii.ae`
[2] Radboud University, Nijmegen, The Netherlands
`krijn@cs.ru.nl`

**Abstract.** At PQCrypto-2020, Castryck and Decru proposed CSURF (CSIDH on the surface) as an improvement to the CSIDH protocol. Soon after that, at Asiacrypt-2020, together with Vercauteren they introduced radical isogenies as a further improvement. The main improvement in these works is that both CSURF and radical isogenies require only one torsion point to initiate a chain of isogenies, in comparison to Vélu isogenies which require a torsion point per isogeny. Both works were implemented using non-constant-time techniques, however, in a realistic scenario, a constant-time implementation is necessary to mitigate risks of timing attacks. The analysis of constant-time CSURF and radical isogenies was left as an open problem by Castryck, Decru, and Vercauteren. In this work we analyze this problem. A straightforward constant-time implementation of CSURF and radical isogenies encounters too many issues to be cost effective, but we resolve some of these issues with new optimization techniques. We introduce projective radical isogenies to save costly inversions and present a hybrid strategy for integration of radical isogenies in CSIDH implementations. These improvements make radical isogenies almost twice as efficient in constant-time, in terms of finite field multiplications. Using these improvements, we then measure the algorithmic performance in a benchmark of CSIDH, CSURF and CRADS (an implementation using radical isogenies) for different prime sizes. Our implementation provides a more accurate comparison between CSIDH, CSURF and CRADS than the original benchmarks, by using state-of-the-art techniques for all three implementations. Our experiments illustrate that the speed-up of constant-time CSURF-512 with radical isogenies is reduced to about 3% in comparison to the fastest state-of-the-art constant-time CSIDH-512 implementation. The performance is worse for larger primes, as radical isogenies scale worse than Vélu isogenies.

**Keywords:** Isogeny-based cryptography · CSIDH · Constant-time implementation · CSURF · Radical isogenies

# 1   Introduction

The first proposal of an isogeny-based Diffie-Hellman key exchange was done by Couveignes [13] and centered on the action of an ideal class group on a set of ordinary elliptic curves. Later Rostovtsev and Stolbunov [23,24] independently rediscovered it and recognized its potential as a possible post-quantum candidate. In the last decade, isogeny-based key exchange developed further, notably with SIDH in [2,14,16]. In Asiacrypt 2018, Castryck, Lange, Martindale, Panny, and Renes introduced CSIDH (a *non-interactive* key exchange) as a reformulation of the Couveignes-Rostovtsev-Stolbunov system using supersingular curves defined over a prime field [9]. With the hope to improve the performance of CSIDH, Castryck and Decru proposed CSURF, which exploits 2-isogenies [7] on the surface of the isogeny graph. Later on, Castryck, Decru, and Vercauteren in Asiacrypt 2020 expanded on the ideas in CSURF to construct isogenies with small odd degree based on radical computations ($N$-th roots) [8]. Using radical isogenies, they claimed a speed-up of about 19% over CSIDH-512, however both of the implementations in [7,8] focus on non-constant-time instantiations. In particular, Castryck, Decru, and Vercauteren left the analysis of a constant-time implementation of CSURF and radical isogenies as an open problem. A constant-time algorithm refers to an algorithm whose running time is independent of (or uncorrelated with) the secret input. This implies the variability in the running time depends on randomness and not on the leakage of information on secret values.

Dealing with constant-time implementations of CSIDH (and CSURF) can be tricky as there are multiple approaches, such as using dummy isogenies or a dummy-free approach. The first constant-time CSIDH instantiation is the procedure using dummy isogenies proposed by Meyer, Campos, and Reith in [18], later improved by Onuki *et al.* in [20]. Subsequently, Cervantes-Vázquez *et al.* proposed a dummy-free variant of CSIDH [10], and more recently, Banegas *et al.* presented CTIDH [3]. This covers the literature that we are aware of.

The general idea to make CSIDH implementations run in constant-time is to perform a fixed number $m$ of isogenies of a certain degree $\ell_i$, independent of the secret key $e_i$. For example, take the CSIDH-512 prime $p = 4 \cdot \prod_{i=1}^{74} \ell_i - 1$, where $\ell_1$ up to $\ell_{73}$ are the smallest 73 odd prime numbers and $\ell_{74} = 587$. Let $E/\mathbb{F}_p \colon y^2 = x^3 + Ax^2 + x$ be a supersingular Montgomery curve with $(p+1)$ rational points. Assuming we require exactly $m = 5$ isogenies per $\ell_i$, then our key space corresponds with the integer exponent[1] vectors $(e_1, \ldots, e_{74}) \in [\![-m \mathinner{..} m]\!]^{74}$. A dummy-based variant of constant-time CSIDH performs $|e_i|$ secret $\ell_i$-isogenies and then proceeds by performing $(m - |e_i|)$ dummy-isogenies. The $\ell_i$-isogeny kernel belongs to either $E[\pi - 1]$ or $E[\pi + 1]$, which is determined by the sign of $e_i$. A dummy-free variant (which prevents e.g. fault injection attacks) does not perform the $(m - |e_i|)$ dummy-isogeny constructions, but instead requires $e_i$ to have the same parity as $m$. It then alternates between using kernels in

---

[1] The word *exponent* comes from the associated group action, see Sect. 2.2.

$E[\pi - 1]$ and $E[\pi + 1]$ in such a way that one effectively *applies* $e_i$ isogenies while *performing m* isogenies.

The experiments presented in [8] suggest a speed-up of about 19% when using radical isogenies instead of Vélu's formulas (for a prime of 512 bits). As mentioned above, these experiments focused on a non-constant-time Magma implementation for both the group-action evaluation and the chain of radical isogenies. More specifically, the Magma-code implementation of [8] performs field inversions in variable time depending on the input. Furthermore, the implementation computes exactly $|e_i|$ $\ell_i$-radical isogenies, where $e_i \in [\![-m_i \,..\, m_i]\!]$ is a secret exponent of the private key (for instance when $e_i = 0$ the group action is trivial). Clearly, when measuring random non-constant time instances of CSURF or radical isogenies the average number of $\ell_i$-radical isogenies to be performed is $\frac{m_i}{2}$, whereas in constant-time implementations the number of isogenies of degree $\ell_i$ is the fixed bound $m_i$.

A straightforward constant-time implementation of CSURF and radical isogenies would replace all non-constant-time techniques with constant-time techniques. This would, however, drastically reduce the performance of CSURF and radical isogenies, as inversions become costly and we need to perform more (dummy) isogenies per degree. Furthermore, radical isogenies currently do not use relatively 'cheap' Vélu isogenies of low degree because they are replaced by relatively expensive radical isogenies. Such an implementation would be outperformed by any state-of-the-art CSIDH implementation in constant-time.

*Contributions.* In this paper, we are interested in constant-time implementations of CSURF and radical isogenies. We present two improvements to radical isogenies which reduce their algorithmic cost. Then, we analyze the cost and efficiency of constant-time CSURF and radical isogenies, and benchmark their performance when implemented in CSIDH in number of finite field multiplications. More concretely, our contributions are

1. *fully projective radical isogenies*, a non-trivial reformulation of radical isogenies in projective coordinates, and of the required isomorphisms between curve models. This allows us to perform radical isogenies without leaving the projective coordinates used in CSIDH. This saves an inversion per isogeny and additional inversions in the isomorphisms between curve models, which in total reduces the cost of radical isogenies in constant-time by almost 50%.
2. a *hybrid strategy* to integrate radical isogenies in CSIDH, which allows us to 're-use' torsion points that are used in the CSIDH group action evaluation to initiate a 'chain' of radical isogenies, and to keep cheap low degree Vélu isogenies. This generalizes the 'traditional' CSIDH evaluation, optimizes the evaluation of radical isogenies, and does not require sampling an extra torsion point to initiate a 'chain' of radical isogenies.
3. a *cost analysis* of the efficiency of radical isogenies in constant-time, which describes the overall algorithmic cost to perform radical isogenies, assuming the aforementioned improvements. We show that, although these improvements greatly reduce the total cost in terms of finite field operations, radical

isogenies of degree 5, 7, 11 and 13 are too costly in comparison to Vélu and $\sqrt{}$élu isogenies. We conclude that only radical isogenies of degree 4 and 9 are an improvement to 'traditional' CSIDH. Furthermore, we show that radical isogenies scale worse than Vélu isogenies with regards to the size of the base field, which reduces the effectiveness of CSURF and radical isogenies in comparison to CSIDH for large primes.

4. the *first constant-time implementation* of CSURF and radical isogenies, optimized with concern to the exponentiations used in radical isogenies, and optimal bounds and approximately optimal strategies as in [12,14,15], which allow for a more precise comparison in performance between CSIDH, CSURF and implementations using radical isogenies (CRADS) than [7,8]. Our Python-code implementation allows isogeny evaluation strategies using both traditional Vélu and $\sqrt{}$élu formulas, as well as radical isogenies and 2-isogenies (on the surface), and can thus be used to compare CSURF and CRADS against a state-of-the-art constant-time implementation of CSIDH.

5. a *performance benchmark* of CSURF and CRADS in comparison to 'traditional' CSIDH, in total finite field operations. Our benchmark is more accurate than the original benchmarks from [7,8] and shows that the 5% and 19% speed-up (respectively) diminishes to roughly 3% in a precise constant-time comparison. These results gives a detailed view of the performance of radical isogenies in terms of finite field operations, and their performance when increasing the size of the base field. We show that in low parameter sets, with the additional cost of moving to constant-time, CSURF-512 and CRADS-512 perform a bit better than CSIDH-512 implementations, with a 2.53% and 2.15% speed-up respectively. Additionally, with the hybrid strategy this speed-up is slightly larger for small primes. However, we get no speed-up for larger base fields: For primes of 1792 bits and larger, CSIDH outperforms both CSURF and CRADS due to the better scaling of Vélu isogenies in comparison to radical isogenies.

The (Python) implementation used in this paper is freely available at

https://github.com/Krijn-math/Constant-time-CSURF-CRADS.

The results from the benchmark answer the open question from Castryck, Decru, and Vercauteren in [8]: in constant-time, the CSIDH protocol gains only a small speed-up by using CSURF or radical isogenies, and only for small primes. Even stronger, our hybrid strategy shows that radical isogenies require significant improvements to make them cost effective, as computing a radical per isogeny is in most cases too expensive. Our results illustrate that constant-time CSURF and radical isogenies perform worse than large CSIDH instantiations (i.e. $\log(p) \geq 1792$), at least at the level of finite field operations.[2]

---

[2] We explicitly do not focus on performance in clock cycles; a measurement in clock cycles (on our python-code implementation) could give the impression that the underlying field arithmetic is optimized, instead of the algorithmic performance.

*Outline.* In Sect. 2, we recap the theoretical preliminaries on isogenies, CSIDH, CSURF, and radical isogenies. In Sect. 3, we introduce our first improvement: fully projective radical isogenies. This allows us to analyse the effectiveness and cost of constant-time radical isogenies in Sect. 4. Then we improve the integration of radical isogenies in CSIDH in Sect. 5, using a hybrid strategy. In Sect. 6, we benchmark constant-time CSIDH, CSURF and CRADS in terms of finite field operations, using state-of-the-art techniques for all three. Finally, in Sect. 7 we present our conclusions concerning the efficiency of radical isogenies in comparison to CSIDH in constant-time.

## 2 Preliminaries

In this section we describe the basics of isogenies, CSIDH, CSURF and radical isogenies.

Given two elliptic curves $E$ and $E'$ over a prime field $\mathbb{F}_p$, an isogeny is a morphism $\varphi : E \to E'$ such that $\mathcal{O}_E \mapsto \mathcal{O}_{E'}$. A separable isogeny $\varphi$ has a degree $\deg(\varphi)$ equal to the size of its kernel, and for any isogeny $\varphi : E \to E'$ there is a unique isogeny $\hat{\varphi} : E' \to E$ called the *dual isogeny*, with the property that $\hat{\varphi} \circ \varphi = [\deg(\varphi)]$ is the scalar point multiplication on $E$. A separable isogeny is uniquely defined by its kernel and *vice versa*; a finite subgroup $G \subset E(\overline{\mathbb{F}_p})$ defines a unique separable isogeny $\varphi_G : E \to E/G$ (up to isomorphism).

Vélu's formulas [25] provide the construction and evaluation of separable isogenies with cyclic kernel $G = \langle P \rangle$ for some $P \in E(\overline{\mathbb{F}_p})$. Both the isogeny construction of $\varphi_G$ and the evaluation of $\varphi_G(R)$ for a point $R \in E(\overline{\mathbb{F}_p})$ have a running time of $O(\#G)$, which becomes infeasible for large subgroups $G$. A new procedure presented by Bernstein, De Feo, Leroux, and Smith in ANTS-2020 based on the baby-step giant-step algorithm decreases this cost to $\tilde{O}(\sqrt{\#G})$ finite field operations [4]. We write this procedure as $\sqrt{\text{élu}}$. This new approach is based on multi-evaluations of a given polynomial, although at its core it is based on traditional Vélu's formulas.

Isogenies from $E$ to itself are *endomorphisms*, and the set of all endomorphisms of $E$ forms a ring, which is usually denoted as $\text{End}(E)$. The scalar point multiplication map $(x, y) \mapsto [N](x, y)$ and the Frobenius map $\pi : (x, y) \mapsto (x^p, x^p)$ are examples of such endomorphisms over the finite field of characteristic $p$. In particular, the order $\mathcal{O} \cong \mathbb{Z}[\pi]$ is a subring of $\text{End}(E)$. An elliptic curve $E$ is *ordinary* if it has a (commutative) endomorphism ring isomorphic to a suborder $\mathcal{O}$ of the ring of integers $\mathcal{O}_K$ for some quadratic number field $K$. A *supersingular* elliptic curve has a larger endomorphism ring: $\text{End}(E)$ is isomorphic to an order $\mathcal{O}$ in a quaternion algebra, and thus non-commutative.

### 2.1 CSIDH and Its Surface

CSIDH works with the smaller (commutative) subring $\text{End}_p(E)$ of $\text{End}(E)$, which are rational endomorphisms of a supersingular elliptic curve $E$. This subring $\text{End}_p(E)$ is isomorphic to an order $\mathcal{O} \subset \mathcal{O}_K$. As both $[N]$ and $\pi$ are defined

over $\mathbb{F}_p$, we get $\mathbb{Z}[\pi] \subset \mathrm{End}_p(E)$. To be more precise, the CSIDH protocol is based on the commutative action of the class group $\mathcal{Cl}(\mathcal{O})$ on the set $\mathcal{Ell}_p(\mathcal{O})$ of supersingular elliptic curves $E$ such that $\mathrm{End}_p(E)$ is isomorphic to the specific order $\mathcal{O} \subset \mathcal{O}_K$. The group action for an ideal class $[\mathfrak{a}] \in \mathcal{Cl}(\mathcal{O})$ maps a curve $E \in \mathcal{Ell}_p(\mathcal{O})$ to another curve $[\mathfrak{a}] \star E \in \mathcal{Ell}_p(\mathcal{O})$ (see Sect. 2.2). Furthermore, the CSIDH group action is believed to be a *hard homogeneous space* [13] that allows a Merkle-Diffie-Hellman-like key agreement protocol with commutative diagram.

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

The original CSIDH protocol uses the set $\mathcal{Ell}_p(\mathcal{O})$ with $\mathcal{O} \cong \mathbb{Z}[\pi]$ and $p = 3$ mod 4 (named the *floor*). To also benefit from 2-isogenies, the CSURF protocol switches to elliptic curves on the *surface* of the isogeny graph, that is, $\mathcal{Ell}_p(\mathcal{O})$ with $\mathcal{O} \cong \mathbb{Z}[\frac{1+\pi}{2}]$. Making 2-isogenies useful requires $p = 7 \mod 8$.

## 2.2     The Group Action of CSIDH and CSURF

The traditional way of evaluating the group action of an element $[\mathfrak{a}] \in \mathcal{Cl}(\mathcal{O})$ is by using 'traditional' Vélu's [25] or $\sqrt{}$élu [4] formulas. The group action maps $E \to [\mathfrak{a}] \star E$ and can be described by the kernel $E[\mathfrak{a}]$ of an isogeny $\varphi_{\mathfrak{a}}$ of finite degree. Specifically, $[\mathfrak{a}] \star E = E/E[\mathfrak{a}]$ where

$$
E[\mathfrak{a}] = \bigcap_{\varphi \in \mathfrak{a}} \mathrm{Ker}(\varphi).
$$

In both CSIDH and CSURF, we apply specific elements $[\mathfrak{l}_i] \in \mathcal{Cl}(\mathcal{O})$ such that $\mathfrak{l}_i^{\pm 1} = (\ell_i, \pi \mp 1)$ and $\ell_i$ is the $i$-th odd prime dividing $(p + 1)$. For $\mathfrak{l}_i$, we have

$$
E[\mathfrak{l}_i^{\pm 1}] = E[\ell_i] \cap E[\pi \mp 1],
$$

where $P \in E[\ell_i]$ means $P$ is a point of order $\ell_i$ and $P \in E[\pi \mp 1]$ implies $\pi(P) = \pm P$, so $P$ is either an $\mathbb{F}_p$-rational point or a zero-trace point over $\mathbb{F}_{p^2}$. Thus, the group action $E \to [\mathfrak{l}_i^{\pm 1}] \star E$ is usually calculated by sampling a point $P \in E[\mathfrak{l}_i^{\pm 1}]$ and applying Vélu's formulas with input point $P$. A secret key for CSIDH is then a vector $(e_i)$, which is evaluated as $E \to \prod_i [\mathfrak{l}_i]^{e_i} \star E$. CSURF changes the order $\mathcal{O}$ used to $\mathbb{Z}[\frac{1+\pi}{2}]$ to also perform 2-isogenies on the surface of the isogeny graph; these 2-isogenies do not require the sampling of a 2-order point but can instead be calculated by a specific formula based on radical computations.

*Key space.* Originally, the secret key $e = (e_i)$ was sampled from $[\![-m \mathrel{..} m]\!]^n$ for some bound $m \in \mathbb{N}$. This was improved in [12,15,18] by varying the bound $m$ per degree $\ell_i$ (a weighted $L_\infty$-norm ball). Further developments with regards to improving the key space are presented in [19], using an $(L_1 + L_\infty)$-norm ball, and in CTIDH ([3]). These methods can give significant speed-ups. In their cores, they rely on (variations of) Vélu isogenies to evaluate the group action. In [7,8], the authors compare the performance of radical isogenies to CSIDH by using an unweighted $L_\infty$-norm ball for CSIDH-512 versus a weighted $L_\infty$-norm ball for the implementation using radical isogenies. This gives a skewed benchmark, which favors the performance of CSURF and CRADS. In this paper, to make a fair comparison to the previous work, we continue in the line of [12,15,18] by using *weighted* $L_\infty$-norm balls for the implementations of CSIDH, CSURF and CRADS. It remains interesting to analyse the impact of radical isogenies in key spaces that are not based on weighted $L_\infty$-norm balls. As radical isogenies can easily be made to have exactly the same cost per degree (with only slightly extra cost), they are interesting to analyse with respect to CTIDH.

### 2.3   The Tate Normal Form

CSURF introduced the idea to evaluate a 2-isogeny by radical computations. [8] extends this idea to higher degree isogenies, using a different curve model than the Montgomery curve. To get to that curve model, fix an $N$-order point $P$ on $E$ with $N \geq 4$. Then, there is a unique isomorphic curve $E(b, c)$ over $\mathbb{F}_p$ such that $P$ is mapped to $(0,0)$ on $E(b, c)$. The curve $E(b, c)$ is given by Eq. (1), and is called the Tate normal form of $(E, P)$:

$$E(b,c)/\mathbb{F}_p \colon y^2 + (1 - c)x - by = x^3 - bx^2, \quad b, c \in \mathbb{F}_p. \tag{1}$$
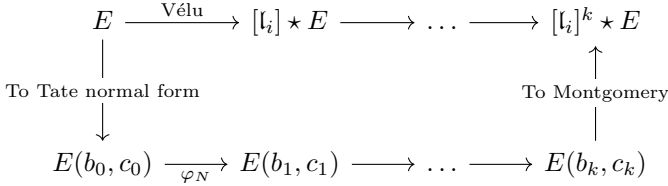
The curve $E(b, c)$ has a non-zero discriminant $\Delta(b, c)$ and in fact, it can be shown that the reverse is also true: for $b, c \in \mathbb{F}_p$ such that $\Delta(b, c) \neq 0$, the curve $E(b, c)$ is an elliptic curve over $\mathbb{F}_p$ with $(0,0)$ of order $N \geq 4$. Thus the pair $(b, c)$ uniquely determines a pair $(E, P)$ with $P$ having order $N \geq 4$ on some isomorphic curve $E$ over $\mathbb{F}_p$. In short, there is a bijection between the set of isomorphism classes of pairs $(E, P)$ and the set of $\mathbb{F}_p$-points of $\mathbb{A}^2 - \{\Delta = 0\}$. The connection with *modular curves* is explored in more detail in [21].

### 2.4   Radical Isogenies

Let $E_0$ be a supersingular Montgomery curve over $\mathbb{F}_p$ and $P_0$ a point of order $N$ with $N \geq 4$. Additionally, let $E_1 = E_0 / \langle P_0 \rangle$, and $P_1$ a point of order $N$ on $E_1$ such that $\hat{\varphi}(P_1) = P_0$ where $\hat{\varphi}$ is the dual of the $N$-isogeny $\varphi \colon E_0 \to E_1$. The pairs $(E_0, P_0)$ and $(E_1, P_1)$ uniquely determine Tate normal parameters $(b_0, c_0)$ and $(b_1, c_1)$ with $b_i, c_i \in \mathbb{F}_p$.

Castryck, Decru, and Vercauteren proved the existence of a function $\varphi_N$ that maps $(b_0, c_0)$ to $(b_1, c_1)$ in such a way that it can be applied iteratively. This computes a chain of $N$-isogenies without the need to sample points of order $N$

per iteration. As a consequence, by mapping a given supersingular Montgomery curve $E/\mathbb{F}_p$ and some point $P$ of order $N$ to its Tate normal form, we can evaluate $E \to [\mathfrak{l}_i] \star E$ without any points (except for sampling $P$). Thus, it allows us to compute $E \to [\mathfrak{l}_i]^k \star E$ without having to sample $k$ points of order $N$.

$$E \xrightarrow{\text{Vélu}} [\mathfrak{l}_i] \star E \longrightarrow \ldots \longrightarrow [\mathfrak{l}_i]^k \star E$$

To Tate normal form

To Montgomery

$$E(b_0, c_0) \xrightarrow{\varphi_N} E(b_1, c_1) \longrightarrow \ldots \longrightarrow E(b_k, c_k)$$

Notice that the top row and the bottom row of the diagram are isomorphic. The map $\varphi_N$ is an elementary function in terms of $b$, $c$ and $\alpha = \sqrt[N]{\rho}$ for a specific element $\rho \in \mathbb{F}_p(b, c)$: hence the name 'radical' isogeny. Over $\mathbb{F}_p$, an $N$-th root is unique whenever $N$ and $p-1$ are co-prime (as the map $x \mapsto x^N$ is then a bijection). Notice that this in particular holds for all odd primes $\ell_i$ of a CSIDH prime $p = h \cdot \prod \ell_i - 1$ for a suitable cofactor $h$. Castryck, Decru, and Vercauteren provided the explicit formulas of $\varphi_N$ for $N \in \{2, 3, 4, 5, 7, 9, 11, 13\}$. For larger degrees the formulas could not be derived yet. They also suggest the use of radical isogenies of degree 4 and 9 instead of 2 and 3, respectively.

Later work by Onuki and Moriya [21] provides similar radical isogenies on Montgomery curves instead of Tate normal curves. Although their results are of theoretical interest, they only provide such radical isogenies for degree 3 and 4. For degree 3, the use of degree 9 radical isogenies on Tate normal curves is more efficient, while for degree 4 the difference between their formulas and those presented in [8] are negligible. We, therefore, focus only on radical isogenies on Tate normal curves for this work.

## 3   Fully Projective Radical Isogenies

In this section we introduce our first improvement to radical isogenies: *fully projective* radical isogenies. These allow to us bypass all inversions required for radical isogenies. We perform (a) the radical isogenies on Tate normal curves in projective coordinates, and (b) the switch between the Montgomery curve and the Tate normal curve, and back, in projective coordinates. (a) requires non-trivial work which we explain in Sect. 3.1, whereas (b) is only tediously working out the correct formulas. The savings are worth it: (a) saves an inversion per radical isogeny and (b) saves numerous inversions in overhead costs. All in all, it is possible to remain in projective coordinates throughout the whole implementation, which saves about 50% in terms of finite field operations in comparison to affine radical isogenies in constant time.

### 3.1   Efficient Radicals for Projective Coordinates

The cost of an original (affine) radical isogenies of degree $N$ in constant-time is dominated by the cost of the $N$-th root and one inversion per iteration. We

introduce projective radical isogenies so that we do not require this inversion. In a constant-time implementation, projective radical isogenies save approximately 50% of finite field operations in comparison to affine radical isogenies. A straightforward translation to projective coordinates for radical isogenies would save an inversion by writing the Tate normal parameter $b$ (when necessary $c$) as $(X : Z)$. However, this comes at the cost of having to calculate both $\sqrt[N]{X}$ and $\sqrt[N]{Z}$ in the next iteration. Using the following lemma, we save one of these exponentiations.

**Lemma 1.** *Let $N \in \mathbb{N}$ such that $\gcd(N, p - 1) = 1$. Write $\alpha \in \mathbb{F}_p$ as $(X : Z)$ in projective coordinates with $X, Z \in \mathbb{F}_p$. Then $\sqrt[N]{\alpha} = (\sqrt[N]{XZ^{N-1}} : Z)$.*

*Proof.* As $\alpha = (X : Z) = (XZ^{N-1} : Z^N)$, we only have to show that the $N$-th root is unique. But $N$ is co-prime with $p - 1$, so the map $x \mapsto x^N$ is a bijection. Therefore, the $N$-th root $\sqrt[N]{\rho}$ is unique for $\rho \in \mathbb{F}_p$, so $\sqrt[N]{Z^N} = Z$. 

Crucially for radical isogenies, we want to compute $N$-th roots where $N = \ell_i$ for some $i$, working over the base field $\mathbb{F}_p$ with $p = h \cdot \prod_i \ell_i - 1$, and so for such an $N$ we get $\gcd(N, p - 1) = 1$. This leads to the following corollary.

**Corollary 1.** *The representation $(XZ^{N-1} : Z^N)$ saves an exponentiation in the calculation of a radical isogeny of degree $N = \ell_i$ in projective coordinates.*

This brings the cost of a projective radical isogeny of small degree $\ell_i$ down to below $1.25 \log(p)$. Compared with affine radical isogeny formulas in constant-time, which cost roughly two exponentiations, such projective formulas cost approximately half of the affine ones in terms of finite field operations. The effect this has for degrees 2, 3, 4, 5, 7 and 9 can be seen in Table 1. A similar approach as Lemma 1 works for radical isogenies of degree $N = 4$.

### 3.2  Explicit Projective Formulas for Low Degrees

We give the projective radical isogeny formulas for three cases: degree 4, 5 and 7. For larger degrees, it becomes increasingly more tedious to work out the projective isogeny maps. In the repository, we provide formulas for $N \in \{2, 3, 4, 5, 7, 9\}$.

**Projective isogeny of degree 4.** The Tate normal form for degree 4 is $E : y^2 + xy - by = x^3 - bx^2$ for some $b \in \mathbb{F}_p$. From [8], we get $\rho = -b$ and $\alpha = \sqrt[4]{\rho}$, and the affine radical isogeny formula is

$$\alpha \mapsto b' = -\frac{\alpha(4\alpha^2 + 1)}{(2\alpha + 1)^4}.$$

Projectively, write $\alpha$ as $(X : Z)$ with $X, Z \in \mathbb{F}_p$. Then the projective formula is

$$(X : Z) \mapsto (X'Z'^4 : Z') \quad \text{with} \tag{2}$$
$$X' = (4X^2 + Z^2)XZ, \quad \text{and} \quad Z' = 2X + Z.$$

This isogeny is a bit more complex than it seems. First, notice that the denominator of the affine map is a fourth power. One would assume that it is therefore enough to map to $(X' : Z')$ and continue by taking only the fourth

root of $X'$ and re-use $Z' = \sqrt[4]{Z'^4}$. However, as $\gcd(4, p - 1) = 2$, the root $\delta = \sqrt[4]{Z'}$ is not unique. Following [8] we need to find the root $\delta$ that is a quadratic residue in $\mathbb{F}_p$. We can force $\delta$ to be a quadratic residue: notice that $(X' : Z'^4)$ is equivalent to $(X'Z'^4 : Z'^8)$, so that taking fourth roots gives $(\sqrt[4]{X'Z'^4} : \sqrt[4]{Z'^8}) = (\sqrt[4]{X'Z'^4} : Z'^2)$, where we have forced the second argument to be a square, and so we get the correct fourth root.

Therefore, by mapping to $(X'Z'^4 : Z')$ we compute $\sqrt[4]{-b'}$ as $(\sqrt[4]{X'Z'^4} : Z'^2)$ using only one 4-th root. This allows us to repeat Eq. (2) using only one exponentiation, without the cost of the inversion required in the affine version.

**Projective isogeny of degree 5.** The Tate normal form for degree 5 is $E : y^2 + (1 - b)xy - by = x^3 - bx^2$ for some $b \in \mathbb{F}_p$. From [8] we get $\rho = b$ and $\alpha = \sqrt[5]{\rho}$, and the affine radical isogeny formula is

$$\alpha \mapsto b' = \alpha \cdot \frac{\alpha^4 + 3\alpha^3 + 4\alpha^2 + 2\alpha + 1}{\alpha^4 - 2\alpha^3 + 4\alpha^2 - 3\alpha + 1}.$$

Projectively, write $\alpha$ as $(X : Z)$ with $X, Z \in \mathbb{F}_p$. Then the projective formula is

$$(X : Z) \mapsto (X'Z'^4 : Z') \quad \text{with} \tag{3}$$
$$X' = X(X^4 + 3X^3Z + 4X^2Z^2 + 2XZ^3 + Z^4), \quad \text{and}$$
$$Z' = Z(X^4 - 2X^3Z + 4X^2Z^2 - 3XZ^3 + Z^4).$$

Notice that the image is $(X'Z'^4 : Z')$ instead of $(X' : Z') = (X'Z'^4 : Z'^5)$, following Lemma 1. This allows us in the next iteration to compute $\sqrt[5]{b} = (\sqrt[5]{X} : \sqrt[5]{Z}) = (\sqrt[5]{X'Z'^4} : Z')$ using only one 5-th root. This allows us to repeat Eq. (3) using only one exponentiation, without the cost of the inversion required in the affine version.

**Projective isogeny of degree 7.** The Tate normal form for degree 7 is $E : y^2 + (-b^2 + b + 1)xy + (-b^3 + b^2)y = x^3 + (-b^3 + b^2)x^2$ for some $b \in \mathbb{F}_p$, with $\rho = b^5 - b^4$ and $\alpha = \sqrt[7]{\rho}$. However, the affine radical isogeny is already too large to display here, and the projective isogeny is even worse. However, we can still apply Lemma 1. The projective isogeny maps to $(X'Z'^6 : Z')$ and in a next iteration we can compute $\alpha = \sqrt[7]{\rho} = \sqrt[7]{b^5 - b^4}$ as $(\sqrt[7]{X^4Z^2(X - Z)} : Z)$.

### 3.3   Cost of Projective Radical Isogenies per Degree

In Table 1, we compare the cost of affine radical isogenies to projective radical isogenies. In Table 2, we compare the cost in switching between the different curve models for affine and projective coordinates.

In summary, fully projective radical isogenies are almost twice as fast as the original affine radical isogenies for constant-time implementations. Nevertheless, as we will see in the analysis of Sect. 4.1, the radical isogenies of degree 5, 7, 11 and 13 still perform worse than 'traditional' Vélu isogenies in realistic scenarios.

**Table 1.** Comparison between affine radical isogenies from [8] and the projective radical isogenies in this work. The letters **E**, **M**, **S**, **A** and **I** denote exponentiation, multiplication, squaring, addition and inversion respectively. The last column expresses the ratio projective/affine in terms of finite field multiplications over $\mathbb{F}_p$ for a prime of 512 bits, using close-to-optimal addition chains for exponentiation and inversion, assuming **S** = **M** and ignoring **A**.

| | Affine | Projective | Ratio |
|---|---|---|---|
| **Degree** | ([8]) | (This work.) | projective/affine |
| 2-isogeny | **E** + 4**M** + 6**A** + **I** | **E** + 3**M** + 5**S** + 10**A** | 50.4% |
| 3-isogeny | **E** + 6**M** + 3**A** | **E** + 2**M** + 10**A** | 99.3% |
| 4-isogeny | **E** + 4**M** + 3**A** + **I** | **E** + 6**M** + 4**S** + 3**A** | 50.5% |
| 5-isogeny | **E** + 7**M** + 6**A** + **I** | **E** + 8**M** + 6**S** + 18**A** | 50.7% |
| 7-isogeny | **E** + 24**M** + 20**A** + **I** | **E** + 14**M** + 4**S** + 64**A** | 50.5% |
| 9-isogeny | **E** + 69**M** + 58**A** + **I** | **E** + 61**M** + 10**S** + 202**A** | 52.1% |

**Table 2.** Comparison between the cost of different functions to switch curve models, necessary to perform radical isogenies. Affine results from [8] and projective results from this work.

| | Affine | Projective | Ratio |
|---|---|---|---|
| **Function** | ([8]) | (This work.) | projective/affine |
| Mont+ to Mont- | **E** + **M** + **S** + 2**A** + **I** | **E** + 2**M** + 2**S** + 4**A** | 50.1% |
| Mont- to Mont+ | **E** + **M** + **S** + 2**A** + **I** | **E** + 2**S** + 4**A** | 50.0% |
| Mont- to Tate4 | 7**M** + **S** + **A** + **I** | 5**M** + 8**S** + 7**A** | 2.1% |
| Tate4 to Mont- | 2**E** + 3**M** + **S** + 7**A** + 2**I** | 2**E** + 6**M** + **S** + 11**A** | 50.1% |
| Full overhead CSURF | 7**E** + 17**M** + 6**S** + 19**A** + 5**I** | 7**E** + 18**M** + 16**S** + 35**A** | 58.5% |
| Mont+ to TateN | **E** + 9**M** + **S** + 11**A** + **I** | **E** + 13**M** + 7**S** + 13**A** | 51.1% |
| TateN to Mont+ | 3**E** + 20**M** + 7**S** + 34**A** + **I** | 3**E** + 33**M** + 11**S** + 65**A** | 75.9% |
| Full overhead CRADS | 4**E** + 34**M** + 14**S** + 54**A** + 4**I** | 4**E** + 54**M** + 22**S** + 83**A** | 50.9% |

## 4   Cost Analysis of Constant-Time Radical Isogenies

In this section, we analyze the cost and effectiveness of radical isogenies on Tate normal curves in constant-time. In a simplified model, the cost of performing $n$ radical $\ell$-isogenies can be divided into 4 steps.

1. Sample a point $P$ on $E_A$ of order $\ell$;
2. Map $(E_A, P)$ to the (isomorphic) Tate normal curve $E_0$ with $P \mapsto (0, 0)$;
3. Perform the radical isogeny formula $n$ times: $E_0 \to E_1 \to \ldots \to E_n$;
4. Map $E_n$ back to the correct Montgomery curve $E_{A'} = [\mathfrak{l}]^n \star E_A$.

In each of these steps, the cost is dominated by the number of exponentiations (**E**) and inversions (**I**). Using Tables 1 and 2, in an affine constant-time implementation, moving to the Tate normal curve (step 2) will cost close to 1 **E** + 1 **I**, an affine radical isogeny (step 3) costs approximately 1 **E** + 1 **I** per isogeny, and moving back to the Montgomery curve (step 4) will cost about 3 **E** + 1 **I**.

*Inversions.* In contrast to ordinary CSIDH, radical isogenies would require these inversions to be constant-time, as the value that is inverted can reveal valuable information about the isogeny walk related to the secret key. Two methods to compute the inverse of an element $\alpha \in \mathbb{F}_p$ in constant-time are 1) by Fermat's little theorem[3]: $\alpha^{-1} = \alpha^{p-2}$, or 2) by masking the value that we want to invert with a random value $r \in \mathbb{F}_p$, computing $(r\alpha)^{-1}$ and multiplying by $r$ again. Method 1 makes inversion as costly as exponentiation, while method 2 requires a source of randomness, which is an impediment from a crypto-engineering point of view. Using Fermat's little theorem almost doubles the cost of CSURF and of a radical isogeny in low degrees $(2, 3, 4, 5, 7)$ and significantly increases the cost of a radical isogeny of degree 9, 11 or 13. Furthermore, such constant-time inversions increase the overhead of switching to Tate normal form and back to Montgomery form, which in total makes performing $n$ radical isogenies less effective. Both methods of inversion are unfavorable from a crypto-engineering view, and thus we implement the fully projective radical isogenies from Sect. 3 to by-pass all inversions completely for radical isogenies.

*Approximate cost of radical isogenies.* We can now approximate the cost of evaluating fully projective radical isogenies in constant-time. We can avoid (most of) the cost of step 1 with a hybrid strategy (see Sect. 5). Projective coordinates avoid the inversion required in step 2 to move from the Montgomery curve to the correct Tate normal curve, and the inversion required in step 4 to move from the Tate normal curve back to the Montgomery curve. In step 3, projective radical isogenies save an inversion per isogeny, and so step 3 costs approximately $n \mathbf{E}$. In total, performing $n$ radical $\ell$-isogenies therefore costs approximately $(n+4)\mathbf{E}$.

At first sight, this approximated cost does not seem to depend on $\ell$. However, there is some additional cost besides the exponentiation per isogeny in step 3, and this additional cost grows with $\ell$. But, the cost of an exponentiation is larger than $\log_2(p) \mathbf{M}$ and so overshadows the additional cost. For more details, see Table 1. For this analysis, the approximated cost fits for small degrees.

The cost of exponentiation is upperbounded by $1.5 \log(p)$ by the (suboptimal) square-and-multiply method, assuming squaring ($\mathbf{S}$) costs as much as multiplication ($\mathbf{M}$). In total, we get the following approximate cost:

**Lemma 2.** *The cost to perform $n$ radical isogenies (using Tate normal curves) of degree $\ell \in \{5, 7, 9, 11, 13\}$ is at least*

$$(n + 4) \cdot \alpha \cdot \log_2(p),$$

*finite field multiplications (**M**) where $\alpha \in [1, 1.5]$ depends on the method to perform exponentiation (assuming $\mathbf{S} = \mathbf{M}$).*

### 4.1   Analysis of Effectiveness of Radical Isogenies

In this subsection, we analyze the efficiency of radical isogenies in comparison to Vélu isogenies, assuming the results from the previous sections. We argue that

---

[3] Bernstein and Yang [5] give a constant-time inversion based on gcd-computations. We have not implemented this, as avoiding inversions completely is cheaper.

the cost of $(n + 4) \cdot \alpha \cdot \log_2(p)$ from Lemma 2 for radical isogenies is too high and it is therefore not worthwhile to perform radical isogenies for degrees $5, 7, 11$ and $13$. Degrees 2 and 3, however, benefit from the existence of radical isogenies of degree 4 and 9. Degree 4 and 9 isogenies cost only one exponentiation, but evaluate as two. This implies that performing radical isogenies is most worthwhile in degrees 2 and 3. We write 2/4 and 3/9 as shorthand for the combinations of degree 2 and 4, resp. degree 3 and 9 isogenies.

The three crucial observations in our analysis are

1. Current faster $\sqrt{}$élu isogeny formulas require $\mathcal{O}(\sqrt{\ell^{\log_2 3}})$ field multiplications, whereas the cost of a radical isogeny scales as a factor of $\log_2(p)$ (for more details, see [1,4]);
2. The group action evaluation first performs one block using $\sqrt{}$élu isogeny formulas, and then isolates the radical isogeny computations. What is particularly important among these $\sqrt{}$élu isogeny computations, is that removing one specific $\ell'$-isogeny does not directly decrease the number of points that need to be sampled. Internally, the group action looks for a random point $R$ and performs all the possible $\ell_i$-isogenies such that $\left[\frac{p+1}{\ell_i}\right] R \neq \mathcal{O}$.
3. Replacing the smallest Vélu $\ell$-isogeny with a radical isogeny could reduce the sampling of points in that specific Vélu isogeny block. This is because the probability of reaching a random point $R$ of order $\ell$ is $\frac{\ell-1}{\ell}$, which is small for small $\ell$. Additionally, the cost of verifying $\left[\frac{p+1}{\ell}\right] R \neq \mathcal{O}$ is about $1.5 \log_2 \left(\frac{p+1}{\ell}\right)$ point additions $\approx 9 \log_2 \left(\frac{p+1}{\ell}\right)$ field multiplications (for more details see [10]). In total, sampling $n$ points of order $\ell$ costs

$$\texttt{sampling}(n, p, \ell) = 9 \left\lfloor \frac{n\ell}{\ell - 1} \right\rfloor \log_2 \left(\frac{p+1}{\ell}\right) \mathbf{M}$$

$$\approx 9 \left\lfloor \frac{n\ell}{\ell - 1} \right\rfloor (\log_2(p) - \log_2(\ell))\mathbf{M}.$$

Nevertheless, using radical isogenies for these degrees does not save the sampling of $n$ points, just a fraction of them. To be more precise, let $\ell' > \ell$ be the next smallest prime such that the group action requires $n'$ $\ell'$-isogenies and $\left\lfloor \frac{n\ell}{\ell-1} \right\rfloor \geq \left\lfloor \frac{n'\ell'}{\ell'-1} \right\rfloor$. Then the savings are given by their difference with respect to the cost sampling such torsion-points (see Eq. (4)).

$$9 \left( \left\lfloor \frac{n\ell}{\ell - 1} \right\rfloor - \left\lfloor \frac{n'\ell'}{\ell' - 1} \right\rfloor \right) (\log_2(p) - \log_2(\ell))\mathbf{M}. \tag{4}$$

Whenever $\left\lfloor \frac{n\ell}{\ell-1} \right\rfloor < \left\lfloor \frac{n'\ell'}{\ell'-1} \right\rfloor$, using radical isogenies does not reduce the number of points that need to be sampled.

As an example for the cost in a realistic situation, we take the approximately optimal bounds analyzed in [12,20]. In both works, $\log_2(p) \approx 512$ and the first five smallest primes $\ell_i$'s in $\{3, 5, 7, 11, 13\}$ have bounds $m_i$ that satisfy

$$\left\lfloor \frac{m_0\ell_0}{\ell_0 - 1} \right\rfloor = \left\lfloor \frac{m_1\ell_1}{\ell_1 - 1} \right\rfloor = \left\lfloor \frac{m_2\ell_2}{\ell_2 - 1} \right\rfloor = \left\lfloor \frac{m_3\ell_3}{\ell_3 - 1} \right\rfloor = \left\lfloor \frac{m_4\ell_4}{\ell_4 - 1} \right\rfloor .$$

Thus, there are no savings concerning sampling of points when including small degree radical isogenies. Clearly, performing $n$ radical $\ell$-isogenies becomes costlier than using $\sqrt{\text{é}}$lu isogenies, and thus the above analysis suggests radical isogenies need their own optimal bounds to be competitive. The analysis is different for degree 2 and 3, where we can perform 4-and 9-isogenies in $\left\lceil \frac{n}{2} \right\rceil$ radical computations instead of $n$ computations. In fact, 4-isogenies directly reduces the sampling of points by decreasing the bounds of the other primes $\ell_i$'s. Nevertheless, performing $n$ radical isogenies takes at least $(n + 4)\log_2(p)$ field multiplications (Lemma 2), which implies higher costs (and then lower savings) for large prime instantiations. For example, a single radical isogeny in a 1024-bit field costs twice as much as a single radical isogeny in a 512-bit field, and in a 2048-bit field this becomes four times as much. These expected savings omit the cost of sampling an initial point of order $\ell_i$, as we show in Sect. 5 how we can find such points with little extra cost with high probability.

### 4.2 Further Discussion

In this subsection, we describe the two further impacts on performance in constant-time and higher parameter sets in more detail: Radical isogenies scale badly to larger primes, as their cost scales with $\log(p)$, and dummy-free isogenies are more expensive, as we need to switch direction often for a dummy-free evaluation.

*Radical isogenies do not scale well.* Using the results in Tables 1 and 2, the cost of a single radical isogeny is approximately 600 finite field operations, with an overhead of about 2500 finite field operations for a prime of 512 bits. Thus, a CSURF-512 implementation (which uses 2/4- radical isogenies) or a CRADS-512 implementation (which uses 2/4- and 3/9- radical isogenies) could be competitive with a state-of-the-art CSIDH-512 implementation. However, implementations using radical isogenies scale worse than CSIDH implementations, due to the high cost of exponentiation in larger prime fields. For example, for a prime of 2048 bits, just the overhead of switching curve models is already over 8500 finite field operations, which is close to 1% of total cost for a 'traditional' CSIDH implementation. Therefore, CSIDH is expected to outperform radical isogenies for larger primes. In Sect. 6, we demonstrate this using a benchmark we have performed on CSIDH, CSURF, CRADS, and an implementation using the hybrid strategy we introduce in Sect. 5, for six different prime sizes, from 512 bits up to 4096 bits. These prime sizes are realistic: several analyses, such as [6,11,22], call the claimed quantum security of the originally suggested prime sizes for CSIDH (512, 1024 and 1792 bits) into question. We do not take a stance on this discussion, and therefore provide an analysis that fits both sides of the discussion.

*Dummy-free radical isogenies are costly.* Recall that radical isogenies require an initial point $P$ of order $N$ to switch to the right Tate normal form, depending on the direction of the isogeny. So, two kinds of curves in Tate normal form arise: $P$ belongs either to $E[\pi - 1]$ or to $E[\pi + 1]$. Now, a dummy-free chain of radical isogenies requires (at some steps of the group action) to switch the direction of the isogenies, and therefore to switch to a Tate normal form where $P$ belongs to either $E[\pi - 1]$ or $E[\pi + 1]$. As we switch direction $m_i - |e_i|$ times, this requires $m_i - |e_i|$ torsion points. That is, a dummy-free implementation of a chain of radical isogenies will require at least $(m_i - |e_i|)$ torsion points, which leaks information on $e_i$. We can make this procedure secure by sampling $m_i$ points every time, but this costs too much. These costs could be decreased by pushing points through radical isogenies, however, this is still not cost-effective. In any case, we will only focus on dummy-based implementations of radical isogenies.

## 5  A Hybrid Strategy for Radical Isogenies

In this section we introduce our second improvement to radical isogenies: a *hybrid strategy* for integrating radical isogenies into CSIDH. In [8], radical $\ell$-isogenies replace Vélu $\ell$-isogenies, and they are performed before the CSIDH group action, by sampling a point of order $\ell$ to initiate a 'chain' of radical $\ell$-isogenies. Such an approach replaces cheap Vélu $\ell$-isogenies with relatively expensive radical $\ell$-isogenies and requires finding a point of order $\ell$ to initiate this 'chain'. The hybrid strategy combines the 'traditional' CSIDH group action evaluation with radical isogenies in an optimal way, so that we do not sacrifice cheap Vélu isogenies of low degree and do not require another point of order $\ell$ to initiate the 'chain'. This substantially improves the efficiency of radical isogenies.

Concretely, in 'traditional' CSIDH isogeny evaluation, one pushes a torsion point $T$ through a series of $\ell$-isogenies with Vélu's formulas. This implies that at the end of the series of Vélu isogenies, such a point $T$ might still have suitable torsion to initiate a chain of radical isogenies. Re-using this point saves us having to specifically sample a torsion point to initiate radical isogenies. Furthermore, with this approach we can do *both* radical and Vélu isogenies for such $\ell$ where we have radical isogenies. We show this hybrid strategy generalizes CSIDH, CSURF and CRADS (an implementation with radical isogenies) and gives an improved approach to integrate radical isogenies on Tate normal curves in CSIDH.

In this section, $V$ refers to the set of primes for which we have Vélu isogenies (i.e. all $\ell_i$), and $R \subset V$ refers to the set of degrees for which we have radical isogenies. Currently, $R = \{3, 4, 5, 7, 9, 11, 13\}$.

### 5.1  A Hybrid Strategy for Integration of Radical Isogenies

Vélu isogenies for degree $\ell_i$ with $i \in R$ are much cheaper than the radical isogenies for those degrees, as a single radical isogeny always requires at least one exponentiation which costs $O(\log(p))$. The downside to Vélu isogenies is that they require a torsion point per isogeny. However, torsion points can be

re-used for Vélu isogenies of many degrees by pushing them through the isogeny, and so the cost of point sampling is amortized over all the degrees where Vélu isogenies are used. Thus, although for a single degree $n$ radical isogenies are much cheaper than $n$ Vélu isogenies, this does not hold when the cost of point sampling is distributed over many other degrees. The idea of our hybrid strategy is to do both types of isogeny per degree: we need to perform a certain amount of Vélu blocks in any CSIDH evaluation, so we expect a certain amount $T_i$ of points of order $\ell_i$. So, for $i \in R$, we can use $T_i - 1$ of these points to perform Vélu $\ell_i$-isogenies, and use the last one to initiate the chain of radical $\ell_i$-isogenies. Concretely, we split up the bound $m_i$ for $i \in R$ into $m_i^v$ and $m_i^r$. Here, $m_i^v$ is the number of Vélu isogenies, which require $m_i^v$ points of order $\ell_i$, and $m_i^r$ is the number of radical isogenies, which require just 1 point of order $\ell_i$.

Thus, our *hybrid strategy* allows for evaluation of CSIDH with radical isogenies, in such a way that we can the following parameters

- for $i \in R$, $m_i^r$ is the number of radical isogenies of degree $\ell_i$,
- for $i \in V$, $m_i^v$ is the number of Vélu/$\sqrt{}$élu isogenies of degree $\ell_i$.

For $i \in R$, we write $m_i = m_i^v + m_i^r$ for simplicity. What makes the difference with the previous integration of radical isogenies, is that this hybrid strategy allows $R$ and $V$ to overlap! That is, the hybrid strategy does not require you to pick between Vélu *or* radical isogenies for $\{3, 5, 7, 9, 11, 13\}$. In this way, hybrid strategies generalize both CSIDH/CSURF and CRADS:

**Lemma 3.** *The CSIDH/CSURF group action evaluation as in [7, 9], and the radical isogenies evaluation from [8] are both possible in this hybrid strategy.*

*Proof.* Take $m_i^r = 0$ for all $i \in R$ to get the CSIDH/CSURF group action evaluation, and take $m_i^v = 0$ for all $i \in R$ to get the radical isogenies evaluation.

In the rest of this section, we look at non-trivial hybrid parameters (i.e. there is some $i$ such that both $m_i^v$ and $m_i^r$ are non-zero) to improve the performance of CSIDH and CRADS. As we can predict $T_i$ (the number of points of order $\ell_i$) in a full CSIDH group action evaluation, we can choose our parameter $m_i^v$ optimally with respect to $T_i$: for $i \in R$, we take $m_i^v = T_i - 1$ and use the remaining point of order $\ell_i$ to initiate the 'chain' of radical $\ell_i$ isogenies of length $m_i^r$.

### 5.2   Choosing Parameters for Hybrid Strategy

As we explained above, the parameter $m_i^v$ is clear given $T_i$, and we are left with optimizing the value $m_i^r$. Denote the cost of the overhead to switch curve models by $c_{\mathrm{overhead}}$ and the cost of a single isogeny by $c_{\mathrm{single}}(\ell)$, then the cost of performing the $m_i^r$ radical isogenies is clearly $c = c_{\mathrm{overhead}} + m_i^r \cdot c_{\mathrm{single}}(\ell_i)$ (see also Lemma 2). Furthermore, given $m_i^v$, the increase in key space is

$$b = \log_2 \left( \frac{2 \cdot (m_i^r + m_i^v) + 1}{2 \cdot m_i^v + 1} \right).$$

So, we can minimize $c/b$ for a given $m_i^v$ (independent of $p$). This minimizes the number of field operations per bits of security. Notice that for degree 3, the use of 9-isogenies means we get a factor $\frac{1}{2}$ for the cost of single isogenies, as we only need to perform half as many.

It is possible that the 'optimal' number $m_i^r$ is higher, when $c/b$ is still lower than in CSIDH. However, we heuristically argue that such an optimum can only be slightly higher, as the increase in bits of security decreases quite rapidly.

### 5.3   Algorithm for Evaluation of Hybrid Strategy

Evaluating the hybrid strategy requires an improved evaluation algorithm, as we need to re-use the 'left-over' torsion point at the right moment of a 'traditional' CSIDH evaluation. We achieve this by first performing the CSIDH evaluation for all $i \in V$ and decreasing $m_i^v$ by 1 if a Vélu $\ell_i$-isogeny is performed. If $m_i^v$ becomes zero, we remove $i$ from $V$, so that the next point of order $\ell_i$ initiates the 'chain' of radical isogenies of length $m_i^r$. After this, we remove $i$ from $R$ too.

---

**Algorithm 1.** High-level evaluation of hybrid strategy for radical isogenies

**Input:** $A \in \mathbb{F}_p$, a key $(e_1, \ldots, e_n)$, a set $V$ (Vélu isog.) and a set $R$ (radical isog.).
**Output:** $B \in \mathbb{F}_p$ such that $\prod [\mathfrak{l}_i]^{e_i} \star E_A = E_B$
1: **while** $m_i \neq 0$ for $i \in V \cup R$ **do**
2:     Sample $x \in \mathbb{F}_p$, set $s \leftarrow 1$ if $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$, else $s \leftarrow -1$.
3:     Let $S = \{i \in V \cup R \mid m_i^v \neq 0, \mathrm{sign}(e_i) = s\}$. **Restart** if $S$ is empty.
4:     Let $k \leftarrow \prod_{i \in S} \ell_i$ and compute $T \leftarrow [(p+1)/k]P$.
5:     **for** $i \in S \cap V$ **do**
6:         Compute $Q \leftarrow [k/\ell_i]T$. If $Q = \infty$, **skip** this $i$ and set $S \leftarrow S - \{i\}$.
7:         Compute $\phi : E_A \to E_B$ with kernel $\langle Q \rangle$ using Vélu.
8:         When $e_i \neq 0$, set $A \leftarrow B$, $T \leftarrow \phi(T)$, $e_i \leftarrow e_i - s$
9:         Set $m_i^v \leftarrow m_i^v - 1$ and set $S \leftarrow S - \{i\}$
10:        If $m_i^v = 0$, set $V \leftarrow V - \{i\}$
11:    **for** $i \in S \cap R$ **do**
12:        Compute $Q \leftarrow [k/\ell_i]T$. If $Q = \infty$, **skip** this $i$.
13:        Compute $E_B = [\mathfrak{l}_i]^{e_i} \star E_A$ using $m_i^r$ radical $\ell_i$-isogenies
14:        Set $A \leftarrow B$, $R \leftarrow R - \{i\}$, and **start over** at line 1
15: **return** $A$.

---

Effectively, for $i \in V \cap R$, we first check if we can perform a Vélu $\ell_i$-isogeny in the loop in lines 5–9 (a Vélu block). If $m_i^v$ of these have been performed, we check in lines 11–14 if the 'left-over' point $Q$ has order $\ell_i$ for some $i \in S \cap R$, so that it can initiate the 'chain' of radical $\ell_i$-isogenies in line 13. Algorithm 1 does not go into the details of CSURF (i.e. using degree 2/4 isogenies), which can easily be added and does not interfere with the hybrid strategy. Furthermore, Algorithm 1 does not leak any timing information about the secret values $e_i$, only on $m_i$, which is public information. For simplicity's sake, we do not detail many lower-level improvements.

## 6   Implementation and Performance Benchmark

All the experiments presented in this section are centred on constant-time
CSIDH, CSURF and CRADS implementations, for a base field of 512-, 1024-,
1792-, 2048-, 3072-, and 4096-bits. To be more precise, in the first subsection
we restrict our experiments to i) the most competitive CSIDH-configurations
according to [12,15], ii) the CSURF-configuration presented in [7,8] and iii) the
radical isogenies-configuration presented in [8] (i.e. without the hybrid strategy).
As mentioned in Sect. 4, we only focus on dummy-based variants such as MCR-
style [18] and OAYT-style [20]. The experiments using only radical isogenies
of degree 2/4 are labelled CSURF, whereas the experiments using both radical
isogenies of degree 2/4 and 3/9 are labelled CRADS. In the second subsection,
we integrate the hybrid strategy to our experiments, and focus on dummy-based
OAYT-style implementations. This allows us to compare the improvement of the
hybrid strategy against the previous section. When comparing totals, we assume
one field squaring costs what a field multiplication costs ($\mathbf{S} = \mathbf{M}$). Primes used
are of the form $p = h \cdot \prod_{i=1}^{74} \ell_i - 1$, with $h = 2^k \cdot 3$. The key space is about $2^{256}$.

On the optimal exponent bounds (fixed number of $\ell_i$-isogenies required),
the results from [15] give $\approx 0.4\%$ of saving in comparison to [12] (see Table 5
from [12]). The results from [15] are mathematically rich: analysis on the per-
mutations of the primes and the (integer) convex programming technique for
determining an approximately optimal exponent bound. However, their cur-
rent Matlab-based code implementation from [15] only handles CSIDH-512 using
OAYT-style prioritizing multiplicative-based strategies. Both works essentially
give the approximate same expected running time, and by simplicity, we choose
to follow [12], which more easily extends to any prime size (for both OAYT
and MCR styles). Furthermore, all CSIDH-prime instantiations use the approx-
imately optimal exponent bounds presented in [12].

To reduce the cost of exponentiations in radical isogenies, we used short
addition chains (found with [17]), which reduces the cost from $1.5 \log(p)$ (from
square-and-multiply) to something in the range $[1.05 \log(p), 1.18 \log(p)]$. These
close-to-optimal addition chains save at least 20% of the cost of an exponentia-
tion used per (affine or projective) radical isogeny in constant-time.

Our CSURF and CRADS constant-time implementations evaluate the group
action by first performing the evaluation as CSIDH does on the floor of the
isogeny graph, with the inclusion of radical isogenies as in Algorithm 1. After-
wards we move to the surface to perform the remaining 4-isogenies. So, the only
curve arithmetic required is on Montgomery curves of the form $E/\mathbb{F}_p : By^2 =
x^3 + Ax^2 + x$. Concluding, we compare three different implementations which
we name CSIDH, CSURF and CRADS. The CSIDH implementation uses tra-
ditional Vélu's formulas to perform an $\ell_i$-isogeny for $\ell_i \leq 101$ and switches to
$\sqrt{}$élu for $\ell_i > 101$. The CSURF implementation adds the functionality of degree
2/4 radical isogenies, while the CRADS implementation uses radical isogenies of
degree 2/4 and 3/9.

### 6.1   Performance Benchmark of Radical Isogenies

We compare the performance using a different keyspace (i.e., different bounds $(e_i)$) for CSIDH, CSURF, and CRADS than in [7,8], where they have used weighted $L_\infty$-norm balls for CSURF and CRADS to compare against an unweighted $L_\infty$-norm ball for CSIDH. Analysis from [12,15,18] shows that such a comparison is unfair against CSIDH. We therefore use approximately optimal keyspaces (using weighted $L_\infty$-norm) for CSIDH, CSURF and CRADS.

*Suitable bounds.* We use suitable exponent bounds for approximately optimal keyspaces that minimize the cost of CSIDH, CSURF, and CRADS by using a slight modification of the greedy algorithm presented in [12], which is included in the provided repository. In short, the algorithm starts by increasing the exponent bound $m_2 \leq 256$ of two used in CSURF, and then applies the exponent bounds search procedure for minimizing the group action cost on the floor (the CSIDH computation part). Once having the approximately optimal bounds for CSURF, we proceed in a similar way for CRADS: this time $m_2$ is fixed and the algorithm increases the bound $m_3 \in [\![1 \mathinner{\ldotp\ldotp} m_2]\!]$ until it is approximately optimal.

*Comparisons.* The full results are given in Table 3. From Fig. 1a we see that CSURF and CRADS outperform CSIDH for primes of sizes 512 and 1024 bits, and is competitive for primes of sizes 1792 and 2048 bits. For larger primes, CSIDH outperforms both CSURF and CRADS. Using OAYT-style, CSURF-512 provides a speed-up over CSIDH-512 of 2.53% and CRADS-512 provides a speed-up over CSIDH-512 of 2.15%. The speed-up is reduced to 1.26% and 0.68% respectively for 1024 bits. For larger primes both CSURF and CRADS do not provide speed-ups, because radical isogenies scale worse than Vélu's (or $\sqrt{}$élu's) formulas (see Sect. 4.2). This is visible in Fig. 1a and Fig. 1b.



(a) OAYT-style          (b) MCR-style
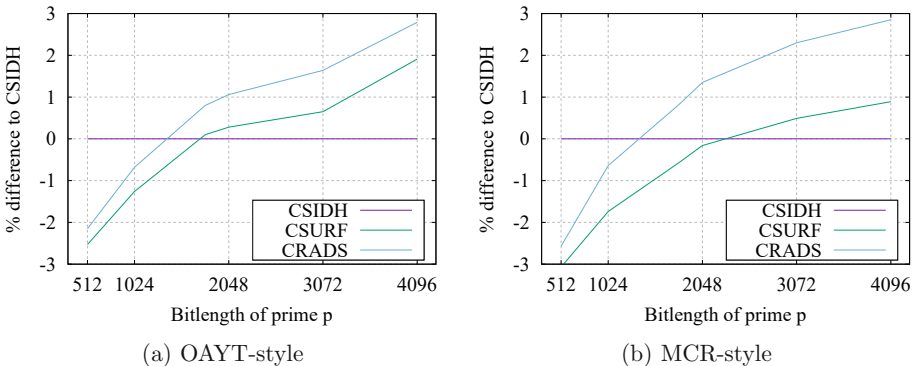
**Fig. 1.** Relative difference between the number of finite field multiplications required for CSURF and CRADS in comparison to CSIDH. The percentage is based on the numbers in Table 3.

**Table 3.** Results for different prime sizes. The numbers are given in millions of finite field multiplications, and the results are the average over 1024 runs. The results count multiplication (**M**) and squaring (**S**) operations, assuming **S** = **M**. Numbers in bold are optimal results for that prime size.

| Dummy-style | 512-bits | 1024-bits | 1792-bits | 2048-bits | 3072-bits | 4096-bits |
|---|---|---|---|---|---|---|
| CSIDH-OAYT | 0.791 | 0.873 | **0.999** | **1.039** | **1.217** | **1.361** |
| CSURF-OAYT | **0.771** | **0.862** | 1.000 | 1.042 | 1.225 | 1.387 |
| CRADS-OAYT | 0.774 | 0.867 | 1.007 | 1.050 | 1.237 | 1.399 |
| CSIDH-MCR | 1.011 | 1.093 | 1.218 | 1.255 | 1.436 | 1.580 |
| CSURF-MCR | 0.980 | 1.074 | 1.211 | 1.253 | 1.443 | 1.594 |
| CRADS-MCR | 0.985 | 1.086 | 1.228 | 1.272 | 1.469 | 1.625 |

Furthermore, the approximately optimal bounds we computed show that the exponents $m_2$ and $m_3$ decrease quickly: from $m_2 = 32$ and $m_3 = 12$ for 512-bits, to $e_0 = e_1 = 4$ for 1792-bits, to $e_0 = e_1 = 2$ for 4096-bits. When using MCR-style CSURF and CRADS are slightly more competitive, although the overall cost is significantly higher than OAYT-style. Table 3 presents the results obtained in this benchmark and highlights the best result per parameter set. Notice that CSURF outperforms CRADS in every parameter set, which implies that replacing Vélu 3-isogenies with radical 9-isogenies is not cost effective.

### 6.2   Performance of Radical Isogenies Using the Hybrid Strategy

In this subsection, we benchmark the performance of radical isogenies when the hybrid strategy of Sect. 5 is used to integrate radical isogenies into CSIDH. Concretely, in comparison to the previous benchmark, we allow a strategy using both Vélu and radical isogenies for degree 3/9. We denote this by CRAD-H, and used our code to estimate the expected running time. We compared this against the results from Table 3. As a corollary of Lemma 3, CRAD-H may effectively become a 'traditional' CSIDH implementation, when $m_i^r = 0$ turns out to be most optimal for all $i \in R$ (i.e. we get the trivial 'hybrid' paramaters of ordinary CSIDH). We see this happening for primes larger than 1024 bits due to the scaling issues with radical isogenies. Table 4 shows the results[4].

Interestingly, an implementation using non-trivial hybrid parameters outperforms any of the other implementations for a prime of 512 bits. This shows that radical isogenies *can* improve performance in constant-time when using the hybrid strategy. We conclude that replacing Vélu 3-isogenies with radical 9-isogenies is too costly, but adding radical 9-isogenies after Vélu 3-isogenies can be cost effective. However, due to the scaling issues of radical isogenies, the value $m_i^r$ quickly drops to 0 and we get trivial parameters for primes larger than 1024 bits, implying that traditional CSIDH performs better for large primes than any

---

[4] Only for OAYT-style. Using the code, we analysed the cost of $m_i^r$ projective radical isogenies and $m_i^v$ Vélu isogenies. The results are realistic; there are no extra costs.

**Table 4.** Results for different prime sizes. The first row is given in millions of finite field multiplications, counting multiplication (**M**) and squaring (**S**) operations, assuming **S** = **M**. The second row gives the speed-up in comparison to CSIDH. Trivial parameters are denoted by a superscript $t$.

| CRAD-H | 512-bits | 1024-bits | 1792-bits | 2048-bits | 3072-bits | 4096-bits |
|---|---|---|---|---|---|---|
| Performance | 0.765 | 0.861 | $0.999^t$ | $1.039^t$ | $1.217^t$ | $1.361^t$ |
| Speed-up | 3.3% | 2.2% | - | - | - | - |

implementation using radical isogenies, even when such an implementation does not have to sacrifice the cheap Vélu isogenies of small degree. We conclude that 'horizontally' expanding the degrees $\ell_i$ for large primes $p$ is more efficient than 'vertically' increasing the bounds $m_i$ using radical isogenies. Nevertheless, for small primes, our hybrid strategy for radical isogenies can speed-up CSIDH and requires only slight changes to a 'traditional' CSIDH implementation.

*Remark.* The CTIDH proposal from [3] is about twice as fast as CSIDH by changing the key space in a clever way: CTIDH reduces the number of isogenies by half compared to CSIDH, using this new key space, which requires the *Matryoshka structure* for Vélu isogenies. The radical part of CSIDH is isolated from the Vélu part, and this will be the same case for CTIDH, when using radical isogenies. The CTIDH construction decreases the cost of this Vélu part by approximately a factor half, but the radical part remains at the same cost. Hence, we heuristically expect that the speed-up of radical isogenies in CTIDH decreases from 3.3%, and may even become a slow-down.

## 7    Concluding Remarks and Future Research

We have implemented, improved and analyzed radical isogeny formulas in constant-time and have optimized their integration into CSIDH with our hybrid strategy. We have evaluated their performance against state-of-the-art CSIDH implementations in constant-time. We show that fully projective radical isogenies are almost twice as fast as affine radical isogenies in constant-time. But, when integrated into CSIDH, both CSURF and radical isogenies provide only a minimal speed-up: about 2.53% and 2.15% respectively, compared to state-of-the-art CSIDH-512. Furthermore, larger (dummy-based) implementations of CSURF and CRADS become less competitive to CSIDH as radical isogenies scale worse than Vélu or $\sqrt{\text{é}}$lu isogenies. In such instances ($\log(p) \geq 1792$ bits) the use of constant-time radical isogenies even has a negative impact on performance.

Our hybrid strategy improves this performance somewhat, giving better results for primes of size 512 and 1024, increasing the speed-up to 3.3%, and making radical 9-isogenies cost effective. For larger primes however, our hybrid approach shows that $\sqrt{\text{é}}$lu-isogenies quickly outperformed radical isogenies, as we get 'trivial' parameters for our hybrid strategy. We therefore conclude that,

for larger primes, expanding 'horizontally' (in the degrees $\ell_i$) is more efficient than expanding 'vertically' (in the bounds $m_i$).

Due to the large cost of a single exponentiation in large prime fields, which is required to compute radicals, it is unlikely that (affine or projective) radical isogenies can bring any (significant) speed-up. However, similar applications of modular curves in isogeny-based cryptography could bring improvements to current methods. Radical isogenies show that such applications do exist and are interesting; they might be more effective in isogeny-based cryptography in other situations than CSIDH. For example with differently shaped prime number $p$ or perhaps when used in Verifiable Delay Functions (VDFs).

# References

1. Adj, G., Chi-Domínguez, J., Rodríguez-Henríquez, F.: Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols. IACR Cryptology ePrint Archive, p. 1109 (2020). https://eprint.iacr.org/2020/1109
2. Azarderakhsh, R., et al.: Supersingular isogeny key encapsulation. In: Third Round Candidate of the NIST's Post-Quantum Cryptography Standardization Process (2020). https://sike.org/
3. Banegas, G., et al.: CTIDH: faster constant-time CSIDH. Cryptology ePrint Archive, Report 2021/633 (2021). https://ia.cr/2021/633
4. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. IACR Cryptology ePrint Archive **2020**, 341 (2020)
5. Bernstein, D.J., Yang, B.: Fast constant-time GCD computation and modular inversion. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(3), 340–398 (2019). https://doi.org/10.13154/tches.v2019.i3.340-398
6. Bonnetain, X., Schrottenloher, A.: Quantum security analysis of CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 493–522. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_17
7. Castryck, W., Decru, T.: CSIDH on the surface. In: Ding, J., Tillich, J.P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 111–129. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_7
8. Castryck, W., Decru, T., Vercauteren, F.: Radical isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 493–519. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_17
9. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11274, pp. 395–427. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15
10. Cervantes-Vázquez, D., Chenu, M., Chi-Domínguez, J.J., De Feo, L., Rodríguez-Henríquez, F., Smith, B.: Stronger and faster side-channel protections for CSIDH. In: Schwabe, P., Thériault, N. (eds.) LATINCRYPT 2019. LNCS, vol. 11774, pp. 173–193. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30530-7_9

11. Chávez-Saab, J., Chi-Domínguez, J., Jaques, S., Rodríguez-Henríquez, F.: The SQALE of CSIDH: square-root vélu quantum-resistant isogeny action with low exponents. IACR Cryptology ePrint Archive 2020, 1520 (2020). https://eprint.iacr.org/2020/1520
12. Chi-Domínguez, J., Rodríguez-Henríquez, F.: Optimal strategies for CSIDH. IACR Cryptology ePrint Archive **2020**, 417 (2020)
13. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006). http://eprint.iacr.org/2006/291
14. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014)
15. Hutchinson, A., LeGrow, J., Koziel, B., Azarderakhsh, R.: Further optimizations of CSIDH: a systematic approach to efficient strategies, permutations, and bound vectors. Cryptology ePrint Archive, Report 2019/1121 (2019). https://ia.cr/2019/1121
16. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
17. McLoughlinn, M.B.: Addchain: cryptographic addition chain generation in go. Github Repository (2020). https://github.com/mmcloughlin/addchain
18. Meyer, M., Campos, F., Reith, S.: On Lions and Elligators: an efficient constant-time implementation of CSIDH. In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 307–325. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_17
19. Nakagawa, K., Onuki, H., Takayasu, A., Takagi, T.: $l_1$-norm ball for CSIDH: optimal strategy for choosing the secret key space. Cryptology ePrint Archive, Report 2020/181 (2020). https://ia.cr/2020/181
20. Onuki, H., Aikawa, Y., Yamazaki, T., Takagi, T.: (Short Paper) a faster constant-time algorithm of CSIDH keeping two points. In: Attrapadung, N., Yagi, T. (eds.) IWSEC 2019. LNCS, vol. 11689, pp. 23–33. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26834-3_2
21. Onuki, H., Moriya, T.: Radical isogenies on montgomery curves. IACR Cryptology ePrint Archive 2021, 699 (2021). https://eprint.iacr.org/2021/699
22. Peikert, C.: He gives C-sieves on the CSIDH. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 463–492. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_16
23. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive 2006, 145 (2006). http://eprint.iacr.org/2006/145
24. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Adv. Math. Comm. **4**(2), 215–235 (2010)
25. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris Sér. A-B **273**, A238–A241 (1971)