Multiprime Strategies for Serial Evaluation of eSIDH-Like Isogenies

Jason T. LeGrow[1⋆], Brian Koziel[2], Reza Azarderaskhsh[2]

[1] Department of Mathematics, Virginia Polytechnic Instutite and State University, Blacksburg, Virginia, USA
jlegrow@vt.edu

[2] Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, Florida, USA
{bkoziel2017,razarderakhsh}@fau.edu

Abstract. We present new results and speedups for the large-degree isogeny computations within the extended supersingular isogeny Diffie-Hellman (eSIDH) key agreement framework. As proposed by Cervantes-Vázquez, Ochoa-Jiménez, and Rodríguez-Henríquez, eSIDH is an extension to SIDH and fourth round NIST post-quantum cryptographic standardization candidate SIKE. By utilizing multiprime large-degree isogenies, eSIDH and eSIKE are faster than the standard SIDH/SIKE and amenable to parallelization techniques that can noticeably increase their speed with multiple cores. Here, we investigate the use of multiprime isogeny strategies to speed up eSIDH and eSIKE in *serial* implementations. These strategies have been investigated for other isogeny schemes such as CSIDH. We apply them to the eSIDH/eSIKE scenario to speed up the multiprime strategy by about 10%. When applied to eSIDH, we achieve a 7-8% speedup for Bob's shared key agreement operation. When applied to eSIKE, we achieve a 3-4% speedup for key decapsulation. Historically, SIDH and SIKE have been considerably slower than its competitors in the NIST PQC standardization process. These results continue to highlight the various speedups achievable with the eSIKE framework to alleviate these speed concerns. Though eSIDH and eSIKE are susceptible to the recent devastating attacks on SIKE, our analysis applies to smooth degree isogeny computations in general, and isogeny-based signature schemes which use isogenies of smooth (not necessarily powersmooth) degree.

Keywords: Isogeny-based cryptography · large-degree isogeny · post-quantum cryptography

⋆ Jason T. LeGrow is funded in part by New Zealand's Ministry of Business, Innovation, and Employment fund UOAX1933 and the Commonwealth of Virginia's Commonwealth Cyber Initative (CCI), an investment in the advancement of cyber R&D, innovation, and workforce development. For more information about CCI, visit www.cyberinitiative.org. This work was partially performed while the first author was in the Department of Mathematics at the University of Auckland.

1 Introduction

The impending implementation of large-scale quantum computers necessitates the development of post-quantum cryptosystems to ensure the continued safety of our communications systems. One family of post-quantum primitives are supersingular isogeny-based protocols, whose security is based on the presumed hardness of finding isogenies between supersingular elliptic curves (and many variants of this problem). One such protocol was Supersingular Isogeny Diffie-Hellman (SIDH) [12] key exchange, which underlies the NIST post-quantum cryptography (PQC) competition fourth round alternate candidate supersingular isogeny key encapsulation (SIKE) [2] mechanism which was recently broken [5, 16, 18]. Among post-quantum cryptosystems, SIDH and SIKE were appealing because of their small key sizes, which reduce both the communication and storage components when transferring or storing public keys, respectively. Unfortunately, SIDH and SIKE suffered from slow speeds, but these concerns continue to be alleviated as more applied research accelerated them.

In both SIDH and SIKE, the most time-consuming computational task that each party must perform is to determine the codomain of an isogeny which is of cryptographically-large but smooth degree. This is a large-degree isogeny computation. For the SIKE parameter sets which were considered for standardization [2, Section 1.6], Alice computes isogenies of degree 2^{e_A} for e_A approximately between 200 and 400, while Bob computes isogenies of degree 3^{e_B} for e_B approximately between 130 and 230. More exotic protocols built on SIDH have parties construct isogenies of the form ℓ^e where ℓ is a small prime and e is a small exponent which depends on the desired security level; for instance, 3-party key establishment of the kind given in [1, 13] uses isogenies of degree 5^{e_C} for a third party, Charlie.

Any isogeny ψ of degree $N = \ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}$ defined on an elliptic curve E defined over a field $\mathbb{F}$ of characteristic coprime to $\ell_1, \ell_2, \ldots, \ell_n$ factors as

$$\psi = \psi_{1,1} \circ \cdots \circ \psi_{1,e_1} \circ \psi_{2,1} \circ \cdots \circ \psi_{2,e_2} \circ \cdots \circ \psi_{n,1} \circ \cdots \circ \psi_{n,e_n} \tag{1}$$

where for each $1 \leq j \leq n$ and $1 \leq k \leq e_j$ we have $\deg \psi_{j,k} = \ell_j$. The best known algorithms for computing the codomain of an isogeny of prime degree ℓ requires time fully exponential in ℓ (*cf.* Vélu's formulas [19] and the recent improved Vélu's formulas [4]); however, for isogenies of degrees of the form used in SIKE or CSIDH, the factorization given in Equation (1) naturally yields a polynomial-time algorithm for computing the required isogenies, for *constant* values of $\ell_1, \ell_2, \ldots, \ell_n$ and polynomially-sized $e_1, e_2, \ldots, e_n$.

The isogenies of small prime degree which are chained together to construct the isogenies of cryptographically-large degree in SIKE and CSIDH are typically computed using Vélu's formulas; to compute an isogeny of degree ℓ_i, a point $P_i \in E(\mathbb{F}_{p^2})$ of order ℓ_i is required, and exactly which point P_i is required at each step depends on the protocol being used and the user's secret key. In the case of SIKE, Alice's secret key is an integer $0 \leq m_A \leq \ell_A^{e_A} - 1$, and (part of) her public ephemeral key is $\psi_A(E_0)$, where E_0 is a global public curve and ψ_A is the unique (up to composition with an isomorphism) isogeny whose kernel is

$\ker \psi_A = \langle P_A + m_A Q_A \rangle$, for some global public points P_A, Q_A which generate $E(\mathbb{F}_{p^2})[2^{e_A}]$. This isogeny factors as $\psi_A = \psi_A^{e_A} \circ \psi_A^{e_A - 1} \circ \cdots \circ \psi_A^1$ where for each $1 \leq j \leq e_A$, we have $\ker \psi_A^j = \langle [2^{e_A - j}] \psi_A^{j-1} \circ \psi_A^{j-2} \circ \cdots \circ \psi_A^1 (P_A + m_A Q_A) \rangle$. This suggests a straightforward method to compute ψ_A from $R_A^{(0)} = P_A + m_A Q_A$: first construct $[2^{e_A - 1}] R_A^{(0)}$, which generates $\ker \psi_A^1$. Using Vélu's formulas, construct $R_A^{(1)} = \psi_A^1(R_A^{(0)})$. Now construct $[2^{e_A - 2}] R_A^{(1)}$, which generates $\ker \psi_A^2$. Continue in this fashion, alternately multiplying by $[2^{e_A - j}]$ and applying ψ_A^j, until you have performed all e_A isogenies. This simple technique requires the application of $e_A - 1 = O(e_A)$ 2-isogeny evaluations, and $\sum_{j=1}^{e_A} (e_A - j) = \binom{e_A}{2} = O(e_A^2)$ point doublings, which is clearly polynomial-time in e_A.

Strategies have been proposed (first in [12] for SIDH and SIKE, and in [15] for CSIDH [6]) as faster alternatives to this naïve method. Formally, a strategy is a Steiner arborescence in a directed grid graph with particular root and terminals (depending on the degree of the isogeny to be computed) and with edge weights which encode the cost of two basic operations: prime-degree isogeny evaluation and scalar multiplication. A strategy corresponds to an algorithm for constructing an isogeny codomain by associating to each vertex a point on an elliptic curve and to each edge a basic operation. A strategy of minimal total weight thus corresponds to a fastest algorithm (from the class of algorithms which are in correspondence with strategies) for constructing the codomain of an isogeny. The straightforward algorithm of the previous paragraph corresponds to the "multiplication-based" strategy; generally, other strategies are more efficient.

Extended SIDH (eSIDH) [8] is a protocol derived from SIDH, in which Bob's prime-power-degree isogenies are replaced by isogenies whose degrees are a product of powers of small primes (3 and 5, for all proposed parameter sets in [8]). The most obvious benefit of this change is that Bob's computations are more amenable to parallelization; in particular, the kernel of an $3^{e_B} 5^{e_C}$ isogeny can be generated by a point $R_B = P_B + m_B Q_B$ of order 3^{e_B} and a point of $R_C = P_C + m_C Q_C$ of order 5^{e_C}, and these points can be constructed in parallel. To yield the same security level as Alice's 2^{e_A}-isogeny, it suffices to take e_B, e_C such that $e_B \log_2 3 + e_C \log_2 5 \approx e_A$; in particular, taking $e_B \log_2 3 \approx e_C \log_2 5 \approx \frac{e_A}{2}$, constructing R_B and R_C in parallel takes approximately half as long as computing R_A, which saves a significant amount of time over a parallel implementation of SIDH. Aside from parallelization, this method allows Bob to compute his isogeny using two small strategies rather than one large strategy; this is more efficient even in the serial setting than the corresponding computation in SIDH. We refer to the technique of using multiple small strategies as the "split prime" setting.

Unfortunately, this split prime optimization only yields a net benefit in the first round of eSIDH. In contrast, in the second round, expensive scalar multiplications are required to generate the necessary "auxiliary points" used in constructing the corresponding kernel generators. However, as the authors note in [8, Section 3.3], Bob can instead construct a single point R'_{BC} which generates the kernel of his full $3^{e_B} 5^{e_C}$-isogeny, and construct that isogeny using a single strategy of size $e_B + e_C$ which is formed from the smaller strategies from round 1. The authors of [8] refer to this as the "CRT-based" (Chinese Remainder The-

orem) approach; more broadly, we refer to a single strategy which is used to compute an isogeny of non-prime-power degree as a "multiprime strategy."

Contributions. In this work we consider multiprime strategies of a more general form than those of [8], as well as permutations of the multisets of small primes associated to the strategy, as in [15]. Using these more general strategies along with different techniques for secret key selection and kernel generator construction we accelerate Bob's large-degree isogeny by about 10% over the eSIDH implementations of [8] in the serial setting. In eSIDH, this equates to a 7-8% performance boost for Bob's shared secret generation. In eSIKE, this equates to a 3-4% performance boost for key decapsulation. We note that these performance boosts are achieved simply through pre-computation; the rest of the eSIDH/eSIKE algorithm remains the same. Next, we searched for other eSIDH/eSIKE-friendly primes, proposing new primes that our estimates suggest would provide a further 3% improvement in the isogeny computation. Lastly, we implement our multiprime stategies for some eSIDH primes in the eSIDH library to confirm these performance gains. We stress that these techniques apply to eSIDH variants of SIDH-based signatures based on zero-knowledge proofs of isogeny knowledge [11], such as those of [14], which remain secure despite the recent attacks on SIDH-based key establishment.

The rest of this paper is organized as follows: in Section 2 we give the necessary mathematical background for SIDH and eSIDH, which we then describe in Section 3. In Section 4 we discuss strategies abstractly, and then their applications to isogeny-based protocols. Finally, we present our implementation results in Section 5 and conclude in Section 6.

2 Mathematical Background: Isogenies

In this section we give a brief introduction to isogenies and how they are represented in SIDH and eSIDH. The contents of this section are adapted from [12, Section 2]. Note that SIKE and eSIKE are variants of SIDH and eSIDH, respectively, where the order of isogenies (among other things) is modified to achieve IND-CCA2 security. SIKE and eSIKE functions include key generation, key encapsulation, and key decapsulation.

Let E_1 and E_2 be elliptic curves defined over a finite field $\mathbb{F}_q$. An *isogeny* from E_1 to E_2 defined over $\mathbb{F}_q$ is a surjective rational map $\psi\colon E_1 \to E_2$ which is also a group homomorphism of $E_1(\overline{\mathbb{F}}_q)$ to $E_2(\overline{\mathbb{F}}_q)$. When an isogeny from E_1 to E_2 defined over $\mathbb{F}_q$ exists we say that E_1 and E_2 are $\mathbb{F}_q$*-isogenous*. Since each isogeny $\psi\colon E_1 \to E_2$ has a dual isogeny $\hat{\psi}\colon E_2 \to E_1$, the property of being isogenous is an equivalence relation on the set of elliptic curves defined over $\mathbb{F}_q$.

An *endomorphism* of an elliptic curve E defined over $\mathbb{F}_q$ is an isogeny $\psi\colon E \to E$ defined over $\mathbb{F}_{q^e}$ for some $e \in \mathbb{N}$. The set of endomorphisms E, together with the zero map, forms a ring, called the *endomorphism ring* of E, and denoted $\mathrm{End}(E)$. The endomorphism ring is isomorphic to an order either in a quadratic number field—in which case we say that E is *ordinary*—or in a quaternion algebra, in which case we say that E is *supersingular*.

3 eSIDH Protocol Description

Here we present eSIDH, to give context for the use of strategies in Section 4.

Concisely, eSIDH is constructed from SIDH [12] (described in Appendix A) by having Bob compute isogenies of non-prime-power degree. The protocol flow is much the same; the necessary changes are:

1. Fix $\ell_A = 2$, and choose distinct odd primes $\ell_1, \ell_2, \ldots, \ell_n$.
2. The prime takes the form $p = 2^{e_A}\ell_1^{e_1}\cdots\ell_n^{e_n}f - 1$ with $2^{e_A} \approx \ell_1^{e_1}\cdots\ell_n^{e_n}$
3. We require torsion bases $\{P_i, Q_i\}$ for $E[\ell_i^{e_i}]$ for $i = 1, 2, \ldots, n$, along with a basis $\{P_A, Q_A\}$ for $E[2^{e_A}]$. As well, set $P_B = P_1 + P_2 + \cdots + P_n$ and $Q_B = Q_1 + Q_2 + \cdots + Q_n$.
4. In Bob's key generation round, he chooses secrets $\beta_i \in \{0, 1, \ldots, \ell_i^{e_i} - 1\}$ and constructs the points $R_i = P_i + \beta_i Q_i$. He will construct n isogenies, whose kernels are given by

$$\begin{aligned}\ker\psi_1 &= \langle R_1\rangle\\ \ker\psi_i &= \langle\psi_{i-1}\circ\cdots\circ\psi_1(R_i)\rangle \text{ for } i \geq 2.\end{aligned}$$

   He also sets $\psi_B = \psi_n \circ \cdots \circ \psi_1$. His secret key is $\mathsf{sk}_B = (\beta_1, \ldots, \beta_n)$ and his public key is $\mathsf{pk}_B = (\psi_B(E), \psi_B(P_A), \psi_B(Q_A))$.
5. In Bob's key establishment round, he constructs the points

$$R'_i = \prod_{j\neq i}\ell_j^{e_j}\,(S_A + \beta_i T_A) \text{ for } i = 1, 2, \ldots, n$$

   (where $S_A = \psi_A(P_B)$ and $T_A = \psi_A(Q_B)$) and uses them to construct isogenies $\psi'_1, \ldots, \psi'_n, \psi'_B$ as in his key generation round. His key is $K_B = j(\psi_B(E_A))$.

The structure of Bob's computations in eSIDH. As the authors note in [8, Section 3.3], Bob's isogeny construction does not need to be decomposed into prime-power-order isogenies; instead, he can construct a generator R_B of $\ker\psi_B$ by "combining" his secret values $\beta_1, \ldots, \beta_n$ using the Chinese remainder theorem. This is valuable in the second round of eSIDH, since constructing the points R'_i requires many costly point multiplications. This would allow Bob to use a single strategy (which we describe in the next section) to construct his key K_B.

Note that this idea works "in reverse" as well; rather than choosing $\beta_1, \ldots, \beta_n$, Bob can choose $\beta \in \{0, 1, \ldots, \ell_1^{e_1}\cdots\ell_n^{e_n} - 1\}$ and then set each $\beta_i = \beta \mod \ell_i^{e_i}$.

4 Strategies and their Applications to (e)SIDH

Strategies were first introduced in [12, Section 4.2.2] (as "full, well-formed" strategies) as a method to define algorithms for computing prime-power-degree isogenies. They were later slightly reformulated in [15, Section 2.1] to be more compatible with certain optimization techniques relevant to CSIDH—we present (a slightly modified version of) that formulation here.

Definition 1 (The Triangle Graphs T_n). *For $n \in \mathbb{N}$, we denote by T_n the directed graph whose vertices and edges are*

$$\mathcal{V}(T_n) = \{\vec{x} \in \mathbb{Z}^2 \,:\, x_1 + x_2 \leq n-1 \text{ and } x_1, x_2 \geq 0\}$$
$$\mathcal{E}(T_n) = \left\{(\vec{x}, \vec{y}) \in \mathcal{V}(T_n)^2 \,:\, \vec{y} - \vec{x} \in \left\{(1,0),(0,1)\right\}\right\}$$

We call T_n the triangle graph *of side n.*

Definition 2 (Steiner Arborescence). *Let G be a graph, and let $r \in \mathcal{V}(G)$ and $L \subseteq \mathcal{V}(G) \setminus \{r\}$. A* Steiner arboresence *for (G, r, L) is a subgraph S of G such that:*

1. *For each $t \in L$, S contains a directed path from r to t, and;*
2. *S contains no undirected cycles.*

We call r the root *of the arborescence, and L the* terminals.

Definition 3 (Strategy). *A* strategy *S of size n is a Steiner arborescence for (T_n, r, L_n), where $r = (0,0)$ and $L_n = \{\vec{x} \in \mathcal{V}(T_n) \,:\, x_1 + x_2 = n-1\}$.*

Definition 4 (The Join Operator). *Given two strategies S_1, S_2 of sizes n_1 and n_2, respectively, we define their join, denoted $S_1 \# S_2$ to be the strategy of size $n_1 + n_2$ whose edges are*

$$\begin{aligned}\mathcal{E}(S_1 \# S_2) = &\left\{\left(\vec{x} + (0, n_2), \vec{y} + (0, n_2)\right) \,:\, (\vec{x}, \vec{y}) \in \mathcal{E}(S_1)\right\} \\ &\sqcup \left\{\left(\vec{x} + (n_1, 0), \vec{y} + (n_1, 0)\right) \,:\, (\vec{x}, \vec{y}) \in \mathcal{E}(S_2)\right\} \\ &\sqcup \left\{\left((x,0),(x+1,0)\right) \,:\, x = 0, 1, \ldots, n_1 - 1\right\} \\ &\sqcup \left\{\left((0,y),(0,y+1)\right) \,:\, y = 0, 1, \ldots, n_2 - 1\right\}\end{aligned}$$

More intuitively, $S_1 \# S_2$ is the subgraph of $T_{n_1+n_2}$ which contains: A path from $(0,0)$ to $(n_1, 0)$; A path from $(0,0)$ to $(0, n_2)$; A copy of S_1, shifted n_2 units up, and; A copy of S_2, shifted n_1 units right. We note that the join operator is both non-commutative and non-associative.

Of particular interest are so-called *canonical strategies*:

Definition 5 (Canonical Strategy). *A strategy S of size n is* canonical *if: $n = 1$, or $S = S_1 \# S_2$, where S_1 and S_2 are canonical strategies.*

When S is a canonical strategy we let $S^L = S_1$ and $S^R = S_2$ denote its *left* and *right substrategies*, respectively.

Figure 1 depicts a canonical strategy in T_9 and highlights its left and right substrategies.

For optimization purposes we will need to assign weights to the edges of a strategy, which will be inherited from an assignment of weights to a triangle graphs. In this work weights are assigned to triangle graphs by *measures*.

Definition 6 (Measure). *A measure of size n is a triple $M = (\{\ell_i\}_{i=1}^n, f_H, f_V)$ where $\{\ell_i\}_{i=1}^n$ is a sequence of positive numbers, and $f_H, f_V \colon \mathbb{R}_+ \to \mathbb{R}_+$ are a pair of* weight functions.

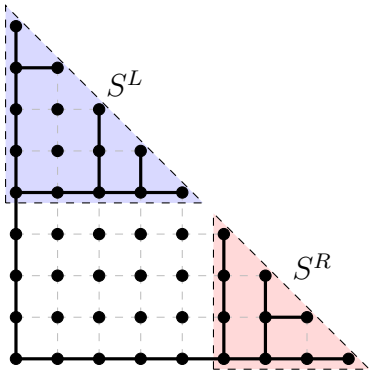

Fig. 1. A canonical strategy S of size 9 (black edges) and its left and right substrategies (blue and red shaded regions, respectively) embedded in T_9 (vertices, black edges, and dashed grey edges).

A measure $M = (\{\ell_i\}_{i=1}^n, f_H, f_V)$ assigns weights to T_n as follows:

- Each edge of the form $e = ((x-1, y), (x, y))$ has weight $w_e^{(M)} = f_H(\ell_x)$.
- Each edge of the form $e = ((x, y-1), (x, y))$ has weight $w_e^{(M)} = f_V(\ell_{n-y+1})$.

A strategy S is assigned by M the weights it inherits from T_n. The *cost* of a strategy S with respect to measure M is denoted $(S)_M$, defined as

$$(S)_M = \sum_{e \in \mathcal{E}(S)} w_e^{(M)}.$$

4.1 Applying Strategies and Measures to Isogeny-Based Protocols

In isogeny-based protocols, strategies are used to define algorithms for computing the codomain of and evaluating smooth-degree isogenies. In this section we discuss the connection between strategies and isogeny algorithms in the contexts of SIDH and CSIDH. We also discuss the relevance of measures in this context.

Strategies and Measures in SIDH. Without loss of generality let us consider Alice's computations only; Bob's computations are analogous. For simplicity of notation, we will omit the subscript A in Alice's computations. In SIDH, Alice must construct the codomain of an isogeny ψ of degree ℓ^e (in SIKE standardization candidate parameter sets we always have $\ell = 2$, but this is not strictly necessary) whose kernel is $\ker \psi = \langle P + mQ \rangle$ where P, Q are public and generate $E[\ell^e]$, and m is chosen by Alice. As discussed in Section 1, this is done by constructing the generators of the kernels of e ℓ-isogenies, whose kernels are given by

$$\ker \psi_j = \Big\langle \overbrace{[\ell^{e-j}]\psi_{j-1} \circ \psi_{j-2} \circ \cdots \circ \psi_1(P + mQ)}^{R_j} \Big\rangle.$$

For $0 \leq j \leq e$, let E_j denote the curve $\psi_j \circ \psi_{j-1} \circ \cdots \circ \psi_1(E_0)$. We decorate the graph T_e by assigning:

- To each horizontal edge $e = ((x-1, y), (x, y))$ the map $P \mapsto [\ell]P$;
- To each vertical edge $e = ((x, y-1), (x, y))$ the map $P \mapsto \psi_y(P)$;
- To the vertex $(0, 0)$ the point $R_{0,0} = P + mQ$, and;
- To each vertex (x, y) a point $R_{x,y}$ on E_y, obtained by applying the maps corresponding to the edges in any path from $(0, 0)$ to (x, y) to the point $R_{0,0}$.

We note that since isogenies are group homomorphisms the maps ψ_i commute with mutlipltication-by-ℓ, and so any two paths from $(0, 0)$ to (x, y) will yield the same point $R_{x,y}$, so this decoration is well-defined. Any strategy S of size e inherits these decorations from T_e. By [12, Lemma 4.2], this decoration corresponds to an algorithm to compute the isogeny ψ with $\ker \psi = \langle P + mQ \rangle$, by first constructing the point $R_{0,0}$, and then applying the maps corresponding to the edges of S, in depth-first, bottom-first order, noting that the points $R_{e-y,y-1}$ generate the kernels of the ψ_y. This decoration is depicted in Figure 2.

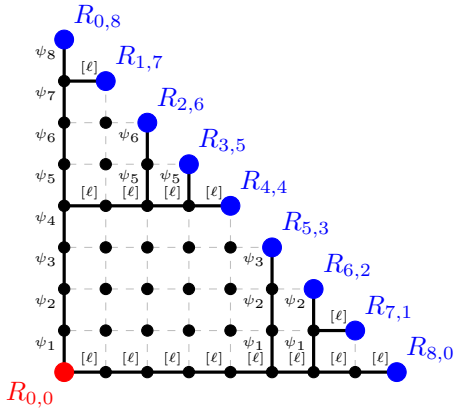

Fig. 2. A strategy decorated in the SIDH style. The root point $R_{0,0}$ is labelled and highlighted in red, and the terminals $R_{e-y,y-1}$ for $y = 1, 2, \ldots, e$ are labelled and highlighted in blue.

If we assign weights to a strategy S by the measure $M = (\{\ell_j\}_{j=1}^{e}, f_H, f_V)$, where $\ell_j = \ell$ for all j, and $f_H(\ell)$ and $f_V(\ell)$ are the cost of evaluating $P \mapsto [\ell]P$ and $P \mapsto \psi_j(P)$ for any j (this cost does not depend on j, only on the degree of the isogeny, which is ℓ), then $(S)_M$ is precisely the cost of computing Alice's isogeny ψ. Thus the authors of [12] use strategies which have minimal weight with respect to this measure to construct their isogeny construction algorithms.

Adapting measures to eSIDH. Taking $e = e_1 + e_2 + \cdots + e_n$ and the sequence $\{\ell_i\}_{i=1}^{e}$ to be (some permutation of) $\underbrace{\ell_1, \ell_1, \ldots \ell_1}_{e_1}, \underbrace{\ell_2, \ell_2, \ldots \ell_2}_{e_2}, \ldots, \underbrace{\ell_n, \ell_n, \ldots \ell_n}_{e_n}$ as

Table 1. Cost of functions in SIDH in quadratic extension field arithmetic

Operation	**M**	**S**	**a**	Normalized Cost ($\mathbf{S} = 0.66\,\mathbf{M}$, $\mathbf{a} = 0.05\,\mathbf{M}$)
Ladder Step	7	4	8	10.04
xTPL	7	5	10	10.8
xQPL	11	6	14	15.66
eval3Iso	4	2	4	5.52
eval5Iso	8	2	8	9.72
get3Iso	2	3	13	4.63
get5Iso	6	6	0	9.96

described in Section 3, then $(S)_M$ is the cost of constructing Bob's isogeny in eSIDH for a prime p of the form $p = 2^{e_A}\ell_1^{e_1}\cdots\ell_n^{e_n}f - 1$.

4.2 Optimized Strategies for Multiprime Large-Degree Isogenies

We propose new multiprime strategies for large-degree isogenies in the eSIDH landscape. In particular, this only applies to *Bob's* large-degree isogeny, which is of the form $\ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}$. In eSIDH [8], Bob's large-degree isogeny is of the form $3^{e_B}5^{e_C}$, where $3^{e_B} \approx 5^{e_C}$ and $2^{e_A} \approx 3^{e_B}5^{e_C}$. In the first case, $3^{e_B} \approx 5^{e_C}$ so that the eSIDH kernel generations $R_B = P_B + n_BQ_B$ and $R_C = P_C + n_CQ_C$ can be efficiently parallelized into 2 cores to reduce kernel generation latency. The second case, $2^{e_A} \approx 3^{e_B}5^{e_C}$ so that Alice and Bob perform approximately the same magnitude of isogeny.

eSIDH proposes efficient primes with a similar security as the SIKE parameter levels [2]. These correspond to NIST security levels, ranging from 1 to 5, where SIKE has parameter sets at levels 1, 2, 3, and 5. NIST security level 1 is conjectured to be as hard to break as a brute-force attack on AES128, NIST security level 2 is conjectured to be as hard to break as finding a hash collision in SHA256, NIST security level 3 is conjectured to be as hard to break as a brute-force attack on AES192, and NIST security level 5 is conjectured to be as hard to break as a brute-force attack on AES256. For an initial experiment, we compared the cost of multiprime large-degree isogenies for eSIDH primes $p_{443} = 2^{222}3^{73}5^{45} - 1$ and $p_{765} = 2^{391}3^{119}5^{81} - 1$.

Building a cost model for the isogeny operations. Similar to computing optimal strategies for SIDH/SIKE, one must first identify the weights of the edges in the strategy graphs. For this purpose, we used the fastest formulas for 3 and 5-isogenies as available in the literature. Luckily, these formulas were available in the literature as they were essential for speeding up CSIDH isogenies of any odd prime. In general, the optimized 3-isogeny formulas come from [9] and 5-isogeny formulas (and higher degree) come from [7]. We summarize the cost of these formulas in terms of finite field arithmetic multiplication, squaring, and addition in Table 1. In the realm of SIDH and SIKE, these finite field operations are in quadratic extension field $\mathbb{F}_{p^2}$. To give a normalized cost of

operations, we assumed that the cost of $\mathbb{F}_{p^2}$ squaring is approximately 2/3 the cost of $\mathbb{F}_{p^2}$ multiplication, and $\mathbb{F}_{p^2}$ addition is approximately 1/20 the cost of $\mathbb{F}_{p^2}$ multiplication. This generates a single value with which we can compare the cost of a strategy. Other cost models can be made, depending on the selected device. We chose $S = 0.66M$ because there are 2 $\mathbb{F}_p$ multiplication operations in $\mathbb{F}_{p^2}$ squaring and 3 $\mathbb{F}_p$ multiplication operations in $\mathbb{F}_{p^2}$ multiplication.

Table 1 lists the costs of large subroutines in SIDH and SIKE. The large-degree isogeny operation uses small point multiplication operations (xTPL for $Q = 3P$, xQPL for $Q = 5P$), small isogeny evaluations where a point is pushed from one elliptic curve to an isogenous curve (eval3Iso for pushing a point through a 3-isogeny mapping, eval5Iso for pushing a point through a 5-isogeny mapping), and small isogeny computations where you compute an isogenous mapping of a small degree (get3Iso for computing an isogeny mapping of degree 3 from a point of order 3, get5Iso for computing an isogeny mapping from a point of order 5). In order to create a multiprime strategy, the cost of point multiplication and isogeny evaluation creates the weighting of the large-degree isogeny graph. In particular, a horizontal edge is a point multiplication by ℓ and a vertical edge is an $\ell-$degree isogeny evaluation. The isogeny computations are computed at the leaf nodes, which is done regardless of strategy.

Finding Multiprime Strategies. As in CSIDH, when constructing strategies in the multiprime setting there are two orthogonal algorithmic concepts to optimize: the strategy itself, and the permutation of the list of primes, which determines the order in which the small prime degree isogenies are computed. As shown in [15] for CSIDH, there are efficient techniques to construct optimal strategies for fixed permutations (using dynamic programming, as in [12]) and to construct an optimal permutation of the primes for fixed strategies (using linear programming); however, it is not presently known how to construct globally optimal (permutation, strategy) pairs. For the purposes of this work, we construct (permutation, strategy) pairs using a straightforward randomized alternating algorithm: we choose a random starting permutation, and then alternately optimize our strategy (fixing the permutation) and our permutation (fixing the strategy) until the (permutation, strategy) pair stabilizes. The resulting pair is not globally optimal, in general, so we run 10 trials with different random starting permutations, and then choose the best resulting (permutation, strategy) pair for implementation.

4.3 Evaluating the Costs of Multiprime Strategies

Based on the cost models that we proposed in the previous section and the obtained multiprime strategies, we can evaluate the cost of these new strategies and compare them to the state-of-the-art. We note that the implementation is slightly different in the split-prime and multiprime strategy. First, the split-prime uses a different method to generate the secret kernels. In split-prime eSIDH, Bob initially computes two kernel generators $R_B = P_B + n_B Q_B$ and $R_C = P_C + n_C Q_C$, which have order 3^{e_B} and 5^{e_C}, respectively. Thus, the split-prime strategy will first compute a large-degree isogeny of degree 3^{e_B} over kernel R_B

Table 2. SIDH large operation costs for Bob's round functions

SIDH Round	Operation	Split Prime	Multi Prime	Single Prime
		$p_{443} = 2^{222}\mathbf{3^{73}5^{45}} - 1$		$p_{434} = 2^{216}\mathbf{3^{137}} - 1$
Round 1	Kernel	2,219	2,209	2,179
	Isogeny	10,479	11,027	10,757
	Total	12,698	13,236	12,935
Round 2	Kernel	3,712	2,209	2,179
	Isogeny	7,958	8,506	8,488
	Total	11,670	10,715	10,667
		$p_{765} = 2^{391}\mathbf{3^{119}5^{81}} - 1$		$p_{751} = 2^{372}\mathbf{3^{239}} - 1$
Round 1	Kernel	3,795	3,775	3,795
	Isogeny	19,297	20,392	20,345
	Total	23,092	24,168	24,141
Round 2	Kernel	6,349	3,775	3,795
	Isogeny	14,965	16,060	16,388
	Total	21,313	19,835	20,183

and then a second large-degree isogeny of degree 5^{e_C} over kernel R_C (this order could be flipped if it is faster). The caveat here is that whichever large-degree isogeny is computed first, you have to apply the small degree isogeny evaluations to the other kernel. The multiprime strategy is different in that you compute only a single kernel, $R_{BC} = P_{BC} + n_{BC}Q_{BC}$. This single kernel is then used to compute a large-degree isogeny of order $3^{e_B}5^{e_C}$ using a strategy.

We summarize the total cost of Bob's large-degree isogeny operations over the split prime strategy, multiprime strategy, as well as baseline SIDH/SIKE single prime strategy in Table 2. These are provided for the smallest and largest SIKE parameter sets. The two largest operations are kernel generation, where a double-point multiplication generates the secret kernel, and then the large-degree isogeny where you compute the isogeny over that secret kernel. The split prime strategy generates 2 kernels. This is based on the calculated number of functions used in each of the strategies, where each function cost is taken from Table 1. This estimate does not include the cost of some other functions, such as the finite field inversion or setup, which is expected to be a similar cost amongst SIDH/SIKE primes.

We further break down the cost of operations in Table 3. Here, we list the major SIDH/SIKE operations as well as the functions that are used many times within. For each function, we also specify whether its cost is included in the first or second round of Bob's large-degree isogeny operation. In the first round, Bob can use the public parameters to generate the kernel, whereas the second round uses Alice's public key to generate the kernel. As is specified in the split prime

Table 3. SIDH round breakdown of costs in Bob's large operations

Op	Function	SIDH Round		Split	Multi	Single
		R1?	R2?	Prime	Prime	Prime
				p_{443}		p_{434}
Kernel	Ladder Step	✓	✓	221	220	217
	xTPL		✓	73	0	0
	xQPL		✓	45	0	0
Isogeny	xTPL	✓	✓	175	176	466
	xQPL	✓	✓	94	154	0
	eval3Iso	✓	✓	382	399	511
	eval5Iso	✓	✓	175	124	0
	get3Iso	✓	✓	73	73	137
	get5Iso	✓	✓	45	45	0
	R1 eval3Iso	✓		219	219	411
	R1 eval5Iso	✓		135	135	0
				p_{765}		p_{751}
Kernel	Ladder Step	✓	✓	378	376	378
	xTPL		✓	119	0	0
	xQPL		✓	81	0	0
Isogeny	xTPL	✓	✓	315	316	913
	xQPL	✓	✓	220	306	0
	eval3Iso	✓	✓	684	721	982
	eval5Iso	✓	✓	307	259	0
	get3Iso	✓	✓	119	119	239
	get5Iso	✓	✓	81	81	0
	R1 eval3Iso	✓		357	357	717
	R1 eval5Iso	✓		243	243	0

strategy in eSIDH [8], Bob's first round can efficiently compute kernels of order 3^{e_B} and 5^{e_C} because the public parameters include generator points of these orders. For public key size efficiency, Alice applies her large-degree isogeny only over public torsion points of order $3^{e_B}5^{e_C}$. Thus, Bob's second round with the multiprime strategy requires a significant number of point triplings and quintuplings to convert his computed secret kernel of order $3^{e_B}5^{e_C}$ into two secret kernels of order 3^{e_B} and 5^{e_C}. For serial implementations, the magnitude of the Montgomery ladder steps is approximately the same between the three types of strategies. The split strategy suffers significantly in the second round to generate the kernels of correct order.

For the large-degree isogeny operation, we note that the split prime strategy outperforms the multi prime and single prime strategy by about 5% in both rounds 1 and 2. In the view of the strategy, this is to be expected as the split prime strategy begins with a head-start as many of the point multiplications are already computed to get the two kernels. The multiprime strategy is approximately on-par with the single prime strategy, which shows that the inclusion of 5-isogeny operations were efficiently interleaved with the 3-isogeny operations. We note that the R1 eval3Iso and R1 eval5Iso function count is the number of isogeny evaluations applied to the other party's torsion basis, which is only done for the first round.

When we consider the entire round function for SIDH, the split prime strategy is superior for round 1 (key generation), but the multiprime and single prime strategies are superior for round 2 (shared secret generation). The first round's split prime strategy is faster primarily as a result of its more efficient isogeny. However, the second round split prime strategy suffers as it is expensive to generate the kernels. The split prime strategy is about 5% faster for the first round, but the multiprime strategy is about 10% faster for the second round.

4.4 Expanding eSIDH to More Primes

In [8], the authors searched for eSIDH-friendly primes by the following criteria:

1. eSIDH primes of the form $p = 4^{e_A}3^{e_B}5^{e_C}f - 1$, where e_A is the number of 4-isogenies that Alice performs, e_B is the number of 3-isogenies that Bob performs, e_C is the number of isogenies that Bob performs, and f is a number that makes the number prime.
2. $4^{e_A} \approx 3^{e_B}5^{e_C}$,
3. $3^{e_B} \approx 5^{e_C}$,
4. $p \equiv \pm \bmod 2^{\gamma w}$, i.e. the prime has many words that are "0xFFFF..." such as for $w = 64$ for efficient modular multiplication.

The first criterion is necessary to guarantee that the $E[4^{e_A}], E[3^{e_B}], E[5^{e_C}] \subseteq E(\mathbb{F}_{p^2})$ (ensuring that the isogenies can be computed using arithmetic in $\mathbb{F}_{p^2}$, rather than a higher degree extension), while the second is required for the protocol to be secure. The third criterion is a heuristic for optimality in the parallel setting, while the fourth yields only minor computational improvements for software processors; in this paper we experiment with how eSIDH performs with multiprime strategies in the serial setting, so we simplify the search for eSIDH serial-efficient primes by removing the third and fourth criteria above. In addition, instead of limiting to only two primes for Bob, we also experiment with the base prime 7.

Another benefit of the eSIDH framework is that it opens up a vast number of primes that can be used and optimized. For instance, the chance of an odd number of the form $p = 4^{e_A}3^{e_B}f - 1$ is approximately $2/\ln p$. For a 434-bit number as is used as a prime to specify SIKE's NIST level 1 parameter set, there is approximately a 1 in 150 chance that the number is actually prime. With the prime search criterion that $4^{e_A} \approx 3^{e_B}$, this further limits the pool of

Table 4. Costs of NIST Security Level 1 multiprime strategies considered in this paper in terms of field multiplications, under three standard cost models. In the SIDH/eSIDH landscape, this cost is the "second round" cost.

Isogeny degree	Cost Model		
	$\mathbf{S} = 0.8\,\mathbf{M}$ $\mathbf{a} = 0\,\mathbf{M}$	$\mathbf{S} = 0.8\,\mathbf{M}$ $\mathbf{a} = 0.05\,\mathbf{M}$	$\mathbf{S} = 0.66\,\mathbf{M}$ $\mathbf{a} = 0.05\,\mathbf{M}$
$3^{73}5^{45}$	8858.4	9230.9	8752.3
$3^{97}5^{27}$	8535.2	8915.1	8430.8
$3^{132}5^{3}$	8388.2	8794.1	8289.5
$3^{91}5^{25}7^{5}$	8614.2	8990.6	8506.3
$3^{93}5^{25}7^{4}$	8610.2	8988.4	8502.7
$3^{109}5^{15}7^{3}$	8500.4	8888.4	8396.5

good parameters. If one isogeny graph is significantly larger than the other, then one party's computations will be asymptotically more expensive.

By changing the form of the number to $p = 4^{e_A}3^{e_1}5^{e_2}\ldots\ell_n^{e_n}f - 1$, and using the stipulation $4^{e_A} \approx 3^{e_1}5^{e_2}\ldots\ell_n^{e_n}$, we greatly increase the pool of numbers that can potentially be used as parameter sets. We used this relaxed methodology to find many different eSIDH primes. We highlight some NIST Security Level 1 parameter sets we found in Table 4. Among the three different cost models, we see that the fastest large-degree isogeny $3^{132}5^3$ uses the most 3-isogenies. This is to be expected as the larger isogeny formulas are not nearly as optimized as the 3-isogeny formulas.

For our recommended prime parameter sets, we chose 3- and 5-isogeny primes with a preference for more 3-isogenies. We were also able to ensure that the magnitude of the large-degree isogeny is approximately the same as Alice's large-degree isogeny. In the SIKE parameter set, this is especially imbalanced for $p_{751} = 2^{372}3^{237}$, where 3^{237} is over 100 times larger than 2^{372}. Our proposed eSIDH primes are shown in Table 4.4. Across the board, we achieve about a 3% performance improvement by going with a multiprime strategy-friendly prime.

5 Software Implementation

As a further investigation to the effectiveness of our multiprime strategy, we implemented our multiprime strategies on top of the eSIDH version 2.0 library*. This library was based on the SIKE team's implementation, SIDH Library v3.2. This eSIDH library features support for SIKE parameter sets $p_{434} = 2^{216}3^{137} - 1$ and $p_{751} = 2^{372}3^{239} - 1$, and eSIDH/eSIKE parameter sets $p_{443} = 2^{222}3^{73}5^{45} - 1$ and $p_{765} = 2^{391}3^{119}5^{81} - 1$. We applied no modifications to the lower-level finite field arithmetic. We ran our modified code on an Intel i7-8650u processor running at 1.9 GHz. All tests were run on a single core with turbo boost disabled.

* Commit b8f4486 at https://github.com/dcervantesv/eSIDH

Table 5. eSIDH/eSIKE timing results on Intel i7-8650u processor.

Scheme	Operation	Timings (Mcycles)		Improvement
		Split Prime	Multiprime	
$p_{443} = 2^{222}3^{73}5^{45} - 1$				
eSIDH	Bob R1	7.44	7.75	-4.03%
	Bob R2	**7.00**	**6.47**	**8.24%**
eSIKE	Keygen	7.43	7.74	-4.01%
	Decap	**13.71**	**13.17**	**4.10%**
$p_{765} = 2^{391}3^{119}5^{81} - 1$				
eSIDH	Bob R1	27.14	28.37	-4.34%
	Bob R2	**25.56**	**23.90**	**6.96%**
eSIKE	Keygen	27.14	28.39	-4.42%
	Decap	**50.47**	**48.83**	**3.36%**

Our eSIDH and eSIKE timing results are summarized in Table 5. These results summarize the affected SIDH and SIKE operations (only Bob's large-degree isogeny is affected by the changes). As we can see, the multiprime strategy results in a 4% slowdown for Bob's first round, but an 8% speedup for Bob's second round, when considering p_{443}. For p_{765}, Bob's first round is again 4% slower and his second round is 7% faster. For the SIKE operations, key generation is almost identical to Bob's round 1 in SIDH, resulting in about a 4% slowdown. SIKE key decapsulation then uses Bob's round 2 in SIDH, where we see a 4% speedup for p_{443} and a 3% speedup for p_{765}.

Implementing Multiprime Strategies. When implementing the multiprime strategy, we used a very similar algorithm as that of SIDH/SIKE. A strategy describes the order in which we traverse from the root of the large-degree isogeny strategy graph to its leaves. The primary difference is that we define a prime list that includes the order in which isogenies are performed. In the single prime strategy this is unnecessary, but is needed in the multiprime isogenies as you are mixing the order of prime isogenies. We defined this list as the order they are to be used in the corresponding strategy. In particular, primes are listed in the order that they are multiplied into the starting point by a strategy. For instance, if we had a small strategy using only 3s and 5s, and we had the sublist [5,3,3,5,3], this would mean that the first isogeny would be of degree 3, since we would multiply out 5, then 3, then 3, and then 5, leaving a point of degree 3. The next isogeny would be of degree 5, then degree 3, degree 3, and finally degree 5. In short, the order of the isogenies is the reverse of this list.

We then decomposed the strategy into a single list specifying how many point multiplications by small-degree primes are required to perform to generate a pivot point. Since the prime list is ordered, a strategy may dictate a mix of primes to multiply by to create a pivot point. In Fig. 1, a pivot point is created

Algorithm 1: Computing and evaluating a multiprime $\ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}$-isogeny with a strategy

function `multiprime_iso`

Static Parameters: Small prime numbers $\ell_1, \ell_2, \ldots \ell_n$ and Integers $e_1, e_2, \ldots e_n, e_S = (e_1 + e_2 + \ldots + e_n)$ from public parameters, a list describing an order of point multiplications by small prime $M = (m_1, \ldots, m_{e_S}) \in (\mathbb{N}^+)^{e_S}$, a *strategy* $(s_1, \ldots, s_{e_S-1}) \in (\mathbb{N}^+)^{e_S-1}$

Input: Curve E_0 and point S on E_0 with exact order $\ell_1^{e_1}\ell_2^{e_2}\cdots\ell_n^{e_n}$

Output: Curve $E = E_0/\langle S\rangle$

Initialize empty deque D
`push(`D, (e_S, S)`)`
$E \leftarrow E_0$, $i \leftarrow 1$, $h \leftarrow e_S$, $k = e_S$
while D *not empty* **do**
  $(h, R) \leftarrow$`pop(`D`)`
  if $h = 1$
    $(E', \phi) \leftarrow$ `compute_`ℓ`_iso(`E,R, $\ell = m_k$`)`
    Initialize empty deque D'
    while D *not empty* **do**
      $(h, R) \leftarrow$ `pull(`D`)`
      $R \leftarrow$ `evaluate_`ℓ`_iso(`E', ϕ, R, $\ell = m_k$`)`
      `push(`$D', (h-1, R)$`)`
    $D \leftarrow D', E \leftarrow E'$, $k \leftarrow k - 1$
  elif $0 < s_i < h$
    `push(`$D, (h, R)$`)`
    for $j \leftarrow e_S - h$ **to** $e_S - h + s_i$ **do**
      $R \leftarrow$ `mult_by_`ℓ`(`R, E, $\ell = m_j$`)`
    `push(`$D, (h - s_i, R)$`)`, $i \leftarrow i + 1$
  else
    Error: Invalid strategy
return $E = E_0/\langle S\rangle$

and stored whenever there is a vertical edge. These points are then pushed through isogeny evaluations after each isogeny computation. Upon reaching a leaf node, an isogeny would be computed as is specified in the prime list.

Algorithm 1 shows our algorithm for performing a multiprime large-degree isogeny. This is generalized from the SIKE submission's [2] algorithm for computing an ℓ^e isogeny. In terms of parameters, the primary difference here is that we have multiple small primes to use. Thus, we have one summation term, e_S to denote the total number of isogenies to perform. Then, the order of the isogenies matter, so this is represented by a list describing the order of the strategy. In total, there are e_S isogenies that need to be performed, so the point multiplica-

tion ordered list should have e_S entries, which can further be broken down into e_B entries of ℓ_B, e_C entries of ℓ_C, and so on.

In terms of the algorithm flow, we only include one more variable, k, which represents the next isogeny to perform. This acts as a reverse iterator of the list M, so after computing an ℓ-isogeny and evaluating each stored pivot point, we decrement the counter in line 13. Otherwise, we now have to identify which ℓ we are using at each isogeny operation or point multiplication. For isogeny computation and evaluation, this is variable k. For the point multiplication, we are still using the same strategy flow. By this, we mean that say $s_i = 5$. This means that we will perform 5 point multiplications starting at some index in the point multiplication list and iterating through the next 5 entries. We have represented this index by updating the for loop to start at $e_S - h$ as is shown in line 16. h represents the current number of point multiplications that have been applied to the current point, so we subtract this amount from the total number of isogenies to get the current index into the multiplication list.

Co-Implementing Multiprime and Split-Prime Strategies. As our results show in Table 5, the split prime approach is ideal for SIDH round 1 operations and the multiprime approach is ideal for SIDH round 2 operations. It is simple to use different large-degree isogeny algorithms for both rounds. The only caveat is that you must calculate a different representation of the private key for each round. For instance, if you start with two private keys to generate the two kernels as part of round 1, then you need to use the Chinese Remainder Theorem to combine these keys. If you start with a single large private key, then you must generate the two smaller keys by performing modulus operations, such as $n_B = n_{BC} \mod \ell_B^{e_B}$ and $n_C = n_{BC} \mod \ell_C^{e_C}$. These can be done efficiently using Montgomery or Barrett reduction [17, 3] (with precomputed values), but special care should be taken here to prevent side-channel attacks.

6 Conclusions

We applied the concept of multiprime large-degree isogeny strategies to the extended SIDH framework. We see that multiprime strategies can be used to accelerate Bob's large-degree isogeny by about 10% for the balanced eSIDH primes in the serial setting. We applied multiprime strategies to generalized prime forms for Bob, finding new primes that could further accelerate Bob's large-degree isogeny by a further 3%. The beauty of eSIDH is that the generalized form of prime allows for a variety of optimization targets, including parallelization and a prime of a target size. This work continues to push the envelope for performance gains when (e)SIDH parameters are chosen well.

References

[1] Reza Azarderakhsh et al. *Practical Supersingular Isogeny Group Key Agreement*. Cryptology ePrint Archive, Report 2019/330. 2019.

[2] Reza Azarderakhsh et al. *Supersingular Isogeny Key Encapsulation*. Tech. rep. Available online at `https://www.sike.org`. 2020.
[3] Paul Barrett. "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 311–323.
[4] Daniel Bernstein et al. "Faster computation of isogenies of large prime degree". In: *Open Book Series* 4 (Dec. 2020), pp. 39–55.
[5] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH*. Cryptology ePrint Archive, Paper 2022/975. 2022.
[6] Wouter Castryck et al. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven Galbraith. Springer International Publishing, 2018, pp. 395–427.
[7] Daniel Cervantes-Vázquez and Francisco Rodríguez-Henríquez. *A note on the cost of computing odd degree isogenies*. Cryptology ePrint Archive, Report 2019/1373. 2019.
[8] Daniel Cervantes-Vázquez et al. *eSIDH: the revenge of the SIDH*. Cryptology ePrint Archive, Report 2020/021. 2020.
[9] Craig Costello and Huseyin Hisil. "A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 303–329.
[10] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (), pp. 644–654.
[11] Luca De Feo et al. *SIDH Proof of Knowledge*. Cryptology ePrint Archive, Paper 2021/1023. 2021.
[12] Luca De Feo et al. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Journal of Mathematical Cryptology* 8.3 (2014), pp. 209–247.
[13] Satoshi Furukawa et al. "Multi-party Key Exchange Protocols from Supersingular Isogenies". In: *2018 International Symposium on Information Theory and Its Applications (ISITA)*. 2018, pp. 208–212.
[14] Wissam Ghantous et al. *Efficiency of SIDH-based signatures (yes, SIDH)*. Cryptology ePrint Archive, Paper 2023/433. 2023.
[15] Aaron Hutchinson et al. "Further Optimizations of CSIDH: A Systematic Approach to Efficient Strategies, Permutations, and Bound Vectors". In: *Applied Cryptography and Network Security*. Ed. by Mauro Conti et al. Cham: Springer International Publishing, 2020, pp. 481–501.
[16] Luciano Maino and Chloe Martindale. *An attack on SIDH with arbitrary starting curve*. Cryptology ePrint Archive, Paper 2022/1026. 2022.
[17] Peter L. Montgomery. "Modular multiplication without trial division". In: *Mathematics of Computation* 44.170 (1985), pp. 519–521.
[18] Damien Robert. *Breaking SIDH in polynomial time*. Cryptology ePrint Archive, Paper 2022/1038. 2022.
[19] Jacques Vélu. "Isogénies entre courbes elliptiques". In: *C. R. Acad. Sci. Paris Sér. A-B* 273 (1971), A238–A241.

A SIDH Protocol Description

Supersingular Isogeny Diffie-Hellman (SIDH) was introduced by De Feo, Jao, and Plût in 2011 [12]. Superficially the protocol resembles the classical Diffie-Hellman protocol [10], with the base group replaced by a set of elliptic curves, and the group operation replaced with isogeny codomain construction.

Setup: We require the following global parameters:

1. A prime $p = \ell_A^{e_A} \ell_B^{e_B} f \pm 1$ where ℓ_A and ℓ_B are prime, and $\ell_A^{e_A} \approx \ell_B^{e_B}$;
2. A supersingular elliptic curve $E/\mathbb{F}_{p^2}$; and,
3. Four points $P_A, P_B, Q_A, Q_B \in E(\mathbb{F}_{p^2})$ such that $E[\ell_A^{e_A}] = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}] = \langle P_B, Q_B \rangle$.

One party (Alice) will use the $\ell_A^{e_A}$-torsion subgroup, and the other (Bob) will use the $\ell_B^{e_B}$-torsion subgroup.

Key Generation: Alice:

1. Selects $\alpha \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ uniformly at random;
2. Constructs the isogeny $\psi_A \colon E \to E_A = E/\langle P_A + \alpha Q_A \rangle$; and,
3. Constructs the auxiliary points $S_A = \psi_A(P_B)$ and $T_A = \psi_A(Q_B)$.

Alice's private/public keypair is

$$\mathsf{sk}_A = \alpha \text{ and } \mathsf{pk}_A = (E_A, S_A, T_A).$$

Bob proceeds analogously.

Communication: The parties exchange their public keys.

Key Establishment: Alice computes

$$K_A = j\left(E_B/\langle S_B + \alpha T_B \rangle\right)$$

Bob proceeds analogously to find his key K_B. We have $K_A = K_B$.

The protocol is depicted in Figure 3.

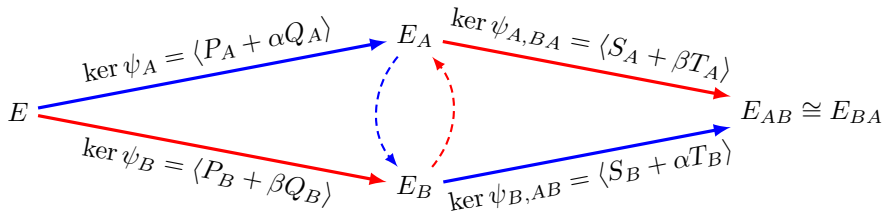

Fig. 3. The computations involved in SIDH. Alice follows the solid blue arrows by finding the codomain curve of the indicated isogeny, and follows the dashed blue arrow by reading the message she receives from Bob. Bob analogously follows the red arrows.

The underlying hard problem of SIDH is the following:

Problem 1 (Supersingular Decisional Diffie-Hellman Problem). Let $\phi_A \colon E \to E_A$ be an isogeny with kernel $\langle P_A + \alpha Q_A \rangle$ where α is chosen uniformly at random from $\mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$. Similarly, let $\phi_B \colon E \to E_B$ be an isogeny with kernel $\langle P_B + \beta Q_B \rangle$ where β is chosen uniformly at random from $\mathbb{Z}/\ell_B^{e_B}\mathbb{Z}$. Given a tuple

$$(E, E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_C)$$

where either $E_C = E_{AB} = E/\langle P_A + \alpha Q_A, P_B + \beta Q_B \rangle$ or E_C is sampled uniformly at random from the set of all curves of the form

$$E/\langle P_A + y_A Q_A, P_B + y_B Q_B \rangle$$

where y_A and y_B are chosen with the same conditions as α and β, respectively, each with probability $\frac{1}{2}$, the *supersingular decisional Diffie-Hellman problem (SSDDH)* is to determine which is the case.