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
[2]Radboud University, Nijmegen, The Netherlands

September 11, 2021

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 propose an improved evaluation strategy to save point samplings. 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 (and without) radical isogenies is reduced to about 2% in comparison to the fastest state-of-the-art constant-time CSIDH-512 implementation.

[*]jesus.dominguez@tii.ae
[†]krijn.reijnders@ru.nl

The performance is worse for larger primes, as radical isogenies scale worse than Vélu 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 [24, 23] 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 [16, 14, 2]. 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, however both of the works in [7] and [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. In particular this means 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, Bernstein *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, 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 ℓ_1 up to ℓ_{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 ℓ_i, then our key space corresponds with the integer exponent[1] vectors $(e_1, \ldots, e_{74}) \in$

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

$[\![-m \,..\, m]\!]$[74]. A dummy-based variant of constant-time CSIDH performs $|e_i|$ secret ℓ_i-isogenies and then proceeds by performing $(m - |e_i|)$ dummy-isogenies. The ℓ_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 $E[\pi - 1]$ 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 $|e_i|$ ℓ_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 ℓ_i-radical isogenies to be performed is $\frac{m_i}{2}$, whereas in constant-time implementations the number of isogenies of degree ℓ_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. 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 with regards to 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. an *improved evaluation strategy* for radical isogenies, which allows us to 're-use' torsion points that are used in the CSIDH group action

evaluation to initiate a 'chain' of radical isogenies. This saves the cost of having to sample a 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 of an implementation with 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 traditional Vélu 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 when the size of the base field grows, reducing the speed-up obtained by implementing CSURF and radical isogenies.

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

5. a *performance benchmark* of CSURF and radical isogenies in comparison to "traditional" CSIDH, in total finite field operations. We perform this comparison for six different parameter sets of 512, 1024, 1792, 2048, 3074, and 4096 bits. Our benchmark is more accurate than the original benchmarks from [7] and [8] and shows that the 5% and 19% speed-up (respectively) diminishes to roughly 2% 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. However, this speed-up reduces for larger primes: 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.

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. 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 Section 2, we recap the theoretical preliminaries on isogenies, CSIDH, CSURF, and radical isogenies. In Section 3, we introduce two improvements: fully projective radical isogenies, and an improved evaluation strategy for radical isogenies. In Section 4, we analyze the efficiency of constant-time radical isogenies assuming these improvements. In Section 5, we compare constant-time CSIDH, CSURF and CRADS in terms of finite field operations. Finally, in Section 6 we present our conclusions concerning the efficiency of radical isogenies in comparison to CSIDH.

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 φ 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 φ_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 $\sqrt{}$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 $\mathrm{End}(E)$. 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 $\mathrm{End}(E)$. 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: $\mathrm{End}(E)$ 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 $\mathrm{End}_p(E)$ of $\mathrm{End}(E)$, which are rational endomorphisms of a supersingular elliptic curve E. This subring $\mathrm{End}_p(E)$ is isomorphic to an order $\mathcal{O} \subset \mathcal{O}_K$. As both $[N]$ and π 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 Section 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

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

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 ℓ_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 ℓ_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 $[\![-m \,..\, m]\!]^n$ for some bound $m \in \mathbb{N}$. This was improved in [15, 18, 12] by varying the bound m per degree ℓ_i (a weighted L_∞-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_∞-norm ball for CSIDH-512 versus a weighted L_∞-norm ball for the implementation using radical isogenies. In this paper, to make a fair comparison to the previous work, we continue in the line of [15, 18, 12] by using *weighted* L_∞-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_∞-norm balls. The fact that radical isogenies can be made to have exactly the same cost per degree without dummy operations (with only slightly extra cost) make them 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 Equation (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 an N-order point with $N \geq 4$. Additionally, let $E_1 = E_0/\langle P_0 \rangle$, and P_1 an N-order point 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 φ_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 N-torsion point P to its Tate normal form, we can evaluate $E \to [\mathfrak{l}_i] \star E$ without sampling a points (except for sampling P). Specifically, it allows us to compute $E \to [\mathfrak{l}_i]^k \star E$ without having to sample k points of order N.

$$\begin{array}{ccccccc}
E & \xrightarrow{\text{Vélu}} & [\mathfrak{l}_i] \star E & \longrightarrow & \cdots & \longrightarrow & [\mathfrak{l}_i]^k \star E \\
\Big\downarrow & & & & & & \Big\uparrow \\
\text{To Tate normal form} & & & & & & \text{To Montgomery} \\
\Big\downarrow & & & & & & \Big| \\
E(b_0,c_0) & \xrightarrow[\varphi_N]{} & E(b_1,c_1) & \longrightarrow & \cdots & \longrightarrow & E(b_k,c_k)
\end{array}$$

Notice that the top row and the bottom row of the diagram are isomorphic. The map φ_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 ℓ_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 φ_N for small values of $N \in \{2,3,4,5,7,9,11,13\}$. For larger degrees the formulas

become too complex. 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 Improvements to radical isogenies

In this section we introduce two improvements to radical isogenies. First, in Section 3.1 we present *fully projective* radical isogenies, which 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 in projective coordinates. (a) requires non-trivial work which we explain in Section 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.

Second, in Section 3.2 we introduce an evaluation strategy that is an improvement over the original evaluation strategy for radical isogenies, by mixing the 'traditional' CSIDH group action evaluation with radical isogenies. This saves us having to sample a torsion point to initiate radical isogenies. Concretely, in 'traditional' CSIDH isogeny evaluation, one pushes a torsion point T through a series of ℓ-isogenies with Vélu's formulas. This implies that at the end, such a point T can 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.

Both techniques also improve performance in a non-constant implementation. However, in this section we assume a constant-time approach to compare cost, as this is a more realistic approach for a cryptographical implementation.

3.1 Fully projective radical isogenies

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. In this subsection 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 can save one of these exponentiations.

Lemma 3.1. *Let N be a natural number such that $\gcd(N, p-1) = 1$. Let $\alpha \in \mathbb{F}_p$. Write α 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 3.1. *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 ℓ_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 3.1 works for radical isogenies of degree $N = 4$. It is worth mentioning that 3-radical isogenies do not perform field inversions, and thus it is not required to write them in projective representation. We give three examples of these projective radical isogenies.

Example 3.1 (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*

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

In projective form, write α as $(X : Z)$ with $X, Z \in \mathbb{F}_p$. Then the projective

transformation becomes

$$\begin{aligned}(X:Z) &\mapsto (X'Z'^4 : Z') \quad \textit{with} \\ X' &= (4X^2+Z^2)XZ, \quad \textit{and} \\ Z' &= 2X+Z.\end{aligned} \tag{2}$$

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* δ *that is a quadratic residue in* $\mathbb{F}_p$*. We can force* δ *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, if we map to $(X'Z'^4 : Z')$ *then we can compute* $\sqrt[4]{-b'}$ *as* $(\sqrt[4]{X'Z'^4} : Z'^2)$ *using only one 4-th root. This allows us to repeat Equation* (2) *using only one exponentiation, without the cost of the inversion required in the affine version.*

Example 3.2 (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*

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

In projective form, write $\alpha = X/Z$ *with* $X, Z \in \mathbb{F}_p$ *and work with* $(X : Z)$*. Then the projective transformation becomes*

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

Notice that the image is $(X'Z'^4 : Z')$ *instead of* $(X' : Z') = (X'Z'^4 : Z'^5)$*, following Lemma 3.1. 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 Equation* (3) *using only one exponentiation, without the cost of the inversion required in the affine version.*

Example 3.3 (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 3.1. 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* $(\sqrt[7]{X^4Z^2(X-Z)} : Z)$*.*

For larger degrees, it becomes increasingly more tedious to work out the projective isogeny maps. In the code repository, we provide formulas for $N \in \{2, 3, 4, 5, 7, 9\}$. 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.

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

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 $\mathbf{S} = \mathbf{M}$ and ignoring **A**.

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

Table 2: Comparison between the cost of different functions to switch curve models, required to perform radical isogenies. Affine results from [8] and projective results from 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 $\mathbf{S} = \mathbf{M}$ and ignoring **A**.

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 Section 4.1, the radical isogenies of degree 5, 7, 11 and 13 still perform worse than 'traditional' Vélu isogenies in realistic scenarios.

3.2 Improved evaluation strategies for radical isogenies

In this subsection we describe an improved evaluation strategy that saves the cost of having to sample a torsion point to initiate a 'chain' of radical isogenies. As described in Section 2.4, to start off a chain of radical isogenies of degree ℓ on E_A, it is necessary to have an ℓ-torsion point P on E_A so that we can map (E_A, P) to the (isomorphic) Tate normal curve $E_{(b,c)}$ with

$P \mapsto (0,0)$. We write V (for Vélu) for the set of primes ℓ_i which we want to evaluate using traditional CSIDH isogeny evaluations such as Vélu and $R \subset \{2,3,5,7,11,13\}$ for the set of primes for which we use radical isogenies.

In [8], the implementation first performs radical isogenies for all $\ell \in R$ by sampling an appropriate ℓ-torsion point on the Montgomery curve, swap to the Tate normal form, perform the radical isogenies, and move back to the Montgomery representation. Afterwards, the rest of the ℓ-isogenies for $\ell \in V$ are performed using 'traditional' CSIDH methods.

Algorithm 1 High-level evaluation strategy for radical isogenies

Inputs: $A \in \mathbb{F}_p$, a list of integers $(e_1, \ldots, e_n)$, and two sets of primes V (Vélu isogenies) and R (radical isogenies), and $r = \prod_{i \in R} \ell_i$.
Output: $B \in \mathbb{F}_p$ such that $\prod [\mathfrak{l}_i]^{e_i} \star E_A = E_B$

while $e_i \neq 0$ for $i \in V$ **do**
  Sample $x \in \mathbb{F}_p$.
  Set $s \leftarrow 1$ if $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$, else $s \leftarrow -1$.
  Let $S = \{i \in V \mid e_i \neq 0, \mathrm{sign}(e_i) = s\}$. Restart if S is empty.
  Let $k \leftarrow r \cdot \prod_{i \in S} \ell_i$ and compute $T \leftarrow [(p+1)/k]P$.
  for $i \in S$ **do**
    Compute $Q \leftarrow [k/\ell_i]T$. If $Q = \infty$, skip this i.
    Compute $\phi : E_A \to E_B$ with kernel $\langle Q \rangle$ using Vélu.
    Set $A \leftarrow B$, $T \leftarrow \phi(T)$, and $e_i \leftarrow e_i - s$.
  end for
  if R is not empty **then**
    Select one j from R such that $Q = [r/\ell_j]T$ is different from ∞
    Compute $E_B = [\mathfrak{l}_j]^{e_j} \star E_A$ using radical ℓ_j-isogenies.
    Set $A \leftarrow B$, and $R \leftarrow R - \{j\}$ and start over.
  end if
end while
for $i \in R$ **do**
  Sample an ℓ_i-torsion point Q
  Compute $E_B = [\mathfrak{l}_i]^{e_i} \star E_A$ using radical ℓ_i-isogenies.
  Set $A \leftarrow B$, and $R \leftarrow R - \{i\}$.
end for
return A.

We improve upon this strategy by mixing the 'traditional' CSIDH method with radical isogenies. Concretely, in 'traditional' CSIDH isogeny evaluation, one pushes a torsion point T through a series of ℓ-isogenies with Vélu's formulas, for $\ell \in V$. This implies that at the end, such a point T can still have ℓ-torsion with $\ell \in R$. In such a case, we do not need to sample a point, but can reuse T to instantiate the chain of radical isogenies of degree ℓ. Notice that we can only use one torsion point T to set of one chain of radical isogenies, even if T might have more torsion, as we do not push T

through the radical isogenies. With high probability, T will have ℓ-torsion at least once for every $\ell \in R$ during 'traditional' CSIDH evaluation. This approach does not leak any information with regards to the exponent e_i of ℓ_i, when using dummy isogenies. This strategy is summarized in high-level in Algorithm 1. The algorithmic description ignores other improvements (e.g. optimal strategies) for simplicity's sake. Notice that the last for loop of the algorithm will almost never happen due to the high probability of T having the correct torsion at least once in the main while loop.

4 Cost analysis of constant-time radical isogenies

In this section, we analyze the number of finite field operations required in a constant-time implementation of CSURF or radical isogenies. In a simplified model, the cost of performing n radical isogenies of degree ℓ using Tate normal curves can be divided into 4 steps.

1. Sample a point P on E_A of order ℓ;
2. Map the pair (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 step, the cost is dominated by the number of exponentiations (**E**) and inversion (**I**). Using Tables 2 and 3, in a straightforward constant-time implementation, step 2 will cost 1 **E** + 1 **I**, step 3 costs approximately 1 **E** + 1 **I** per isogeny, and step 4 will cost 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[2]: $\alpha^{-1} = \alpha^{p-2}$, 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

[2] There is also a new constant-time inversion based on gcd-computations by Bernstein and Yang [5].

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 Section 3.1 to by-pass all inversions completely for radical isogenies.

Approximate cost of radical isogenies. We can now approximate the cost of evaluating radical isogenies in constant-time, using the improvements introduced in Section 3. The evaluation strategy from Section 3.2 shows that we can avoid (most of) the cost of step 1. 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 **E**. In total, performing n radical ℓ-isogenies therefore costs approximately $(n+4)$**E**.

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

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

Lemma 4.1. *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 the cost of $(n+4) \cdot \alpha \cdot \log_2(p)$ from Lemma 4.1 for radical isogenies is too high and it is therefore not worthwhile to perform radical isogenies in comparison to Vélu 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 √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 [4] and [1]);

2. The group action evaluation first performs one block using √élu isogeny formulas, and then isolates the radical isogeny computations. What is particularly important among these √élu isogeny computations, is that removing one specific ℓ'-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 ℓ_i-isogenies such that $\left[\frac{p+1}{\ell_i}\right] R \neq \mathcal{O}$.

3. Replacing the smallest Vélu ℓ-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 ℓ is $\frac{\ell-1}{\ell}$, which is small for small ℓ. 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 ℓ costs

$$\begin{aligned}\texttt{sampling}(n,p,\ell) &= 9\left\lfloor\frac{n\ell}{\ell-1}\right\rceil \log_2\left(\frac{p+1}{\ell}\right)\mathbf{M}\\ &\approx 9\left\lfloor\frac{n\ell}{\ell-1}\right\rceil (\log_2(p)-\log_2(\ell))\mathbf{M}.\end{aligned}$$

   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' ℓ'-isogenies and $\left\lfloor\frac{n\ell}{\ell-1}\right\rceil \geq \left\lfloor\frac{n'\ell'}{\ell'-1}\right\rceil$. Then the savings are given by their difference with respect to the cost sampling such torsion-points (see Equation (4)).

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

   Whenever $\left\lfloor\frac{n\ell}{\ell-1}\right\rceil < \left\lfloor\frac{n'\ell'}{\ell'-1}\right\rceil$, 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 [20] and [12]. In both works, $\log_2(p) \approx 512$ and the first five smallest primes ℓ_i's in $\{3,5,7,11,13\}$ have bounds m_i that satisfy

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

Thus, there are no savings concerning sampling of points when including small degree radical isogenies. Clearly, performing n ℓ-radical isogenies becomes costlier than using $\sqrt{\text{élu}}$ isogenies, and thus the above analysis suggests radical isogenies need their own optimal bounds to be competitive. For this reason, we look for optimal bounds for radical isogenies as follows: i) we set as initial bounds the optimal ones of CSIDH-512, ii) we set to zero the exponent e of the ℓ-radical isogeny, iii) we increase (as much as possible) the radical bound e, and iv) we apply a greedy-based algorithm to reduce the $\sqrt{\text{élu}}$-isogeny bounds.

The analysis is different for degree 2 and 3, where we can perform 4-and 9-isogenies in $\lceil \frac{n}{2} \rceil$ radical computations instead of n computations. In fact, 4-isogenies directly reduces the sampling of points by decreasing the bounds of the other primes ℓ_i's. Nevertheless, performing n radical isogenies takes at least $(n+4)\log_2(p)$ field multiplications (Lemma 4.1), 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 ℓ_i, as we have shown in Section 3.2 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 do not scale well to larger primes, as their cost scales with $\log(p)$, and dummy-free isogenies are more expensive, as we will need to switch direction often to perform 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 Section 5, we demonstrate this effect using a benchmark we have performed on CSIDH, CSURF and CRADS for six different prime sizes, from 512 bits up to 4096 bits. These prime sizes are realistic: several analyses, such as [6, 22, 11], 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 Implementation and performance benchmark

All the experiments presented in this section are centred on constant-time CSIDH, CSURF and CRADS instantiations with 512-, 1024-, 1792-, 2048-, 3072-, and 4096-bits. To be more precise, we restrict our experiments to i) the most competitive CSIDH-configurations according to [15, 12], ii) the CSURF-configuration presented in [7, 8] and iii) the radical isogenies-configuration presented in [8]. As mentioned in Section 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. When comparing totals, we assume one field squaring costs what a field multiplication costs ($\mathbf{S} = \mathbf{M}$). The 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 size is about 2^{256}.

On the optimal exponent bounds (fixed number of ℓ_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 permutations of the primes and the (integer) convex programming technique for

determining an approximately optimal exponent bound. However, their current 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 approximately 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 exponentiation used per (affine or projective) radical isogeny in constant-time.

Our CSURF and CRADS constant-time implementations evaluate the group action by doing first performing the evaluation as CSIDH does on the floor of the isogeny graph, with the inclusion of radical isogenies as in Algorithm 1. Afterwards 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 traditional Vélu's formulas to perform an ℓ_i-isogeny for $\ell_i \leq 101$ and switches to $\sqrt{\text{élu}}$ 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 $3/9$.

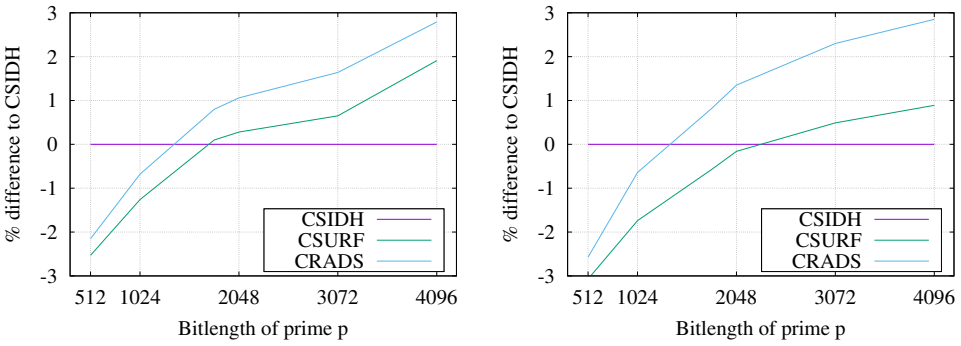
5.1 Performance benchmark

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_∞-norm balls for CSURF and CRADS to compare against an unweighted L_∞-norm ball for CSIDH. As analysis from [15, 18, 12] shows, such a comparison is unfair against CSIDH. We therefore use approximately optimal keyspaces (using weighted L_∞-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 summary, the greedy 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 exponent bound $m_3 \in [\![1 \,..\, m_2]\!]$ until it is approximately optimal.

Comparisons. The full results are given in Table 3. From Figure 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 √élu's) formulas (see Section 4.2). This is visible in Figure 1a and Figure 1b.

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.



(a) OAYT-style (b) MCR-style

Figure 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 number of multiplications required, and corresponds with an average over 1024 runs, assuming $\mathbf{S} = \mathbf{M}$.

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

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 ($\mathbf{M}$) and squaring ($\mathbf{S}$) operations, assuming $\mathbf{S} = \mathbf{M}$. Numbers in bold are optimal results for that prime size.

6 Concluding remarks and future research

We have implemented and improved the radical isogenies formulas and evaluation strategy, and evaluated their performance against state-of-the-art CSIDH implementations in constant-time. In theory, fully projective radical isogenies are almost twice as fast as affine radical isogenies in a constant time implementation. However, when integrated as CSURF and CRADS, radical isogenies provide only a minimal speed-up for a prime of 512 bits: about 2.53% and 2.15%, respectively (compared to state-of-the-art CSIDH). Furthermore, larger (dummy-based) implementations of CSURF and CRADS become less competitive to CSIDH as radical isogenies scale worse than Vélu and $\sqrt{}$élu isogenies. In such instances ($\log(p) \geq 1792$ bits) the use of constant-time radical isogenies even has a negative impact on performance. 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 much more effective in isogeny-based cryptography in other situations than CSIDH (or perhaps with different prime number shapes).

Acknowledgements. The authors would like to thank Simona Samardjiska, Peter Schwabe, Francisco Rodríguez-Henríquez, and anonymous reviewers for their valuable comments that helped to improve the technical material of this paper. The European Commission initially supported this work through the ERC Starting Grant 804476 (SCARE), while J. J. Chi-Domínguez was a postdoctoral researcher at Tampere University.

References

[1] Gora Adj, Jesús-Javier Chi-Domínguez, and Francisco Rodríguez-Henríquez. Karatsuba-based square-root Vélu's formulas applied to two isogeny-based protocols. *IACR Cryptol. ePrint Arch.*, page 1109, 2020.

[2] Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik. Supersingular isogeny key encapsulation. *Third round candidate of the NIST's post-quantum cryptography standardization process, 2020.*, 2020.

[3] Gustavo Banegas, Daniel J. Bernstein, Fabio Campos, Tung Chou, Tanja Lange, Michael Meyer, Benjamin Smith, and Jana Sotáková.

Ctidh: faster constant-time csidh. Cryptology ePrint Archive, Report 2021/633, 2021. https://ia.cr/2021/633.

[4] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *IACR Cryptol. ePrint Arch.*, 2020:341, 2020.

[5] Daniel J. Bernstein and Bo-Yin Yang. Fast constant-time gcd computation and modular inversion. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):340–398, 2019.

[6] Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, volume 12106, pages 493–522, 2020. https://eprint.iacr.org/2018/537.

[7] Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 2020.

[8] Wouter Castryck, Thomas Decru, and Frederik Vercauteren. Radical isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 493–519. Springer, 2020.

[9] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

[10] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. Stronger and faster side-channel protections for CSIDH. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 173–193. Springer, 2019.

[11] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: square-root

vélu quantum-resistant isogeny action with low exponents. *IACR Cryptol. ePrint Arch.*, 2020:1520, 2020.

[12] Jesús-Javier Chi-Domínguez and Francisco Rodríguez-Henríquez. Optimal strategies for CSIDH. *IACR Cryptol. ePrint Arch.*, 2020:417, 2020.

[13] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `http://eprint.iacr.org/2006/291`.

[14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Math. Cryptol.*, 8(3):209–247, 2014.

[15] Aaron Hutchinson, Jason LeGrow, Brian Koziel, and Reza Azarderakhsh. 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] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011.

[17] Michael B. McLoughlinn. addchain: Cryptographic addition chain generation in go. Github Repository, 2020.

[18] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 307–325. Springer, 2019.

[19] Kohei Nakagawa, Hiroshi Onuki, Atsushi Takayasu, and Tsuyoshi Takagi. 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] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. (short paper) A faster constant-time algorithm of CSIDH keeping two points. In Nuttapong Attrapadung and Takeshi Yagi, editors, *Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019, Tokyo, Japan, August 28-30, 2019, Proceedings*, volume 11689 of *Lecture Notes in Computer Science*, pages 23–33. Springer, 2019.

[21] Hiroshi Onuki and Tomoki Moriya. Radical isogenies on montgomery curves. *IACR Cryptol. ePrint Arch.*, 2021:699, 2021.

[22] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, volume 12106, pages 463–492, 2020. `https://eprint.iacr.org/2019/725`.

[23] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptology ePrint Archive*, 2006:145, 2006.

[24] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. in Math. of Comm.*, 4(2):215–235, 2010.

[25] Jacques Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. A-B*, 273:A238–A241, 1971.