Don't forget the constant-time in CSURF

Jesús-Javier Chi-Domínguez [*1] and Krijn Reijnders [†2]

[1]Tampere University, Tampere, Finland
[2]Radboud University, Nijmegen, The Netherlands

March 3, 2021

Abstract

CSURF (CSIDH on the surface) was recently proposed by Castryck, and Decru in PQCrypto-2020, and then improved by the radical isogeny formulæ in Asiacrypt-2020. The main advantage of using CSURF and radical isogenies is the possibility of using isogenies of degree two and radical isogeny chains of odd degree requiring only a single random sampling of points. This work addresses the practical implications of a constant-time implementation of CSURF and the radical isogeny procedures. In particular, this paper introduces the first constant-time formulation and implementation of the radical isogenies using projective representation, which are twice as efficient as the original radical isogeny formulæ. Nevertheless, the overhead introduced by going to constant-time is significant: in terms of finite field operations, our experiments illustrate that the speed-up of using a constant-time CSURF-512 is reduced to 1.64% in comparison to the fastest state-of-the-art constant-time CSIDH-512 implementation. Furthermore, these savings disappear when using constant-time radical isogenies and when moving to higher parameter sets. This negatively answers the open question from Castryck and Decru: constant-time CSIDH implementations outperform both CSURF and radical isogenies.

1 Introduction

The first proposal of an isogeny-based Diffie-Hellman key exchange was done by Couveignes [9] and centered on the action of an ideal class group on a set of ordinary elliptic curves. But, it wasn't until Rostovtsev and Stolbunov [16, 15] independently rediscovered it and recognized its potential as

[*]jesus.chidominguez@tuni.fi
[†]reijnderskrijn@gmail.com

a strong post-quantum candidate. After this, isogeny-based cryptography developed further with SIDH in [11, 10, 1]. In Asiacrypt 2018, Castryck, Lange, Martindale, Panny, and Renes introduced CSIDH (as a possible improvement to SIDH) by reformulating Couveignes' system by focusing on supersingular curves defined over a prime field [6] and only using odd degree isogenies. With the hope to improve CSIDH performance, Castryck and Decru proposed CSURF by giving an interesting way of exploiting and performing degree-2 isogenies [4] by moving to the surface of the isogeny graph. Indirectly hinted by CSURF's nature, Castryck, Decru, and Vercauteren presented in Asiacrypt 2020 a couple of new ideas for constructing isogenies with small odd degree based on radical computations (N-th roots) [5]. With this they gained a speed-up of about 19% over CSIDH. It is worth mentioning that both of the works in [4] and [5] focused 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.

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 [13], and then improved by Onuki *et al.* in [14]. Subsequently, Cervantes-Vázquez *et al.* proposed a dummy-free variant of CSIDH [7]. In summary, all of the previously mentioned constant-time implementations of CSIDH perform a fixed number of isogeny constructions.

Let's explain the general idea of how a constant-time CSIDH implementations works by using CSIDH-512 as an example. That is, we use the 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 $\ell_{74} = 587$. Next, let $E/\mathbb{F}_p \colon y^2 = x^3 + Ax^2 + x$ be a supersingular Montgomery curve with $(p+1)$ rational points. Additionally, assume we require exactly $m = 5$ isogenies per ℓ_i, then our keyspace corresponds with the integer exponent vectors $(e_1, \ldots, e_{74}) \in [\![-m \,..\, m]\!]^{74}$ [1]. Finally, we have all the ingredients for describing what the main block of CSIDH is. In a dummy-based variant, one starts performing $|e_i|$ secret degree-ℓ_i isogenies and then proceeds by computing $(m - |e_i|)$ dummy-isogenies. The degree-ℓ_i isogeny kernel belongs to either $E[\pi - 1]$ or $E[\pi + 1]$ (the sign of e_i determines which one will be used). Now, the dummy-free variant removes the $(m - |e_i|)$ dummy-isogeny constructions by assuming e_i has same parity as m, and alternatingly uses kernels in $E[\pi - 1]$ and $E[\pi + 1]$.

Contributions. We have investigated constant-time implementation of CSURF and radical isogenies, which claimed to be 5% and 19% faster than CSIDH in a non-constant-time setting, and we have explored different ap-

[1] The word *exponent* comes from the associated group action (see section 2)

proaches to constant-time implementations. More specifically, the main contributions of this work are the new projective radical isogeny formulæ and the first constant-time implementation of CSURF and chain of projective radical isogenies of degree up to nine. The proposed projective radical isogeny formulæ are almost twice as efficient as the original (affine) radical isogeny formulæ in constant time. We have also provided further optimizations by speeding up the costly exponentiations used in radical isogenies, and we have added the optimal bounds and optimal strategies as in [8]. Our Python-code implementation allows isogeny evaluation strategies using both traditional and Vélu square-root formulæ, as well as radical isogenies and degree-2 isogenies (on the surface), and is freely available at

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

The provided implementation allows us to compare CSURF and radical isogenies against state-of-the-art constant-time implementations of CSIDH. We have performed this comparison both theoretically and practically using six different parameter sets of 512-, 1024-, 1792-, 2048-, 3074-, and 4096-bits. We show that in low parameter sets, with the additional cost of moving to constant-time, CSURF performs on par with current CSIDH implementations, with a 1.64% speed-up for 512 bits and a 0.34% speed-up for 1024 bits. In high parameter sets, or when using radical isogenies, there is no speed-up in comparison to CSIDH. This negatively answers an open question in [5]: the impact of moving to constant-time is significant for CSURF and radical isogenies, and we do not achieve a speed-up by implementing it. In high parameter sets, we show that the cost of CSURF and radical isogenies scale worse than those of current CSIDH implementations. Our results illustrate that constant-time CSURF and radical isogenies perform worse than large CSIDH instantiations, at least at the level of finite field operations.

Outline. Section 2 recaps the theoretical preliminaries on isogenies, the Tate normal form, CSIDH, CSURF and radical isogenies. Our constant-time proposed affine and projective radical isogenies formulæ are presented in Section 3, as well as the use of fast exponentiations with addition chains. In addition, Section 3 also focuses on the performance impact of moving to constant-time implementations and concludes with a theoretical cost analysis of degree-2 and odd degree radical isogeny constructions. Our experiments in comparing CSIDH with CSURF and radical isogenies are described in Section 4. Finally, Section 5 presents the concluding remarks and lists a number of open problems.

2 Preliminaries

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

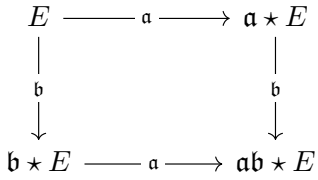
Given two elliptic curves E and E' over a prime field $\mathbb{F}_p$, an isogeny is a morphism $\varphi : E \to E'$ such that $\mathcal{O}_E \mapsto \mathcal{O}_{E'}$. A separable isogeny φ has a degree $\deg(\varphi)$ equal to the size of its kernel, and for any isogeny $\varphi : E \to E'$ a there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

Vélu's formulæ [17] provide the construction and evaluation of separable isogenies with cyclic kernel $G = \langle P \rangle$ for some N-torsion point $P \in E$. Both the isogeny construction of φ_G and the evaluation of $\varphi_G(R)$ for $R \in E$ 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 [2]. This new approach is based on multi-evaluations of a given polynomial and at its heart is still the traditional Vélu's formulæ.

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 $\mathcal{O} \xrightarrow{\sim} \mathbb{Z}[\pi]$ 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. Now, 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 a smaller commutative subring $\mathrm{End}_p(E) \subset \mathrm{End}(E)$ of isogenies of a supersingular elliptic curve defined over a prime field $\mathbb{F}_p$. In addition, $\mathrm{End}_p(E) \xrightarrow{\sim} \mathcal{O} \subset \mathcal{O}_K$ and both $[N]$ and π are defined over $\mathbb{F}_p$, which implies $\mathbb{Z}[\pi] \subset \mathrm{End}_p(E)$. 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 for some 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 $\mathfrak{a} \star E$ also in $\mathcal{Ell}_p(\mathcal{O})$ (see section 2.2). Furthermore, the CSIDH group action is believed to be a *hard homogeneous space* [9] that allows a Merkle-Diffie-Hellman-like key agreement protocol with commutative diagram



The original CSIDH protocol uses the set $\mathcal{E}\ell\ell_p(\mathcal{O})$ with $\mathcal{O} \xrightarrow{\sim} \mathbb{Z}[\pi]$ and $p = 3 \mod 4$. Now, CSURF protocol was designed to benefit from degree-2 isogenies by switching to the elliptic curves on the *surface* of the isogeny graph by using $\mathcal{E}\ell\ell_p(\mathcal{O})$ for order $\mathcal{O} \xrightarrow{\sim} \mathbb{Z}[\frac{1+\pi}{2}]$ and $p = 7 \mod 8$.

2.2 The group action of CSIDH and CSURF

The traditional way of evaluating the group action of an element $\mathfrak{a} \in \mathcal{C}\ell(\mathcal{O})$ is by using traditional [17] or square-root [2] Vélu's formulæ. 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. Moreover, $[\mathfrak{a}] \star E = E/E[\mathfrak{a}]$ and

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

In both CSIDH and CSURF, we apply specific elements $\mathfrak{l}_i \in \mathcal{C}\ell(\mathcal{O})$ 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 formulæ with input point P. On the other hand, CSURF takes advantage of changing the order $\mathcal{O}$ to also perform degree-2 isogenies on the surface of the isogeny graph; these degree-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.

2.3 The Tate normal form

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.

$$E(b,c)/\mathbb{F}_p \colon y^2 + (1-c)x - by = x^3 - bx^2 \tag{1}$$

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 elliptic 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 $\mathbb{A}^2 - \{\Delta = 0\}$.

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 degree-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 theoretical existence of a function φ_N that maps (b_0,c_0) to (b_1,c_1) [5]. Such a φ_N can be applied iteratively to compute a chain of degree-N isogenies without sampling points of order N. As a consequence, writing a given supersingular Montgomery curve $E/\mathbb{F}_p$ into Tate normal form allows a generalization of the idea of evaluating $E \to [\mathfrak{l}_i] \star E$ without sampling a point every time (we only require one initial point of order ℓ_i to move to the right Tate normal form). Specifically, it allows us to compute $E \to [\mathfrak{l}_i]^k \star E$ without having to sample k points of order N.

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

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 some suitable cofactor h. Castryck, Decru, and Vercauteren provided the explicit formulæ of φ_N for small values of $N \in \{2,3,4,5,7,9,11,13\}$, for larger degrees the formulæ become too complex. They also suggest the use of radical isogenies of degree 4 and 9 instead of 2 and 3, respectively.

3 Constant-time radical isogenies

The experiments presented in [5] suggest a speed-up of about 19% when using radical isogenies instead of Vélu's formulæ (for a prime of 512 bits). However,

these experiments focused on a non-constant-time Magma implementation for both the group action evaluation and the chain of radical isogenies. This implies that the radical isogeny Magma implementation computes exactly $|e_i|$ degree-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 instances of CSURF the average number of degree-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. Furthermore, in the original implementation of [5] the field inversion is performed in variable time depending on the input, as Magma prioritizes speed above constant-time.

In this section we analyse the theoretical cost of radical isogenies in constant-time. We describe three major performance impacts of moving to constant-time and higher parameter sets: constant-time field inversions cost what an exponentiation costs, exponentiations do not scale well to larger primes, and dummy-free isogenies are more expensive. Part of this impact can be remedied by two improvements: firstly, by moving to a projective version of radical isogenies to save an expensive inversion per iteration; secondly, by decreasing the cost per exponentiation using efficient addition chains. Notice, the use of addition chains and projective radical isogenies can also be used in the original implementation of [5] to provide a small speed-up.

3.1 Performance of radical isogenies in constant time

In a constant-time implementation one can compute the inverse of an element $\alpha \in \mathbb{F}_p$ by Fermat's little theorem: $\alpha^{-1} = \alpha^{p-2}$ [2]. Therefore, inversion becomes as costly as exponentiation. This almost doubles the cost of CSURF and radical isogenies in low degrees (2, 3, 4, 5, 7) and significantly increases the cost of radical isogenies of degree 9, 11 and 13, and the overhead of switching to Tate normal form and back to Montgomery form, and then performing e_N radical isogenies becomes less effective.

The cost of a radical isogeny of degree N is dominated by the cost of one N-th root and one inversion. On average an exponentiation costs $1.5\log(p)$ multiplications (using the square and multiply method). For radical isogenies, specific fixed exponentiations are required instead of generic ones, which makes it possible to decrease their cost by using short addition chains. Each N-th root and the inversion in $\mathbb{F}_p$ correspond to an exponentiation by a specific μ_N (with μ_{-1} for the inversion). We therefore decided to look for a close-to-optimal addition chain using [12], which reduces the cost to something in the range $[1.05\log(p), 1.19\log(p)]$. These close-to-optimal addition chains save at least 20% and at most 30% of the cost of an exponentiation,

[2] There is also a new Constant-time inversion based on gcd computations by Bernstein and Yang [3], but in this work we remove the inverse computaions required.

bringing the total cost of an (affine) radical isogeny in constant-time to the range of $[2.1\log(p), 2.4\log(p)]$.

Dummy-free radical isogenies. Recall, radical isogenies require sampling an initial N-order point P 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 us to sample $m_i - |e_i|$ points. That is, a dummy-free implementation of a chain of radical isogenies will require sampling at least $(m_i - |e_i|)$ points, and this will not be a constant-time procedure. We can make this procedure constant-time by sampling m_i points every time, but this costs too much and kills the idea of radical isogenies. Clearly, these costs may be decreased by pushing points through radical isogenies, which is still an open problem. In any case, we will only focus on dummy-based implementations of radical isogenies.

3.2 Decreasing cost using projective coordinates

The original (affine) radical isogenies cost approximately two exponentiations (the N-th root and the inversion) per iteration. In this subsection we show that one exponentiation can be saved by performing the radical isogeny with projective coordinates, at the cost of a few extra multiplications per iteration. After a chain of radical isogenies, one inversion is required to go back to affine coordinates.

One can do a straightforward translation to projective coordinates for radical isogenies. Such an approach saves an inversion by writing the Tate normal parameter b (respectively c) as $(X : Z)$, but comes at the cost of having to calculate both $\sqrt[N]{X}$ and $\sqrt[N]{Z}$ in the next iteration. We can save one exponentiation using the following lemma.

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 $(XZ^{N-1} : Z^N)$ in projective coordinates with $X, Z \in \mathbb{F}_p$. Then $\sqrt[N]{\alpha} = (\sqrt[N]{XZ^{N-1}} : Z)$.*

Proof. As N is co-prime with $p-1$, 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 $\sqrt[N]{Z^N} = Z$. □

Corollary 3.1. *The representation $(XZ^{N-1} : Z^N)$ brings the cost of a projective radical isogeny of small degree ℓ_i down to below $1.25\log(p)$ as it saves an exponentiation in the calculation of a radical isogeny in projective coordinates. Moreover, compared with the original affine radical isogeny formulæ,*

which roughly cost two exponentiations, our projective formulæ cost half of the affine ones.

The effect this has for small degrees can be seen in Table 1a. A similar approach as Lemma 3.1 works for radical isogenies of degree $N = 4$. It is worth mentioning that degree-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 [5], 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*

$$\begin{aligned} (X : Z) &\mapsto (X'Z'^4 : Z') \\ X' &= (4X^2 + Z^2)XZ \\ Z' &= 2X + Z \end{aligned} \tag{2}$$

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 [5] 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 [5] 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

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

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

Projective isogenies of degree $N \geq 9$**.** As mentioned by Castryck, Decru, and Vercauteren in [5], the effectiveness of using radical isogenies is most noticeable for small degrees such as 2,3,4,5,6, and 7. This is also the case for their projective versions. For that reason, we do not focus on implementing projective radical isogenies of degrees larger than 7. We limit our implementation to also include the affine radical isogenies of degree nine.

3.3 Summary of cost in field operations

Addition chains and projective coordinates give a decrease in cost of constant-time radical isogenies from about $3\log(p)$ to less than $1.25\log(p)$. Table 1 shows the exact cost of such a single constant-time radical isogeny and their overhead to move between the curve models. These numbers allow us to compute the theoretical cost of k radical isogenies for a specific degree and compare this to the number of bits of security this provides. For example, for a prime of 512 bits we have a per bit of security the theoretical optimal bounds $e_2 = 6$, $e_3 = 5$, $e_4 = 8$, $e_5 = 5$, $e_7 = 5$ and $e_9 = 3$ (OAYT-style, excluding the cost of point sampling). For low parameter sets, this is around 2000 finite field operations per bit of security for the most efficient isogeny, but high parameter sets give something close to 15000 finite field operations per bit of security.

From Table 1, CSURF-512 (as it uses only degree 2 isogenies) should be very competitive against the fastest CSIDH-512 instantiations. Nevertheless, large CSURF instantiations are expected to be slower than any of CSIDH,

Degree	**512**	**1024**	**1792**	**2048**	**3072**	**4096**
2	607	1114	1888	2144	3165	4192
3	606	1162	1934	2216	3243	4285
4	612	1119	1893	2149	3170	4197
5	619	1164	1967	2230	3264	4288
7	638	1190	1965	2245	3282	4320
9	1254	2354	3948	4465	6553	8607

(a) Radical isogeny cost.

Degree	**512**	**1024**	**1792**	**2048**	**3072**	**4096**
2	5469	10188	17262	19582	28851	38102
3	3677	6846	11542	13112	19284	25465
4	7910	14735	24959	28311	41704	55067
5	4292	8007	13504	15334	22547	29757
7	4287	8002	13499	15329	22542	29752
9	4290	8005	13502	15332	22545	29755

(b) Overhead cost

Table 1: Number of finite field operations required for (a) a radical isogeny of a certain degree and (b) their corresponding overhead when moving to curve models. It considers only multiplication (**M**) and squaring (**S**) operations, and assumes $\mathbf{S} = \mathbf{M}$.

because of the high cost of the exponentiation (at least in the current approach). Effectively, this implies the optimal bound for degree 2 becomes 0. Due to the increased cost per radical isogeny in constant time, radical isogenies of degree 3 to 9 seem to cost more than the traditional Velu's formulæ.

4 Experimental results

All the experiments presented in this section are centred on constant-time CSIDH and CSURF 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 [8], ii) the CSURF-configuration presented in [4] and iii) the radical isogenies-configuration presented in [5]. As mentioned in section 3, we only focus on dummy-based variants such as MCR-style [13] and OAYT-style [14]. The experiments regarding CSURF using radical isogenies are labelled by CRADS, and we assume one field squaring costs what a field multiplication costs. The primes used are of the form $p = h \cdot \prod_{i=1}^{74} \ell_i - 1$, with $h = 2^k$ or $h = 2^k \cdot 3$, and the key space size is about 2^{256}. All the CSIDH instantiations use the optimal exponent bounds presented in [8].

Our CSURF and CRADS constant-time implementations were done by doing first a group action as CSIDH does on the floor, then performing the 2-isogenies on the surface and finishing with the radical isogenies on the floor if applicable. So, the only curve arithmetic required is on Montgomery curves of the form $E/\mathbb{F}_p : By^2 = x^3 + Ax^2 + x$. We noticed no performance improvements in using degree 4 radical isogenies in comparison to degree 2 isogenies, due to the larger overhead of performing degree 4 radical isogenies in constant time. We therefore use degree 2 isogenies on the surface.

Concluding, we compare three different implementations which we name CSIDH, CSURF and CRADS. The CSIDH implementation uses traditional Vélu's formulæ to perform an ℓ_i-isogeny for $\ell_i \leq 101$ and switches to square-

root Vélu for $\ell_i > 101$. The CSURF implementation adds the functionality of degree 2 isogenies on the surface. The CRADS implementation uses degree 2 isogenies and uses radical isogenies to compute the isogenies of degree 3, 5 and 7.

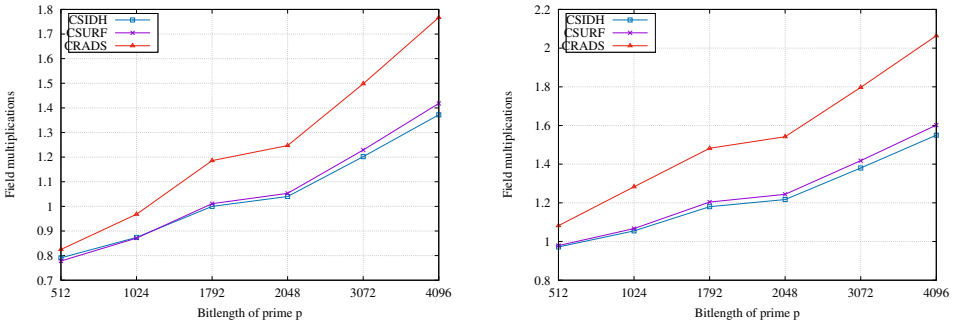
4.1 Benchmarking performance

We first compare the performance of these different implementations using the "traditional" bounds proposed initially in [4] and [5]. After that, we compare performance using more suitable bounds for CSURF and CRADS using an adapted algorithm from [8].

Traditional bounds. We first focus on CSURF and CRADS instantiations using the exponent bounds given in [4] and [5]. We use the same prime as [4]. Then, Table 2 compares constant-time CSIDH-512, CSURF-512 and CRADS-512 under state-of-the-art configurations, and illustrates CSIDH-512 beats both of CSURF-512 and CRADS-512. However, these bounds have been computed using non-constant-time cost assumptions, which gives an unfair advantage to the CSIDH strategy.

Suitable bounds. Next we use suitable exponent bounds that minimize the cost of CSURF and CRADS by using a slight modification of the greedy algorithm presented in [8], which is included in the provided repository. In summary, the greedy algorithm starts by increasing the exponent bound $m_2 \leq 256$ of two (required for 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 optimal bound for CSURF, we proceed in a similar way for CRADS: this time the optimal exponent bound m_2 of CSURF is fixed and the algorithm increases the exponent bounds $m_3, m_5, m_7 \in [\![1 \,..\, m_2]\!]$ of radical isogenies until it is optimal.

Comparisons. The full results are given in Table 2. From Figure 1a we can see that CSURF-512 and CSURF-1024 now have a smaller running time than CSIDH-512 and CSIDH-1024 have. To be more precise, using OAYT-style, CSURF-{512,1024} provides a speed-up over CSIDH-{512,1024}, respectively by 1.64% and 0.34%. Nevertheless, larger instantiations of CSURF becomes less competitive to CSIDH as the 2-isogenies scale worse than Vélu's (square-root) formulæ. As hinted in section 3, CRADS instantiations do not provide a speed-up (the radical computations scale with respect to $\log(p)$). On the other hand, CSIDH gives a better performance against CSURF and CRADS when using MCR-style. Table 2 presents the results obtained in this benchmark and highlights the best result per parameter set.



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

Figure 1: Running time regarding the group action evaluations of CSIDH, CSURF, and CRADS. The numbers are given in millions of multiplications, and they correspond with the average of 1024 random instances.

Dummy-style	512-bits	1024-bits	1792-bits	2048-bits	3072-bits	4096-bits
CSURF (traditional bounds)	0.895	-	-	-	-	-
CRADS (traditional bounds	1.013	-	-	-	-	-
CSIDH-MCR	0.972	1.055	1.180	1.217	1.380	1.550
CSURF-MCR	0.979	1.067	1.204	1.244	1.418	1.602
CRADS-MCR	1.082	1.283	1.482	1.542	1.797	2.064
CSIDH-OAYT	0.791	0.874	**1.000**	**1.040**	**1.202**	**1.372**
CSURF-OAYT	**0.778**	**0.871**	1.011	1.053	1.229	1.418
CRADS-OAYT	0.825	0.968	1.186	1.247	1.498	1.767

Table 2: Results for different prime sizes. The numbers are given in millions of multiplications, and they correspond with the average of 1024 random instances. It considers only multiplication (**M**) and squaring (**S**) operations, and assumes $\mathbf{S} = \mathbf{M}$. Numbers in bold are optimal results for that prime size.

5 Concluding remarks and future research

Our proposed constant-time chain of projective radical isogenies integrated to CSURF-512 and CSURF-1024 provides a speed-up over CSIDH-512 and CSIDH-1024 of about 1.64% and 0.34%, respectively. Nevertheless, larger dummy-based instantiations of CSURF become less competitive to CSIDH (the degree-2 isogenies scale worse than Vélu square-root formulæ), and thus the use of constant-time radical isogenies has a significant negative impact on performance.

We mentioned that a dummy-free chain of radical isogenies requires sampling many points. For that reason, we list a number of open questions that could improve the performance of CSURF and radical isogenies in constant-time:

1. It is currently not known if points can be pushed through radical isogenies, which would allow a random-free variant of radical isogenies.

2. A fully projective version of CSURF and radical isogenies might save some overhead required to switch between different forms of curves.

3. In the end, radical isogenies are given for curves in Tate normal form. Thus, having a fully Tate normal curve arithmetic could save the overhead from moving between the required curve models.

Acknowledgements. The authors would like to thank Simona Samardjiska for her valuable comments that helped to improve the technical material of this paper. This work has been supported by the European Commission through the ERC Starting Grant 804476 (SCARE).

References

[1] R. Azarderakhsh, M. Campagna, C. Costello, L. D. Feo, B. Hess, A. Jalali, B. Koziel D. Jao, B. LaMacchia, P. Longa, M. Naehrig, G. Pereira, J. Renes, V. Soukharev, and D. Urbani. Supersingular isogeny key encapsulation. *Third round candidate of the NIST's post-quantum cryptography standardization process, 2020.*, 2020.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

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