A Direct Key Recovery Attack on SIDH

Luciano Maino[1](✉), Chloe Martindale[1], Lorenz Panny[2], Giacomo Pope[1,3],
and Benjamin Wesolowski[4,5,6]

[1] University of Bristol, Bristol, UK
{luciano.maino,chloe.martindale}@bristol.ac.uk, giacomo.pope@nccgroup.com
[2] Academia Sinica, Taipei, Taiwan
lorenz@yx7.cc
[3] NCC Group, Cheltenham, UK
[4] Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, 33400 Talence, France
benjamin.wesolowski@math.u-bordeaux.fr
[5] INRIA, IMB, UMR 5251, 33400 Talence, France
[6] ENS de Lyon, CNRS, UMPA, UMR 5669, Lyon, France

Abstract. We present an attack on SIDH utilising isogenies between polarized products of two supersingular elliptic curves. In the case of arbitrary starting curve, our attack (discovered independently from [8]) has subexponential complexity, thus significantly reducing the security of SIDH and SIKE. When the endomorphism ring of the starting curve is known, our attack (here derived from [8]) has polynomial-time complexity assuming the generalised Riemann hypothesis. Our attack applies to any isogeny-based cryptosystem that publishes the images of points under the secret isogeny, for example Séta [13] and B-SIDH [11]. It does not apply to CSIDH [9], CSI-FiSh [3], or SQISign [14].

Keywords: SIDH · Elliptic curve · Isogeny · Cryptanalysis

1 Introduction

Supersingular Isogeny Diffie-Hellman (SIDH) [19] is a key exchange proposed in 2011 by Jao and De Feo. It has since become an archetype of *isogeny-based*

Author list in alphabetical order; see https://ams.org/profession/leaders/CultureStatement04.pdf. This paper is a merge of [24] by Maino and Martindale, which gives an attack on SIDH, and [39] by Wesolowski, which constitutes the proof of the main result in this paper. The implementation and algorithmic details of the implementation were contributed by Panny and Pope. This research was funded in part by the UK Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training (CDT) in Trust, Identity, Privacy and Security in Large-scale Infrastructures (TIPS-at-Scale) at the Universities of Bristol and Bath, the Academia Sinica Investigator Award AS-IA-109-M01, the Agence Nationale de la Recherche under grant ANR MELODIA (ANR-20-CE40-0013), and the France 2030 program under grant agreement No. ANR-22-PETQ-0008 PQ-TLS.

C. Hazay and M. Stam (Eds.): EUROCRYPT 2023, LNCS 14008, pp. 448–471, 2023.
https://doi.org/10.1007/978-3-031-30589-4_16

cryptography, a branch of cryptography whose security relates to the presumed hardness of computing isogenies: given two (supersingular) elliptic curves over a finite field, find an isogeny between them. Many other such cryptosystems have been developed [3,9,11,13,14], fuelled by the presumed *quantum* hardness of the isogeny problem, thereby providing security against quantum adversaries. The influence of SIDH is notably illustrated by its incarnation "Supersingular Isogeny Key Encapsulation" (SIKE) [18], a primitive submitted to the NIST standardisation effort to find a new quantum-safe cryptographic standard [27].

Yet, the security of SIDH (hence, SIKE) is not guaranteed by the hardness of the 'pure' isogeny problem. It in fact relies on a variant, where the image of some torsion points under a hidden isogeny are also revealed. This is the *supersingular isogeny with torsion* (SSI-T) problem.

> **Supersingular Isogeny with Torsion (SSI-T):**
> Given coprime integers A and B, two supersingular elliptic curves $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ connected by an unknown degree-A isogeny $\varphi_A : E_0 \to E_A$, and given the restriction of φ_A to the B-torsion of E_0, recover an isogeny φ matching these constraints.

This variant has been shown to be weaker than the pure isogeny problem in a line of work pioneered by Petit [30] in 2017 and expanded in multiple papers in the last 5 years [5,16,31]. However, the SIKE parameters had not been affected by these attacks, which all applied only to variants of SIDH.

In this paper, we present an algorithm that solves SSI-T for parameters that were believed to be secure, including SIKE as well as a few other similar protocols such as B-SIDH [11] and Séta [13]. The first such polynomial-time algorithm was described (and demonstrated against SIKE) by Castryck and Decru [8]: they show that when the endomorphism ring $\mathrm{End}(E_0)$ is known (as is the case in SIKE, B-SIDH or Séta), then SSI-T can be solved in polynomial time, under plausible heuristic assumptions. The idea of the algorithm of [8] is the following. First, they guess a small part of the isogeny φ_A. Based on this guess, they construct some isogeny $\Phi : E_A \times E \to X$, where E is a carefully crafted elliptic curve, and X is some abelian surface. They prove that the guess is correct if X is itself a product of elliptic curves, which can be efficiently detected. This guessing approach allows one to reconstruct φ_A one ternary-bit at a time, at a cost dominated by the many 2-dimensional isogenies Φ that must be computed.

The present work is semi-independent: it is the merge of a mostly independently discovered[1] attack against SIDH [24], with another work [39] subsequent to [8]. In addition to the independent discovery to [8] of such an attack, our main contributions reside in:

- **Practicality:** We develop methods fast enough to possibly find constructive applications. Similarly to [8], we solve SSI-T via isogenies between elliptic products like $E_A \times E$, but we avoid using the iterative 'decision strategy'.

[1] Maino had been working together with Castryck and Decru on a tangentially related project using similar underlying ideas.

Instead, we recover the isogeny φ_A directly from a component of the matrix form of a (B, B)-isogeny, for some integer $B > 0$. As a result, in favourable settings, only one 2-dimensional isogeny computation is required,[2] instead of one per ternary-bit of the secret.

Provability: When $\mathrm{End}(E_0)$ is known, we prove that our method runs in provable polynomial time, assuming the generalised Riemann hypothesis (GRH). When $\mathrm{End}(E_0)$ is unknown, we prove that there is a subexponential attack.

The attack is further supported by a SageMath [36] proof-of-concept implementation available at:

https://github.com/Breaking-SIDH/direct-attack

In the case where $\mathrm{End}(E_0)$ is unknown, Robert [32] proved, following the first version of this work, that there also is a polynomial-time attack. This is asymptotically the fastest known attack in this setting. However, it involves the computation of a special 8-dimensional endomorphism of $E_0^4 \times E_A^4$ (or 4-dimensional, under plausible heuristics), which may limit its practicality.

Finally, note that as in [8] and [32], our attack makes full use of the public torsion points, and as such, it has no effect on isogeny-based cryptosystems that do not publish images of points under a secret isogeny, such as CSIDH [9], CSI-FiSh [3], and SQISign [14].

Outline

The success of our attack on the SSI-T problem relies on Theorem 1, which is proved in Sect. 2. This section additionally includes background material on polarized abelian surfaces. Section 3 describes a subexponential algorithm to solve the SSI-T problem without using knowledge of the endomorphism ring of the starting curve. In Sect. 4, we then show how knowledge of the endomorphism ring improves the performance of the attack, resulting in a provable polynomial time algorithm to solve the SSI-T problem. The paper concludes with Sect. 5 which we use to discuss future work.

Acknowledgements

We would like to thank Christophe Petit for useful comments regarding methods to compute isogenies with irrational kernel points and Eda Kirimli, for useful discussions. We are also extremely grateful to Luca De Feo, who shared with us a better method to find attack parameters during ANTS-XV, which in particular led to the argument in this paper that our algorithm has subexponential complexity. We would also like to thank COSIC and KU Leuven, especially Wouter Castryck and Thomas Decru, for hosting Luciano Maino as an intern, sparking his collaboration that led to this paper.

[2] Together with the computation of the image of one point under said isogeny.

2 The Core of the Attack

Let all notation be as in the SSI-T problem statement above. The core of the attack is the following. First suppose that $B > A$, and that we have access to some isogeny $\varphi_f : E \to E_0$ of degree $f = B - A$, given in any form that allows to evaluate it on the B-torsion. We postpone the discussion on finding such a φ_f, as the method may depend on the context. Assuming φ_f is provided, we give an algorithm (Algorithm 1) that recovers a generator of $\ker(\varphi_A)$ (i.e., solves SSI-T), at a cost dominated by one evaluation of a (B, B)-isogeny with known kernel (with an A-torsion point as input), and two evaluations of $\widehat{\varphi_f}$ (with two B-torsion points as input). In this section, we focus on the design and analysis of Algorithm 1 for this core task.

The idea is the following. Write $g_A : E \to F$ for the isogeny of kernel $\hat{\varphi}_f(\ker(\varphi_A))$, and $g_f : F \to E_A$ for the isogeny of kernel $g_A(\ker(\varphi_f))$, so that the following diagram commutes:

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi_A"] & E_A \\
E \arrow[u, "\varphi_f"] \arrow[r, "g_A"'] & F. \arrow[u, "g_f"']
\end{tikzcd} \qquad (1)$$

Now, consider the 2-dimensional isogeny

$$\begin{aligned} \varPhi : E \times E_A &\longrightarrow E_0 \times F \\ (P, Q) &\longmapsto (\varphi_f(P) - \widehat{\varphi_A}(Q), g_A(P) + \hat{g_f}(Q)). \end{aligned}$$

Observe that $-\widehat{\varphi_A}$ is equal to the composition

$$E_A \xrightarrow{0 \times \mathrm{id}_{E_A}} E \times E_A \xrightarrow{\varPhi} E_0 \times F \xrightarrow{\mathrm{pr}_1} E_0,$$

where the first map is the inclusion map with image $\{0\} \times E_A$, the middle map is $\varPhi$, and the last is the natural projection map. Assuming that each map in this composition is efficiently computable, then we can evaluate $\widehat{\varphi_A}$ on any input. That directly leads to a recovery of $\ker(\varphi_A)$, hence to a solution of SSI-T. The difficulty is in proving that each step is indeed efficiently computable. The computation of the first inclusion is trivial. The step $\varPhi$ requires a delicate analysis of this 2-dimensional isogeny, to prove that its kernel can be computed, and that this kernel permits an efficient evaluation of $\varPhi$. The last step—the projection—may seem clear, but in fact hides a subtlety. The decomposition $E_0 \times F$ is only available if $\varPhi$ is of a certain kind: it must behave well with respect to the implicit *product polarizations* of the domain and codomain.

2.1 Isogenies Between Abelian Surfaces

Abelian surfaces can come equipped with a *polarization*. A polarization of X is an isogeny $\lambda_X : X \to X^\vee$ to the dual variety $X^\vee$. For a primer on the theory of polarizations, we refer the reader to [26]; for the purpose at hand, we recall in this section the relevant useful facts as a black-box.

Computationally, a polarization is essentially the data of an equation of the abelian surface. A relevant example: given two elliptic curves E_1 and E_2, the surface $E_1 \times E_2$ comes naturally equipped with a product polarization λ_{E_1,E_2}, which is computationally represented by the data of the equations of E_1 and E_2.

The importance of this notion becomes clear in the context of supersingular curves. If $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$ are supersingular, the abelian surface $E_1 \times E_2$ is called *superspecial*. There is a unique isomorphism class of superspecial abelian surfaces over $\mathbb{F}_{p^2}$. In particular, if E_3 and E_4 are any other supersingular curves defined over $\mathbb{F}_{p^2}$, then $E_1 \times E_2$ and $E_3 \times E_4$ are isomorphic *as abelian surfaces*. However, they can be distinguished by their implicit product polarizations: the polarized surfaces $(E_1 \times E_2, \lambda_{E_1,E_2})$ and $(E_3 \times E_4, \lambda_{E_3,E_4})$ are isomorphic if and only if $E_1 \cong E_i$ and $E_2 \cong E_j$ for $\{i, j\} = \{3, 4\}$.

Given a positive integer B, a B-isogeny $\Phi : (X, \lambda_X) \to (Y, \lambda_Y)$ is an isogeny such that $[B] \circ \lambda_X = \Phi^\vee \circ \lambda_Y \circ \Phi$, where $\Phi^\vee : Y^\vee \to X^\vee$ is the dual isogeny. A (B, B)-isogeny is a B-isogeny between abelian surfaces whose kernel is isomorphic to $(\mathbb{Z}/B\mathbb{Z})^2$. As we shall mention in Sect. 3.1, there are algorithms which, given a source (X, λ_X), and the kernel of a (B, B)-isogeny $\Phi : (X, \lambda_X) \to (Y, \lambda_Y)$, compute the target (Y, λ_Y) and can evaluate Φ at points, in time polynomial in $\log(B)$ and in the largest prime factor of B. In particular, if

$$\Phi : E_1 \times E_2 \longrightarrow E_3 \times E_4$$

is a (B, B)-isogeny with respect to the product polarizations, the algorithm is given as input equations of E_1 and E_2, and generators of $\ker(\Phi)$, and recovers equations for E_3 and E_4. It can also take as input two points $P_1 \in E_1$ and $P_2 \in E_2$, and output P_3 and P_4 such that $\Phi(P_1, P_2) = (P_3, P_4)$.

Finally, as products of elliptic curves will be of particular interest, let us introduce some convenient notation. Given four elliptic curves E_1, E_2, E_1', and E_2', and four isogenies $\varphi_{ij} : E_i \to E_j'$ for $i, j \in \{1, 2\}$, the matrix

$$M = \begin{pmatrix} \varphi_{11} & \varphi_{12} \\ \varphi_{21} & \varphi_{22} \end{pmatrix},$$

represents the isogeny

$$\begin{aligned} \Phi : E_1 \times E_2 &\longrightarrow E_1' \times E_2' \\ (P_1, P_2) &\longmapsto (\varphi_{11}(P_1) + \varphi_{12}(P_2), \varphi_{21}(P_1) + \varphi_{22}(P_2)). \end{aligned}$$

We call M a matrix form of Φ.

2.2 The Algorithm

Our attack is a consequence of the following theorem, which is based on a criterion due to Kani [20]. This criterion determines whether a polarized isogeny originating from an elliptic product admits an elliptic product as codomain.

Theorem 1. *Let f, A, and B be pairwise coprime integers such that $B = f + A$, and let E, E_A, E_0, and F be elliptic curves connected by the following commutative diagram of isogenies:*

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi_A"] & E_A \\
E \arrow[u, "\varphi_f"] \arrow[ur, "\varphi"] \arrow[r, "g_A"'] & F \arrow[u, "g_f"']
\end{tikzcd} \tag{2}$$

where $\deg(\varphi_f) = \deg(g_f) = f$ *and* $\deg(\varphi_A) = \deg(g_A) = A$.
The isogeny

$$\Phi = \begin{pmatrix} \varphi_f & -\widehat{\varphi_A} \\ g_A & \widehat{g_f} \end{pmatrix} \in \mathrm{Hom}(E \times E_A, E_0 \times F),$$

is a (B, B)-isogeny with respect to the natural product polarizations on $E \times E_A$ and $E_0 \times F$, and has kernel $\ker(\Phi) = \{([A]P, \varphi(P)) \mid P \in E[B]\}$.

This theorem allows us to compute the isogeny Φ efficiently (as long as B is smooth—preferably a power of 2 for good practical performance). Furthermore, it implies that this computation leads to the product polarization on the codomain. It leads to the following result.

Corollary 1. *Algorithm 1 is correct and costs two evaluations of $\widehat{\varphi_f}$ on B-torsion input points, at most two evaluations of a (B, B)-isogeny (given by its kernel) on A-torsion input points, and one inversion modulo B.*

Algorithm 1: Solving SSI-T, provided an isogeny of degree $B - A$.

```
Input: Coprime integers $A$ and $B$, two supersingular elliptic curves $E_0/\mathbb{F}_{p^2}$ and
       $E_A/\mathbb{F}_{p^2}$ connected by an unknown degree-$A$ isogeny $\varphi_A : E_0 \to E_A$ of
       cyclic kernel, a basis $\{P_B, Q_B\}$ of $E_0[B]$, a basis $\{P_A, Q_A\}$ of $E_A[A]$, the
       image points $P'_B = \varphi_A(P_B)$, $Q'_B = \varphi_A(Q_B)$, an isogeny $\varphi_f : E \to E_0$ of
       degree $f = B - A$.
Output: A generator of $\ker(\varphi_A)$.
Let $c \in \mathbb{Z}$ such that $cf \equiv 1 \bmod B$.
Let $P''_B = [c] \circ \widehat{\varphi_f}(P_B)$ and $Q''_B = [c] \circ \widehat{\varphi_f}(Q_B)$. We have $\varphi_A \circ \varphi_f(P''_B) = P'_B$, and
  $\varphi_A \circ \varphi_f(Q''_B) = Q'_B$.
Let $\Phi : E \times E_A \to E_0 \times F$ be the $(B, B)$-isogeny with kernel
  $\langle([A]P''_B, P'_B), ([A]Q''_B, Q'_B)\rangle$.
Compute $\Phi(0, P_A) = (P'_A, x)$. We have $P'_A = \widehat{\varphi_A}(P_A)$.
Return $P'_A$ if it has order $A$.
Else, compute $\Phi(0, Q_A) = (Q'_A, y)$ (satisfying $Q'_A = \widehat{\varphi_A}(Q_A)$), and return $Q'_A$.
```

Remark 1. Note that while the algorithm necessitates *at most* two evaluations of the (B, B)-isogeny, a single one is often sufficient. Indeed, suppose the basis $\{P_A, Q_A\}$ is uniformly random. If, for instance, $A = 2^a$, then $[2^{a-1}]P_A \notin \ker \widehat{\varphi_A}$ (i.e., P'_A has order A) with probability $2/3$. Even if P'_A does not have order precisely A, it is likely to be close to A, in which case P'_A generates most of $\ker(\varphi_A)$, and a simple exhaustive search can recover the missing information.

2.3 Proof of Theorem 1

In this section, we prove Theorem 1.

Prelude on the Adjoint. Consider an isogeny $\Phi : E_1 \times E_2 \to E'_1 \times E'_2$ represented by the matrix $M = \begin{pmatrix} \varphi_{11} & \varphi_{12} \\ \varphi_{21} & \varphi_{22} \end{pmatrix}$, where $\varphi_{ij} : E_i \to E'_j$. The adjoint of Φ is the isogeny $\tilde{\Phi} : E'_1 \times E'_2 \to E_1 \times E_2$ represented by the matrix

$$\tilde{M} = \begin{pmatrix} \hat{\varphi}_{11} & \hat{\varphi}_{21} \\ \hat{\varphi}_{12} & \hat{\varphi}_{22} \end{pmatrix}.$$

Our interest in this notion is that it offers a practical characterisation of isogenies that preserve the product polarizations: the isogeny Φ is a B-isogeny with respect to the product polarizations if and only if $\tilde{M}M = B\mathrm{Id}_2$, where Id_2 is the identity. While this property seems standard, let us provide a proof that only relies on well-documented properties of pairings. First, we show that the adjoint is closely related to the dual.

Lemma 1. *We have* $\tilde{\Phi} = \lambda_{E_1,E_2}^{-1} \circ \Phi^\vee \circ \lambda_{E'_1,E'_2}$, *where*

$$\Phi^\vee : (E'_1 \times E'_2)^\vee \to (E_1 \times E_2)^\vee,$$

is the dual.

Proof. The dual $\Phi^\vee$ is the unique isogeny that satisfies

$$e_{E'_1 \times E'_2, n}(\Phi(P), Q) = e_{E_1 \times E_2, n}(P, \Phi^\vee(Q)),$$

for any positive integer n, any $P \in (E_1 \times E_2)[n]$, and any $Q \in (E'_1 \times E'_2)^\vee[n]$, where $e_{-\times-,n}$ denotes the (unpolarized) Weil pairings. Let us show that $\Psi = \lambda_{E_1,E_2} \circ \tilde{\Phi} \circ \lambda_{E'_1,E'_2}^{-1}$ satisfies this property (hence $\Psi = \Phi^\vee$, proving the lemma). Recall that the polarized Weil pairing $e_n^{\lambda_{E_1,E_2}}$ (for the product polarization $\lambda_{E_1,E_2} : E_1 \times E_2 \to (E_1 \times E_2)^\vee$) satisfies

$$e_n^{\lambda_{E_1,E_2}}(P, Q) = e_{E_1 \times E_2, n}(P, \lambda_{E_1,E_2}(Q)) = e_{E_1,n}(P_1, Q_1) e_{E_2,n}(P_2, Q_2),$$

where $P = (P_1, P_2)$ and $Q = (Q_1, Q_2)$ are in $(E_1 \times E_2)[n]$, and $e_{E_i,n}$ are the Weil pairings on elliptic curves. We deduce that

$$\begin{aligned} e_n^{\lambda_{E_1',E_2'}}(\Phi(P), Q) &= \prod_j \prod_i e_{E_j',n}(\varphi_{ij}(P_i), Q_j) \\ &= \prod_j \prod_i e_{E_i,n}(P_i, \hat{\varphi}_{ij}(Q_j)) \\ &= e_n^{\lambda_{E_1,E_2}}(P, \tilde{\Phi}(Q)). \end{aligned}$$

It follows that

$$\begin{aligned} e_{E_1' \times E_2',n}(\Phi(P), Q) &= e_n^{\lambda_{E_1',E_2'}}(\Phi(P), \lambda_{E_1',E_2'}^{-1}(Q)) \\ &= e_n^{\lambda_{E_1,E_2}}(P, \tilde{\Phi} \circ \lambda_{E_1',E_2'}^{-1}(Q)) \\ &= e_{E_1 \times E_2,n}(P, \lambda_{E_1,E_2} \circ \tilde{\Phi} \circ \lambda_{E_1',E_2'}^{-1}(Q)), \end{aligned}$$

hence $\Psi = \Phi^\vee$ as desired. □

Lemma 2. *Let B be a positive integer. An isogeny $\Phi : E_1 \times E_2 \to E_1' \times E_2'$ is a B-isogeny with respect to the product polarizations if and only if $\tilde{\Phi} \circ \Phi = [B]$.*

Proof. Recall that Φ is a B-isogeny with respect to the product polarizations if and only if $[B] \circ \lambda_{E_1,E_2} = \Phi^\vee \circ \lambda_{E_1',E_2'} \circ \Phi$. The result thus immediately follows from Lemma 1. □

For the rest of this section, assume the notation from Theorem 1.

Lemma 3. *We have that Φ is a B-isogeny with respect to the product polarizations.*

Proof. The isogeny Φ has matrix form $\begin{pmatrix} \varphi_f & -\widehat{\varphi_A} \\ g_A & \widehat{g_f} \end{pmatrix}$, so its adjoint has matrix form $\begin{pmatrix} \widehat{\varphi_f} & \widehat{g_A} \\ -\varphi_A & g_f \end{pmatrix}$. We have

$$\begin{aligned} \begin{pmatrix} \widehat{\varphi_f} & \widehat{g_A} \\ -\varphi_A & g_f \end{pmatrix} \begin{pmatrix} \varphi_f & -\widehat{\varphi_A} \\ g_A & \widehat{g_f} \end{pmatrix} &= \begin{pmatrix} [\deg(\varphi_f) + \deg(g_A)] & 0 \\ 0 & [\deg(\varphi_A) + \deg(g_f)] \end{pmatrix} \\ &= \begin{pmatrix} [B] & 0 \\ 0 & [B] \end{pmatrix}. \end{aligned}$$

The result follows from Lemma 2. □

Lemma 4. *We have* $\ker(\Phi) = \{([A]P, \varphi(P)) \mid P \in E[B]\}$.

Proof. Let $K = \{([A]P, \varphi(P)) \mid P \in E[B]\}$, and let us show that $\ker(\Phi) = K$. The inclusion $K \subseteq \ker(\Phi)$ follows from

$$\begin{aligned}\Phi([A]P, \varphi(P)) &= (\varphi_f([A]P) - \widehat{\varphi_A} \circ \varphi(P), g_A([A]P) + \widehat{g_f} \circ \varphi(P)) \\ &= ([A] \circ \varphi_f(P) - \widehat{\varphi_A} \circ \varphi_A \circ \varphi_f(P), [A] \circ g_A(P) + \widehat{g_f} \circ g_f \circ g_A(P)) \\ &= ([A - A] \circ \varphi_f(P), [A + f] \circ g_A(P)) \\ &= (0, [B] \circ g_A(P)) = (0, 0).\end{aligned}$$

To show that $\ker(\Phi) \subseteq K$, let $([A]P, Q) \in \ker(\Phi)$. Then, $\varphi_f([A]P) = \widehat{\varphi_A}(Q)$, hence

$$[A] \circ \varphi(P) = \varphi_A \circ \varphi_f([A]P) = \varphi_A \circ \widehat{\varphi_A}(Q) = [A]Q.$$

Since $P \in E[B]$, and A and B are coprime, we deduce $Q = \varphi(P)$, hence $([A]P, Q) \in K$. □

Theorem 1 now follows from Lemma 3 and Lemma 4: the isogeny Φ is a B-isogeny with respect to the product polarizations, with kernel $\ker(\Phi) = \{([A]P, \varphi(P)) \mid P \in E[B]\}$ isomorphic to $(\mathbb{Z}/B\mathbb{Z})^2$, hence it is a (B, B)-isogeny.

3 The Case of Unknown Endomorphism Ring

To use Theorem 1 to solve the SSI-T problem, any f-isogeny $\varphi_f : E \to E_0$ suffices. When $\mathrm{End}(E_0)$ is unknown, for example in the case of a trusted setup, the problem faced by the attacker is that the computation of φ_f is not necessarily easy as there is no reason to expect $B - A$ to be smooth. To mitigate this, we increase our pool of available cofactors f by brute-forcing the last few steps of φ_A and/or by brute-forcing some extra torsion-point images; this amounts to multiplying A and B respectively by small (fractions of) integers. For ease of notation, in all that follows we will assume that $A = \ell_A^a$ and $B = \ell_B^b$, where ℓ_A and ℓ_B are coprime integers.

The picture that we should keep in mind when reading through the attack below is the following commutative diagram, where:

- $\varphi_A : E_0 \to E_A$ is the secret isogeny,
- $\varphi_f : E \to E_0$ is a f-isogeny chosen by the attacker,[3]
- $\varphi_{\ell_A^i} : E' \to E_A$ is a guess of the (dual of the) last i steps of φ_A,
- $\varphi' : E_0 \to E'$ is the corresponding first $a - i$ steps of φ_A such that $\varphi_A = \varphi_{\ell_A^i} \circ \varphi'$, and
- $\varphi : E \to E'$ is the $f\ell_A^{a-i}$-isogeny to which we apply Theorem 1.

[3] In practice, the attacker computes $\widehat{\varphi}_f$ and deduces φ_f from this.

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi'"'] \arrow[rr, bend left, "\varphi_A"] & E' \arrow[r, "\varphi_{\ell_A^i}"'] & E_A \\
E \arrow[u, "\varphi_f"] \arrow[ur, "\varphi"'] & &
\end{tikzcd} \tag{3}$$

The attack is described in Algorithm 2, which is a natural generalisation of Algorithm 1. The parameters e, i, j are introduced to make $f = eB\ell_B^{-j} - A\ell_A^{-i} > 0$ smooth enough and apply Theorem 1 on the parameters $A \rightsquigarrow A\ell_A^{-i}$, $B \rightsquigarrow eB\ell_B^{-j}$, and $f \rightsquigarrow eB\ell_B^{-j} - A\ell_A^{-i}$. Once a f-isogeny $\varphi_f \colon E \to E_0$ is computed, the attacker reconstructs an $eB\ell_B^{-j}$-basis on E matching the B-basis on E_0 defined in the setup stage in SIDH. Then, the attacker guesses the last i steps of the secret isogeny φ_A computing an isogeny $\varphi_{\ell_A^i} : E' \to E_A$ of degree ℓ^i. For each guess, it is necessary to check all the $eB\ell_B^{-j}$-torsion points matching the B-torsion points on E_A defined by the public key. For each pair of the $eB\ell_B^{-j}$-torsion points found, the attacker tries to compute a $(eB\ell_B^{-j}, eB\ell_B^{-j})$-isogeny Φ as in Theorem 1. If the codomain of Φ consists of an elliptic product, the first $a - i$ steps of the secret isogeny are revealed in one of the components of the matrix form of Φ. This high-level overview is made clear in Algorithm 2.

Remark 2. Step 5 in Algorithm 2 has a small chance of causing the overall algorithm to fail, as a split Jacobian may accidentally be the codomain for an incorrect guess. However it is easy to check whether or not E_0 is a factor, and furthermore the chance of failure is very small.

To discuss the complexity of this attack we should split it into three parts:

1. The precomputation step (Step 1); this can be done once and for all for each parameter set A, B.
2. The cofactor isogeny computation (Step 2); if SIDH is set up with a fixed (arbitrary) E_0, this can be done once and for all for this E_0.
3. The online steps (Steps 3 to 7); these steps need to be performed for every new public key.

The Cost of the Cofactor Isogeny Computation. The cofactor isogeny remains fixed and is chosen by the attacker. As such, it does not need to be recomputed at any point due to a wrong guess when brute-forcing. We compute the isogeny φ_f via a chain of isogenies φ_{q_f} of prime degree q_f. It is worth noting that if a square factor appears in the factorization of f, we can simply perform a scalar multiplication $[q_f]$ rather than computing two q_f-isogenies. The cost of computing φ_{q_f} for the larger factors q_f is discussed in detail in Sect. 3.2; an estimate (in terms of $\mathbb{F}_p$-multiplications) can be given as $\tilde{O}({q_f}^2)$.

Algorithm 2: Solving SSI-T, general approach.

```
Input: Coprime integers $A = \ell_A^a$ and $B = \ell_B^b$, two supersingular elliptic curves
       $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ connected by an unknown degree-$A$ isogeny
       $\varphi_A : E_0 \to E_A$, a basis $\{P_B, Q_B\}$ of $E_0[B]$, a basis $\{P_A, Q_A\}$ of $E_0[A]$, the
       image points $\varphi_A(P_B)$, $\varphi_A(Q_B)$.
Output: A generator of $\ker(\varphi_A)$.
1 Compute integers $e$, $j$, $f$, and $i$ such that the overall cost according to the
    estimates in Sect. 3.1 is minimised, and $eB\ell_B^{-j} = f + A\ell_A^{-i}$. For ease of notation,
    we set $A' = A\ell_A^{-i}$ and $B' = B\ell_B^{-j}$.
2 Compute a curve that is $f$-isogenous to $E_0$, define the dual of the computed
    isogeny to be $\varphi_f : E \to E_0$, and compute $\widehat{\varphi}_f(P_B), \widehat{\varphi}_f(Q_B)$. For more details, see
    Sect. 3.2.
3 Compute a basis $\{P_{eB'}, Q_{eB'}\}$ of $E[eB']$ such that $[e]P_{eB'} = [\ell_B^j]\widehat{\varphi}_f(P_B)$ and
    $[e]Q_{eB'} = [\ell_B^j]\widehat{\varphi}_f(Q_B)$.
4 Choose a guess $\varphi_{\ell_A^i} : E' \to E_A$ for the last $i$ steps of $\varphi_A$, recall the definition of the
    corresponding $\varphi : E \to E'$ from diagram (3), and choose $R, S \in E'[eB']$ such that
        $$[e]R = [\ell_A^{-i} f \ell_B^j]\widehat{\varphi}_{\ell_A^i} \circ \varphi_A(P_B)$$
    and
        $$[e]S = [\ell_A^{-i} f \ell_B^j]\widehat{\varphi}_{\ell_A^i} \circ \varphi_A(Q_B),$$
    i.e. $R, S$ are a guess for the images $\varphi(P_{eB'}), \varphi(Q_{eB'})$ respectively.
5 Compute a $(eB', eB')$-isogeny with domain $E \times E'$ and kernel
        $$\ker(\Phi_{\text{guess}}) = \langle ([A']P_{eB'}, R), ([A']Q_{eB'}, S) \rangle.$$
    If the codomain splits, continue (see Remark 2). Else, return to Step 4 and take a
    new guess $(\varphi_{\ell_A^i}, R, S)$. For more details see Sect. 3.3.
6 Choose a basis $\{P, Q\}$ of $E'[A']$; compute $\widehat{\varphi'}(P)$ and $\widehat{\varphi'}(Q)$ via
        $$\Phi(0_E, P) = (-\widehat{\varphi'}(P), \widehat{g}_f(P))) \text{ and } \Phi(0_E, Q) = (-\widehat{\varphi'}(Q), \widehat{g}_f(Q))).$$
7 Compute $\ker(\varphi') = \langle \widehat{\varphi'}(P), \widehat{\varphi'}(Q) \rangle$ and return a generator of $\ker(\varphi_{\ell_A^i} \circ \varphi')$.
```

The Cost of the Online Steps. The discussion in Sect. 3.1 approximates the cost of Steps 3 to 7 by $\approx C \cdot e^4 \ell_A^i q_e^4 \log q_e$, where q_e is the largest prime dividing e and C is polynomial in $\log(p)$. We allow i and e to grow to increase the pool of options for f in order to get a smaller q_f, where q_f is the largest prime dividing f.

The Precomputation. If SIDH is set up to start every key exchange with a new E_0, the optimal choice of (e, i, j, f) for the attacker ensures that the cost of Step 2 is approximately the same as the cost of Steps 3 to 5. One could perform a brute force search over all parameters (e, i, j, f) such that $q_f^2 \leq e^4 \ell_A^i q_e^4 \log q_e$ and $0 \leq j \leq b$, which would be costly.

Even though this exhaustive search should be done once and for all, the search space for SIKE parameters is too big to be bruteforced. However, since sharing the first version of this paper [25], Luca De Feo shared with us a heuristic subexponential algorithm for the precomputation leading both to a subexponential cofactor isogeny computation and to subexponential online steps. His argument is as follows: suppose that we wish to target $A \approx B \approx 2^b$. To achieve subexponential complexity $L_{2^b}(c, 1/2)$, one can see from the complexity discussion of the online and cofactor steps above that it is sufficient to find parameters (e, i, j, f) such that $e, \ell_A^i \approx 2^{\sqrt{b}}$, and f is $\sqrt{b}^{\sqrt{b}}$-smooth.

To achieve this, we search for solutions to the equation

$$xA\ell_A^{-i} + yB\ell_B^{-j} = z, \tag{4}$$

where $x, y \leq 2^{\sqrt{b}}$, z is $\sqrt{b}^{\sqrt{b}}$-smooth, and i and j are fixed at some chosen values such that $\ell_A^i \approx \ell_B^j \approx 2^{\sqrt{b}}$. This corresponds to $e = -y$ (not necessarily coprime to B) and $f = -xz$; if xz, $y > 0$ then we switch the roles of A and B and this will correspond to $e = -x$ and $f = -yz$. Writing $f = -xz$ corresponds to decomposing $\varphi_f : E_f \to E_0$ into a degree-$(-z)$ isogeny $\varphi_{-z} : E_f \to E_0'$ and a degree-x isogeny $\varphi_x : E_0' \to E_0$, and recovering $\varphi_A \circ \varphi_x$ by applying Algorithm 2 with $A = xA$, $E_0 = E_0'$, and $\varphi_A = \varphi_A \circ \varphi_x$. Pictorially, this situation can be summarised by the following diagram.

$$\begin{tikzcd}
E_0' \arrow[r, "\varphi_x"] & E_0 \arrow[r, "\varphi'"'] \arrow[rr, bend left, "\varphi_A"] & E' \arrow[r, "\varphi_{\ell_A^i}"'] & E_A \\
& E \arrow[ul, "\varphi_{-z}"] \arrow[ur, "\varphi"'] & &
\end{tikzcd}$$

To find such (x, y, z) for a given (i, j), we run Euclid's xgcd algorithm on $(A\ell_A^{-i}, B\ell_B^{-j})$ until we find (x_0, y_0, z_0) and (x_1, y_1, z_1) such that $x_i, y_i \approx 2^{\sqrt{b}/2}$; this should correspond to $z_i \approx 2^{b-\sqrt{b}/2}$. Then, we search through all linear combinations $uz_0 + vz_1$ with $u, v \leq 2^{\sqrt{b}/2}$ and save the smoothest result; call this z. An integer (such as z) of size 2^b is $\sqrt{b}^{\sqrt{b}}$-smooth with probability $\rho(\beta)$, where $2^{b/\beta} = \sqrt{b}^{\sqrt{b}}$ and ρ is the Dickman-ρ function which can be approximated by $\rho(\beta) \approx \beta^{-\beta}$. Therefore, we are likely to find a $\sqrt{b}^{\sqrt{b}}$-smooth choice z if the number of choices for (u, v), that is $2^{\sqrt{b}}$, is $\approx \beta^\beta$. A short calculation shows that

$$\log_2(\beta^\beta) = \sqrt{b}\left(1 + \frac{2 - 2\log_2 \log_2 b}{\log_2 b}\right) \approx \log_2(2^{\sqrt{b}}).$$

We give some examples for concrete parameters in Sect. 3.1.

3.1 Heuristic Complexity of Algorithm 2

Here, we give some details on and study the complexity of the first four steps of Algorithm 2 in the case relevant to SIKE, namely $A = 3^a$ and $B = 2^b$, with a focus on the Microsoft challenge parameters $A = 3^{67}$ and $B = 2^{110}$ and the parameters $A = 3^{137}$ and $B = 2^{216}$ that were proposed for NIST Level I.

Choosing Parameters. To understand Step 1, we recall the commutative diagram that we keep in mind during this attack, where:

- $\varphi_A : E_0 \to E_A$ is the secret isogeny,
- $\varphi_f : E \to E_0$ is a f-isogeny chosen by the attacker,
- $\varphi_{\ell_A^i} : E' \to E_A$ is a guess of the last i steps of φ_A,
- $\varphi' : E_0 \to E'$ is the corresponding first $a - i$ steps of φ_A such that $\varphi_A = \varphi_{\ell_A^i} \circ \varphi'$, and
- $\varphi : E \to E'$ is the $f\ell_A^{a-i}$-isogeny to which we apply Theorem 1.

$$\begin{tikzcd}
E_0 \arrow[r, "\varphi'"'] \arrow[rr, bend left, "\varphi_A"] & E' \arrow[r, "\varphi_{\ell_A^i}"'] & E_A \\
E \arrow[u, "\varphi_f"] \arrow[ur, "\varphi"'] & &
\end{tikzcd} \tag{5}$$

Choosing f. The shape of f determines the complexity of computing φ_f. The cofactor f does not need to be small as the isogeny can be precomputed once and for all, but it does need to be smooth: considering the extreme case that f is a prime $\approx A$, computing φ_f directly will be harder than computing φ_A directly (because of the extension field arithmetic). Exactly how smooth we require f to be depends on what we hope we can achieve in complexity for the attack. If q_f is the largest prime divisor (of odd multiplicity) of f, the complexity of Step 2 will be dominated by the cost of the computation of a q_f-isogeny, which involves operations in the field of definition of a generator of the kernel of the isogeny. The field of definition is unfortunately hard to control, and large field extensions can have a very serious performance impact. However, note that the required degree depends on arithmetic properties of the pair (p, q_f), rather than just the size of q_f: for some values of q_f the minimal k for which $E(\mathbb{F}_{p^k})$ contains a q_f-torsion point will be much smaller than q_f, but the typical case in our setting is $k \approx q_f$. Based on this preliminary discussion, we will see in more detail in Sect. 3.2 that the cost of computing φ_{q_f}, and therefore φ_f, can be approximated as $\tilde{O}(q_f^2)$.

Choosing i and e. The cost coming from i is the cost of brute-forcing all the cyclic $3^i = \ell_A^i$-isogenies from E_A, which costs $\approx 3^i$ multiplications in $\mathbb{F}_{p^2}$. This is however multiplied by the brute-force cost of guessing the images of the e-torsion points in Step 4 and by the cost of computing $\varPhi$. Guessing the images of the e-torsion points amounts to checking all the pairs of points of order e on E', which is $\approx e^4$. As a result, we have to run Steps 3 to 5 of Algorithm 2 $\approx e^4 3^i$ times.

Additionally, the isogeny $\varPhi$ (which we will attempt to compute $\approx e^4 3^i$ times) is an (eB', eB')-isogeny; in particular it factors via an (e, e)-isogeny. So, in addition we require e to be q_e-smooth, where q_e is the largest prime for which it is feasible to compute (q_e, q_e)-isogenies (potentially over an extension field, which again will add a non-negligible cost). The need for the computation of the (e, e)-isogeny is the main barrier to implementing our algorithm for the proposed NIST parameters, as to do so requires a working implementation of (q_e, q_e)-isogenies, which while should theoretically be possible and reasonably fast, requires some research to achieve. There exists literature on this topic [4,6,22,23], from which we have made a baseline assumption that computations of (q_e, q_e)-isogenies over $\mathbb{F}_{p^k}$ can be performed in $O(q_e^3)$ multiplications in $\mathbb{F}_{p^k}$. However, there is very little existing work in the way of practical implementation of supersingular Jacobians and products of elliptic curves. We do note here that it would be possible to avoid implementing the factors of the (e, e)-isogenies to also map to and from products of elliptic curves, as we can ensure to start and finish the computation of $\varPhi$ with a (2,2)-isogeny, which may make the practical implementation of (e, e)-isogenies with regards to this attack a more achievable goal.

Working with our baseline assumption that a (q_e, q_e)-isogeny can be computed in approximately q_e^3 multiplications over the base field of its kernel, we expect the cost of computing $\varPhi$ as a (eB, eB)-isogeny to be dominated by the cost of computing a (q_e, q_e)-isogeny where q_e is the largest prime factor of e. We leave a careful analysis of the sizes of the field extensions for genus 2 to later work that includes a practical implementation of (q_e, q_e)-isogenies for prime $q_e \neq 2$, but let us assume for the sake of argument that the slow down for the extension field arithmetic scales with q_e similarly to the elliptic curve case. Then, assuming that the field extensions required are large enough that it is best to use the Fast Fourier Transform for multiplication, we approximate the cost of computing the (q_e, q_e)-isogeny by $O(q_e^3 \cdot q_e \log q_e)$. This is probably an overestimate: more research is needed into the existence of $\sqrt{\text{é}\text{lu}}$-style-algorithms in the case of abelian surfaces. However, if the attack costs 2^λ, note that e is already forced to be relatively small compared to this by the fact that we have to search through $\approx e^4$ pairs of possible images of e-torsion points. Because of this, we can expect e to be fairly smooth compared to f, for example, so q_e (and the corresponding field extension) need not be particularly large.

In our choice of parameters for our toy example, we have chosen to demonstrate the use of e without the need to delve into (q_e, q_e)-isogenies for $q_e > 2$ by choosing $e = 2$. In this case we need a field extension of degree 4 for the 2^{b+1}-torsion points. This is not special to this instance but a consequence of the fact that the pull-back of the multiplication-by-2 map contains a square root (and no other rational but not integral powers), and so each lift of a point of order 2^i to a point of order 2^{i+1} will either double the degree of the field extension or keep it the same.

Choosing j. The choice of j only potentially effects the precomputation step, Step 1 of Algorithm 1, as we achieve $B' = 2^{-j}B$-torsion points by multiplying the known B-torsion by 2^j; for this reason we have no restrictions on non-negative j. Notice that we do not require e to be coprime to B, so e may contain powers of two, accounting also for the possibility of negative j.

Concrete Attack Parameters. We present here some choices of attack parameters in three cases of interest: two toy examples to test our algorithm, the Microsoft challenge parameters, and the parameters of SIKEp434 that were proposed for NIST Level I.

Toy Parameters: First, we construct a small example to test our algorithm using the 34-bit prime $p = 2^{19} \cdot 3^9 - 1$, with attack parameters $e = 2$, $i = 1$, $j = 0$ and $f = 5 \cdot 13 \cdot 17 \cdot 23 \cdot 41$. The largest field extension that we need for the computation of φ_f is $\mathbb{F}_{p^{20}}$, for the 41-isogeny. The largest field extension for $e = 2$ is $\mathbb{F}_{p^4}$, for the pullbacks of the order-2^{19} points to order-2^{20} points. This runs in less than 10 s on a single core on a standard laptop; see our code linked below.

We additionally demonstrate our attack on the 64-bit prime $p = 2^{33} \cdot 3^{19} - 1$, which was introduced in [29] as a small example instance for the Castryck–Decru attack, using the attack parameters $(e, i, j, f) = (1, 3, 5, 5 \cdot 11 \cdot 13 \cdot 19 \cdot 47 \cdot 353)$. The largest field extension involved in computing φ_f is $\mathbb{F}_{p^{176}}$, for the 353-isogeny. As $e = 1$, no extension is required to perform point division. This runs in less than 1 min on a single core on a standard laptop.

Our code for attacking both of the above parameter sets is available at:

https://github.com/Breaking-SIDH/direct-attack

Challenge Parameters: We consider one of the sets of challenge parameters put forward by Microsoft [12]: $A = 3^{67}$, $B = 2^{110}$, $i = 7$, $e = 1$, $j = 2$,

$$f = 5 \cdot 7 \cdot 13^3 \cdot 43^2 \cdot 73 \cdot 151 \cdot 241 \cdot 269 \cdot 577 \cdot 613 \cdot 28111 \cdot 321193.$$

The largest field extension we would need for the computation of φ_{321193} using $\sqrt{\text{élu}}$ is of degree 642384; in this case it might be faster to use a variant of Kohel's algorithm to avoid the extension field arithmetic (see Sect. 3.2). The extension field degrees for all the factors of f are given by

$$\begin{aligned}[k, q_f] = {} & [8, 5], [12, 7], [24, 13], [28, 43], [144, 73], [75, 151], [480, 241], \\ & [67, 269], [1152, 577], [1224, 613], [56220, 28111], [642384, 321193].\end{aligned}$$

The choice of $i = 7$ also means that we need to run Steps 3 to 5 of Algorithm 2 up to $3^7 \approx 2^{11}$ times. In particular, if the SIDH instantiation uses a fixed (arbitrary) starting curve, the computation of φ_f can be performed as a precomputation and the attack on an individual public key is relatively fast, just the computation of some $(2, 2)$-isogenies and 3-isogenies of elliptic curves, repeated potentially 3^7 times.

We have thus far restricted ourselves to e and B being a powers of two, as we want to demonstrate our attack and do not yet have adequate resources at our disposal to compute (ℓ, ℓ)-isogenies for $\ell > 2$. However, looking at the Microsoft challenge parameters can already illustrate the freedom that being able to compute efficiently (ℓ, ℓ)-isogenies for $\ell \neq 2$ can provide: we open up more options for attack parameters, including in this case in which one requires very little brute-force (only repeating Steps 4 to Step 5 up to 4 times): $A = 2^{110}$, $B = 3^{67}$, $A' = 2^{a-j} = 2^{108}$, $B' = 3^{b-i} = 3^{48}$, $e = 1$, and

$$f = 5 \cdot 7 \cdot 13 \cdot 61 \cdot 73 \cdot 431 \cdot 593 \cdot 607 \cdot 881 \cdot 36997 \cdot 139393 \cdot 227233.$$

The extension field degrees for all the factors of f are given by

$$\begin{aligned}[k, q_f] = {} & [8, 5], [12, 7], [24, 13], [60, 61], [144, 73], \\ & [860, 431], [1184, 593], [303, 607], [220, 881], \\ & [73992, 36997], [34848, 139393], [56808, 227233].\end{aligned}$$

NIST Level I Parameters: To select attack parameters for SIKEp434, that is, with $A = 3^{137}$ and $B = 2^{216}$, we rely on the algorithm for parameter selection outlined in the 'precomputation step' complexity analysis of Sect. 3. Table 1 shows some outputs of the algorithm for SIKEp434 parameters; these represent (i, j, x, y, z) such that

$$x3^{137-i} + y2^{216-j} = z.$$

We leave the details on the best parameter choice to further study, as all these parameters require a working implementation of (ℓ, ℓ)-isogenies for $\ell > 2$. Note that the last entry in the table only requires the computation of $(3, 3)$-isogenies, at the expense of some smoothness of $f = -yz$; the largest degree of elliptic-curve isogeny required in this choice is 11144321.

3.2 Computing the Cofactor Isogeny

First, notice that any finite subgroup of an elliptic curve appearing in the SIDH setting defines an $\mathbb{F}_{p^2}$-rational isogeny: this is simply because Frobenius equals a scalar multiplication for the supersingular elliptic curves employed by SIDH, hence stabilizes any subgroup by definition. Thus, when computing isogeny factors $\varphi_q : E_n \to E_{n+1}$ of φ_f, the rationality of E_{n+1} or of the images of rational points on E_n is no concern. Moreover, if Kohel's algorithm or the 'irrational' variant of the $\sqrt{\text{é}}$lu algorithm [1, Sect. 4.14] is used, evaluating the isogeny at points in some $E(\mathbb{F}_{p^r})$ can be done using arithmetic in $\mathbb{F}_{p^r}$ rather than (as is the case for Vélu and $\sqrt{\text{é}}$lu) the potentially much bigger composite of the fields of definition of the kernel points and the evaluation point.

In order to make an approximation of the complexity of computing φ_f on which we can base our search for good parameters for our attack, we ran some

Table 1. Some possible attack parameters for SIKEp434

i	j	x	y	z
19	27	$41 \cdot 2333$	$-101 \cdot 241$	$-5^4 \cdot 19 \cdot 47 \cdot 61 \cdot 857 \cdot 2903 \cdot 60889 \cdot 216617$ $\cdot 342497 \cdot 2309969 \cdot 2945407 \cdot 3951767 \cdot 4037069$
16	24	1823581	$-239 \cdot 6553$	$-11 \cdot 13 \cdot 19 \cdot 29 \cdot 631 \cdot 6043 \cdot 16451 \cdot 29759 \cdot 139987$ $\cdot 364513 \cdot 1850837 \cdot 3464849 \cdot 6344729 \cdot 26440207$
15	27	123551	-2546657	$-5^2 \cdot 29 \cdot 103 \cdot 1549 \cdot 28201 \cdot 55933 \cdot 243431$ $\cdot 1874903 \cdot 4421117 \cdot 6553021 \cdot 14183149 \cdot 39691591$
16	29	$5 \cdot 7^2 \cdot 1171$	-7884713	$-173 \cdot 853 \cdot 883 \cdot 8627 \cdot 26759 \cdot 692929 \cdot 3500557$ $\cdot 5202137 \cdot 6065333 \cdot 15108221 \cdot 28512793$
16	25	$79 \cdot 139 \cdot 499$	$-197 \cdot 47777$	$-5 \cdot 11 \cdot 17 \cdot 571 \cdot 35099 \cdot 40639 \cdot 48889 \cdot 81281$ $\cdot 138899 \cdot 1285429 \cdot 8464307 \cdot 13664309 \cdot 17314859$
16	24	$-467 \cdot 5419$	$5 \cdot 434689$	$-7 \cdot 103 \cdot 109 \cdot 2791 \cdot 3643 \cdot 36191 \cdot 47581 \cdot 99817$ $\cdot 401119 \cdot 749467 \cdot 2690497 \cdot 2863607 \cdot 3014203$
16	25	$-197 \cdot 9391$	$11 \cdot 307 \cdot 941$	$-5 \cdot 233 \cdot 431 \cdot 659 \cdot 4219 \cdot 237277 \cdot 371341 \cdot 820643$ $\cdot 2362589 \cdot 3896323 \cdot 14204429 \cdot 55510211$
17	26	-1	1	$-11 \cdot 23 \cdot 31 \cdot 131 \cdot 281 \cdot 311 \cdot 601 \cdot 3331 \cdot 8059 \cdot 8761$ $\cdot 163411 \cdot 1164091 \cdot 2101681 \cdot 4027511 \cdot 11144321$

experiments to investigate the behaviour of extension degrees for different values of p. As an illustration we consider $E_{1728}/\mathbb{F}_p$ with $p = 2^{216}3^{137} - 1$ as in the proposed NIST Level I parameters for SIKE. Only the even-degree fields are relevant as we are working with extensions of $\mathbb{F}_{p^2}$. Figures 1(a), (b), and (c) show the q_f for which there exists an even $k \leq 1000$ such that there is an $\mathbb{F}_{p^k}$-rational point of order q_f (only the minimal even k is depicted). In total, we find 72% of the primes $< 10^2$ (cf. Fig. 1(a)), 62.5% of the primes $< 10^3$ (cf. Fig. 1(b)), and 22% of the primes $< 10^4$ (cf. Fig. 1(c)). Based on these experiments, to guide our parameter selection for our attack we crudely estimate that the minimal field extension degree k for the maximal q_f dividing f is close to degree q_f over $\mathbb{F}_{p^2}$. Below, we will often use the fact $k \leq q_f$.

To compute with elements in an extension field of degree k, one requires an irreducible polynomial of that degree over the ground field (here, $\mathbb{F}_{p^2}$). There are many algorithms for this task. We specifically mention one approach due to Shoup [33], which has a complexity of $\tilde{O}(k^2 + k \log p)$ operations in $\mathbb{F}_p$.

To find a point of order q_f, we may then sample a random point $P \in E(\mathbb{F}_{p^k})$ and multiply it by a cofactor on the order of p^k. Using square-and-multiply, this amounts to $O(k \log p)$ multiplications in $\mathbb{F}_{p^k}$. Thus, finding a point of order q_f in this way costs $\tilde{O}(k^2 \log p)$ when using FFT-based multiplication for $\mathbb{F}_{p^k}$. Under the assumption that $\log p \in (\log q_f)^{O(1)}$, which would for instance follow from the heuristic estimates on f given above, this gives us a rough estimate of $\tilde{O}(q_f^2)$ for the complexity of computing the kernel of a φ_q-isogeny. Note that if the largest factor of the smoothest possible choice of f only admits very large extension fields, it will be worthwhile to opt for a slightly less smooth f, i.e., a slightly bigger q_f, for which the field extensions are smaller.

To compute a large-degree isogeny from an explicit kernel point over $\mathbb{F}_{p^k}$, we can either apply $\sqrt{\text{élu}}$ directly over $\mathbb{F}_{p^k}$ or first recover the kernel polynomial using [15, Algorithm 4] and then run Kohel's algorithm. The cost for the first method is $\tilde{O}(q_f^{1/2})$ arithmetic operations in $\mathbb{F}_{p^k}$, or $\tilde{O}(q_f^{3/2})$ operations in $\mathbb{F}_p$ using FFT-based multiplication in $\mathbb{F}_{p^k}$. The cost for the second method is $O(q_f^2)$. (Note that the first method will require working in composite extension fields to evaluate the isogeny at points, whereas the second gives an expression for the isogeny with coefficients in $\mathbb{F}_{p^2}$.)

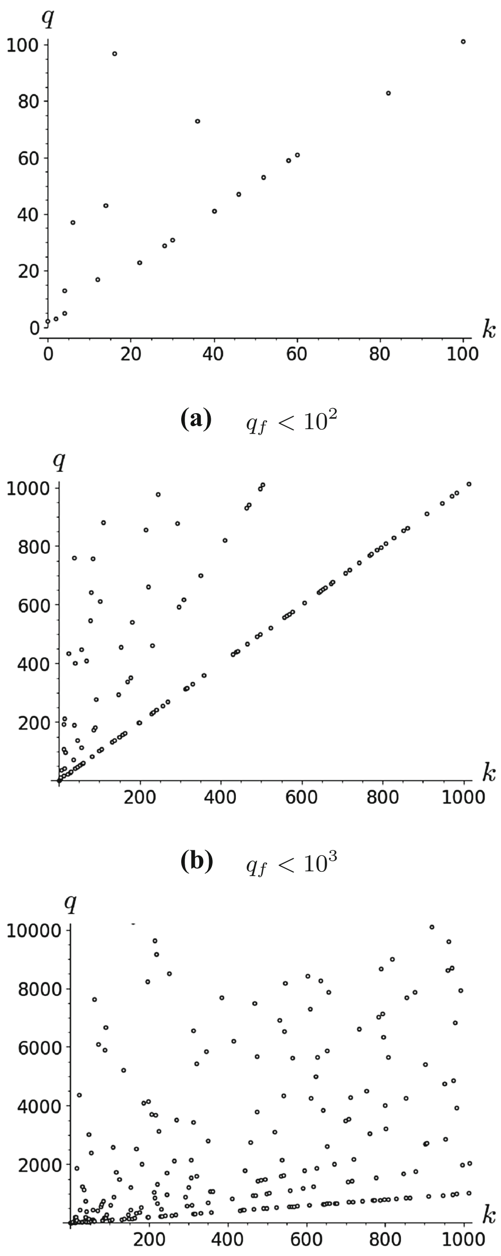
Overall, the dominating part of the algorithm is the large scalar multiplication to find the kernel of a q_f-isogeny. Therefore, to guide our choice of attack parameters, we take the complexity of computing and evaluating large-degree isogenies to be $\tilde{O}(q_f^2)$.

We mention in passing that the field extension degree can be halved whenever it is even, by using x-only elliptic-curve arithmetic.

An Alternative Approach. Instead of finding an irreducible polynomial for $\mathbb{F}_{p^k}$ and computing a large scalar multiplication, it is also possible to extract an isogeny kernel from the q_f-division polynomial directly, as follows.

The q_f-division polynomial for $E/\mathbb{F}_{p^2}$ is the unique monic squarefree polynomial with coefficients in $\mathbb{F}_{p^2}$ whose roots are precisely the x-coordinates of nonzero q_f-torsion points on E. It can either be precomputed for a generic curve E with symbolic coefficients (e.g., a single Montgomery coefficient A) or computed directly for a given E using a recursive expression [34, Exercise 3.7]. A careful analysis of both approaches to computing division polynomials is given in [2, Sect. 9]: Evaluation of a precomputed polynomial can be faster if q_f is fairly small, but once q_f is large enough that multiplying polynomials of degree q_f^2 benefits from FFT-based multiplication, it becomes faster to compute the polynomials instantiate for a given E directly. For these large q_f, the cost of computing the division polynomial is $O(q_f^2 \log q_f)$ base-ring operations.

Let $\mathbb{F}_{p^{2k}}$ be the smallest extension of $\mathbb{F}_{p^2}$ where the q_f-torsion is defined, and define $k' = k/2$ if k is even and $k' = k$ otherwise. All irreducible divisors of the division polynomial have degree k': for the curves used in SIDH, the p^2-Frobenius π equals $[-p]$, hence for any point $P = (x, y)$ of order q_f we have $\pi^k(P) = [(-p)^k]P = P$. Dropping the y-coordinate corresponds to quotienting the elliptic-curve group by negation, which shows $x^{k'} = x$, and k' is minimal with this property since k was minimal. Thus, the irreducible divisor of ψ_{q_f} which vanishes at x has degree k' as claimed. We may thus apply 'equal-degree splitting'—see e.g. [17, Algorithm 14.8])— recursively to find a single irreducible divisor h of ψ_{q_f}. This involves $O(d \log p + \log q_f)$ operations on polynomials of degree $O(q_f^2)$; assuming the use of FFT-based multiplication the cost in $\mathbb{F}_p$-operations is $\tilde{O}(q_f^3) \log p$. By construction h is then a minimal polynomial for a q_f-isogeny in the sense of [15, Definition 15]. We may compute the isogeny in time $O(k' q_f) + \tilde{O}(q_f)$ by running [15, Algorithm 3] and applying Kohel's algorithm. Overall, the cost for



(a) $q_f < 10^2$

(b) $q_f < 10^3$

(c) $q_f < 10^4$

Fig. 1. Extension field degrees < 1000 needed for $\mathbb{F}_{p^k}$-rational q_f-torsion

this is $\tilde{O}(q_f^3)\log p$, which is worse than finding an irreducible polynomial first and then running the multiplication-based method above.

3.3 Computing (ℓ, ℓ)-isogenies

In order for our algorithm to reach its full potential, it is necessary to consider integers e in Step 1 of Algorithm 2 that do not divide B, and in particular are not necessarily powers of two. It may also be that there is a nice parameter choice (e, i, j, f) with A a power of 2 and B a power of 3 (cf. the attack parameter suggestions in Sect. 3.1), or one may want to consider more general setups. In all of these cases, in Step 5 of Algorithm 2 it will be necessary to compute (ℓ, ℓ)-isogenies for $\ell \neq 2$, which as observed above requires more research to achieve practically (for $\ell = 3$ there is however already some interesting work on this topic [6]). For this reason, we leave all instantiations of the attack that use e not dividing B to future work and focus on the case of $(2, 2)$-isogenies, that is, $B = 2^b$ and $e \mid B$. Recall that we set $B' = B2^{-j}$, where $0 \leq j \leq b$.

In order to compute the chain of $(2, 2)$-isogenies whose composition is the (eB', eB')-isogeny Φ, we need to able to compute three different flavours of $(2, 2)$-isogenies between principally polarized abelian surfaces:

- A (2,2)-isogeny from a Jacobian of a genus 2 curve to a Jacobian of a genus 2 curve, for which we refer to reader to [37, Sect. 2.3.1].
- A (2,2)-isogeny from a product of elliptic curves to the Jacobian of a genus 2 curve, for which we refer the reader to [8] for more details. (This is required for the first step of Φ).
- A (2,2)-isogeny from a Jacobian of a genus 2 curve to a product of elliptic curves, for which we refer the reader to [35, Proposition 8.3.1]. (This is required for the last step of Φ).

Our proof-of-concept implementation uses Rémy Oudompheng and collaborators' SageMath implementation [28,29] for these steps.

4 The Case of Known Endomorphism Ring

Algorithm 1 solves SSI-T, assuming that $B > A$, and an isogeny $\varphi_f : E \to E_0$ of degree $B - A$ is known, in a way that allows efficient evaluation of $\widehat{\varphi_f}$ on the B-torsion. In this section, we describe how to find such an isogeny in polynomial time, provided E_0 and a description of the endomorphism ring $\text{End}(E_0)$.

More precisely, we prove the following theorem. An *efficient representation* of an isogeny φ is an encoding of the isogeny, together with an algorithm that can evaluate it on points in time polynomial in the length of the input.

Theorem 2. *Assume the generalised Riemann hypothesis. There is an algorithm that solves the following task in polynomial time (in the length of the input): given a supersingular curve E_0, four endomorphisms of E_0 in efficient representation, and a positive integer f, finds an isogeny $\varphi : E_0 \to E$ of degree f in efficient representation.*

Together with Corollary 1, this theorem immediately implies a polynomial time algorithm for SSI-T, when the endomorphism ring of E_0 is known, and assuming the generalised Riemann hypothesis (GRH).

Algorithm 3: Finding an ideal of prescribed norm.

```
Input: A basis $(\alpha_i)_{i=1}^4$ of $\mathrm{End}(E_0)$ in efficient representation, and an integer $f$
    coprime to 2 and $p$.
Output: A left ideal $I$ of norm $f$ in $\mathrm{End}(E_0)$
1 Find a solution of $\deg(\alpha_0) = z_0^2 f$ with $\alpha_0 \in \mathrm{End}(E_0)$ and $z_0 \in \mathbb{Z}$. It is a
    homogenous quadratic equations of dimension 5, so can be solved in polynomial
    time by [7].
2 Deduce another solution $(\alpha, z)$ for which $z$ is coprime with $f$, using the technique
    of [38, Algorithm 7, Step 3].
3 Return $I = \mathrm{End}(E_0)\alpha + \mathrm{End}(E_0)f$.
```

Proof of Theorem 2. The idea is the following: first, find an ideal I in $\mathrm{End}(E_0)$ of norm f. Then, assuming GRH, one can find the codomain of $\varphi = \varphi_I : E_0 \to E$ and evaluate φ on any input using [16, Lemma 3.3].

Finding the ideal I requires more explanation. First observe that the problem reduces to the case where f is coprime to $2p$: write $f = 2^i p^j f'$ with $(f', 2p) = 1$, solve the problem for f', and then compose the resulting isogeny with i isogenies of degree 2 and j Frobenius isogenies. The steps to find I are then given in Algorithm 3. Let us explain Step (2). Finding the desired solution heuristically is simple, so the motivation of the following discussion is mostly to get a provable method. Write the solutions (α, z) in the form $(x, z) \in \mathbb{Z}^4 \times \mathbb{Z}$, where x represents the coefficients of α in the provided basis of $\mathrm{End}(E_0)$. The equation can then be written as $x^t G x = z^2 f$, or $x^t Q x = 0$, where G is the Gram matrix of the basis, and $Q = G \oplus \langle -f \rangle$ (the 5×5 matrix with G in the upper-left corner, $-f$ in the lower-right corner, and zeros elsewhere). Note that we can assume that x_0 (the vector of coordinates of α_0) is primitive (i.e., the greatest common divisor of its coefficients is 1) and $z_0 \in \mathbb{Z}_{>0}$. We are looking for another solution where x is coprime with f. The rest of the proof reproduces *mutatis mutandi* the technique of [38, Algorithm 7, Step 3]. From [10, Proposition 6.3.2], the general solution $X = (x, z)$ is given by

$$X = d((R^t Q R)X_0 - 2(R^t Q X_0)R),$$

for arbitrary $R \in \mathbb{Q}^5$ and $d \in \mathbb{Q}^*$, where $X_0 = (x_0, z_0)$ is our initial solution. Fix $d = 1$. Write $R = (r_x, r_z)$ with $r_x \in \mathbb{Z}^4$ and $r_z \in \mathbb{Z}$. The last coordinate of X is given by the integral quadratic form

$$r_x^t G r_x z_0 - 2 r_x^t G x_0 r_z + f z_0 r_z^2 = \frac{(r_x z_0 - x_0 r_z)^t G (r_x z_0 - x_0 r_z)}{z_0}.$$

It is of rank 4, so let $M \in M_{4\times 4}(\mathbb{Z})$ be a matrix whose columns generate $\Lambda = z_0\mathbb{Z}^4 + x_0\mathbb{Z}$, and

$$g(v) = \frac{v^t(M^tGM)v}{z_0}.$$

It is positive definite, since G is and $z_0 > 0$. Let us show that g is (almost) primitive. If s is a prime that does not divide z_0, both M and z_0 are invertible modulo s, so g is primitive at s because G is. Now suppose $s \mid z_0$. Then, writing $Mv = r_xz_0 - x_0r_z$, we have

$$g(v) \equiv -2r_x^tGx_0r_z \bmod s.$$

Therefore, if $s \neq 2$ and $Gx_0 \not\equiv 0 \mod s$, then g is primitive at s. If $Gx_0 \equiv 0 \mod s$, since x_0 is primitive, s must divide $\mathrm{disc}(G)$, so s is 2 or p. This proves that the only primes where g might not be primitive are 2 and p. We can then write $g = g'/a$ where g' is primitive and a may only be divisible by the primes 2 and p. Applying [38, Proposition 3.6], we can find in polynomial time a v such that $z' = g'(v)$ is a prime larger than f. With $z = az'$, we obtain a solution of $x^tGx = fz^2$. Since f is coprime to $2p$, it is also coprime to z.

5 Future Work

We have provided an implementation of a toy example, but with a practical implementation of (ℓ,ℓ)-isogenies for $\ell > 2$ it should be possible to provide a practical implementation of larger interesting instances. Additionally, our implementation does not yet incorporate the fast (2,2)-isogeny formulas of Kunzweiler [21], which especially when working over field extensions will have a positive impact on performance.

Finally, given the speed of recovering the secret isogeny using our algorithm, especially in the case of known endomorphism ring, we also hope that it will be possible to use these methods for constructive purposes.

References

1. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. In: Galbraith, S. (ed.) ANTS XIV: Proceedings of the Fourteenth Algorithmic Number Theory Symposium, pp. 39–55. Mathematical Sciences Publishers (2020). https://iac.r/2020/341
2. Bernstein, D.J., Lange, T., Martindale, C., Panny, L.: Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 409–441. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_15
3. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 227–247. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_9

4. Bisson, G., Cosset, R., Robert, D.: AVIsogenies (abelian varieties and isogenies). MAGMA package. https://gitlab.inria.fr/roberdam/avisogenies
5. Bottinelli, P., de Quehen, V., Leonardi, C., Mosunov, A., Pawlega, F., Sheth, M.: The Dark SIDH of Isogenies. Preprint (2019). https://ia.cr/2019/1333
6. Bröker, R., Howe, E.W., Lauter, K.E., Stevenhagen, P.: Genus-2 curves and Jacobians with a given number of points. LMS J. Comput. Math. **18**(1), 170–197 (2015). https://doi.org/10.1112/S1461157014000461
7. Castel, P.: Solving quadratic equations in dimension 5 or more without factoring. Open Book Ser. **1**(1), 213–233 (2013)
8. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH (preliminary version). Preprint (2022). https://ia.cr/2022/975
9. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11274, pp. 395–427. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03332-3_15
10. Cohen, H.: Number Theory: Volume I: Tools and Diophantine Equations, vol. 239. Springer, New York (2008). https://doi.org/10.1007/978-0-387-49923-9
11. Costello, C.: B-SIDH: supersingular isogeny Diffie-Hellman using twisted torsion. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12492, pp. 440–463. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64834-3_15
12. Costello, C.: The case for SIKE: a decade of the supersingular isogeny problem. In: The NIST 3rd Post-Quantum Cryptography Standardization Conference (2021). https://ia.cr/2021/543
13. De Feo, L., et al.: Séta: supersingular encryption from torsion attacks. In: ASIACRYPT (4). LNCS, vol. 13093, pp. 249–278. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92068-5_9
14. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 64–93. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_3
15. Eriksen, J.K., Panny, L., Sotáková, J., Veroni, M.: Deuring for the People: Supersingular Elliptic Curves with Prescribed Endomorphism Ring in General Characteristic. Preprint (2023). https://ia.cr/2023/106
16. Fouotsa, T.B., Kutas, P., Merz, S., Ti, Y.B.: On the isogeny problem with torsion point information. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public Key Cryptography (1). LNCS, vol. 13177, pp. 142–161. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-97121-2_6
17. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 3rd edn. Cambridge University Press, Cambridge (2013)
18. Jao, D., et al.: Supersingular Isogeny Key Encapsulation. Submission to [27] (2017, 2019, 2020). https://sike.org
19. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 19–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2
20. Kani, E.: The number of curves of genus two with elliptic differentials (1997). https://doi.org/10.1515/crll.1997.485.93
21. Kunzweiler, S.: Efficient Computation of $(2^n, 2^n)$-Isogenies. Preprint (2022). https://ia.cr/2022/990
22. Lubicz, D., Robert, D.: Fast change of level and applications to isogenies. In: ANTS XV: Proceedings of the Fifteenth Algorithmic Number Theory Symposium (2022). https://doi.org/10.1007/s40993-022-00407-9

23. Lubicz, D., Somoza, A.: AVIsogenies SageMath package. https://gitlab.inria.fr/roberdam/avisogenies/-/tree/sage
24. Maino, L., Martindale, C.: An attack on SIDH with arbitrary starting curve. Preprint (2022). Version 2: https://eprint.iacr.org/archive/2022/1026/20220825:192029
25. Maino, L., Martindale, C.: An attack on SIDH with arbitrary starting curve. Preprint (2022). Version 1: https://eprint.iacr.org/archive/2022/1026/20220808:211318
26. Milne, J.S.: Abelian varieties. In: Cornell, G., Silverman, J.H. (eds.) Arithmetic Geometry, pp. 103–150. Springer, New York (1986). https://doi.org/10.1007/978-1-4613-8655-1_5
27. National Institute of Standards and Technology: Post-Quantum Cryptography Standardization, December 2016. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization
28. Oudompheng, R., Panny, L., Pope, G., et al.: SageMath Reimplementation of the SIDH key recovery attack (2022). https://github.com/jack4818/Castryck-Decru-SageMath
29. Oudompheng, R., Pope, G.: A note on Reimplementing the Castryck-Decru attack and lessons learned for SageMath. Preprint (2022). https://ia.cr/2022/1283
30. Petit, C.: Faster algorithms for isogeny problems using torsion point images. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 330–353. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_12
31. de Quehen, V., et al.: Improved torsion-point attacks on SIDH variants. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12827, pp. 432–470. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84252-9_15
32. Robert, D.: Breaking SIDH in polynomial time. Preprint (2022). https://ia.cr/2022/1038
33. Shoup, V.: Fast construction of irreducible polynomials over finite fields. J. Symb. Comput. **17**(5), 371–391 (1994). https://doi.org/10.1006/jsco.1994.1025
34. Silverman, J.H.: The Arithmetic of Elliptic Curves, vol. 106. Springer, New York (2009). https://doi.org/10.1007/978-0-387-09494-6
35. Smith, B.: Explicit endomorphisms and correspondences. Ph.D. thesis, University of Sydney (2005)
36. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.6) (2022). https://sagemath.org
37. Ti, Y.B.: Isogenies of Abelian Varieties in Cryptography. Ph.D. thesis, University of Auckland (2019)
38. Wesolowski, B.: The supersingular isogeny path and endomorphism ring problems are equivalent. In: 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, 7–10 February 2022, pp. 1100–1111. IEEE (2021). https://doi.org/10.1109/FOCS52979.2021.00109
39. Wesolowski, B.: Understanding and improving the Castryck-Decru attack on SIDH. Preprint (2022)