B-SIDH: supersingular isogeny Diffie-Hellman using twisted torsion

Craig Costello

Microsoft Research, USA
craigco@microsoft.com

Abstract. This paper introduces a new way of instantiating supersingular isogeny-based cryptography in which parties can work in both the $(p+1)$-torsion of a set of supersingular curves and in the $(p-1)$-torsion corresponding to the set of their quadratic twists. Although the isomorphism between a given supersingular curve and its quadratic twist is not defined over $\mathbb{F}_{p^2}$ in general, restricting operations to the x-lines of both sets of twists allows all arithmetic to be carried out over $\mathbb{F}_{p^2}$ as usual. Furthermore, since supersingular twists always have the same $\mathbb{F}_{p^2}$-rational j-invariant, the SIDH protocol remains unchanged when Alice and Bob are free to work in both sets of twists.
This framework lifts the restrictions on the shapes of the underlying prime fields originally imposed by Jao and De Feo, and allows a range of new options for instantiating isogeny-based public key cryptography. This includes alternatives that exploit Mersenne, Solinas, and Montgomery-friendly primes, the possibility of halving the size of the primes of the Jao-De Feo construction at no known loss of asymptotic security, and more.

Keywords: Post-quantum cryptography, supersingular isogenies, SIDH, SIKE, Montgomery curves, quadratic twists, Pell's equation, Størmer's theorem.

1 Introduction

"His result is very pretty, and there are many applications of it."

– Louis J. Mordell on Størmer's theorem [12].

State-of-the-art instantiations of Jao and De Feo's supersingular isogeny Diffie-Hellman (SIDH) protocol [31] – including those in the actively secure SIKE variant [30] – fix two public values m and n such that $2^m \approx 3^n$ and $p = 2^m 3^n - 1$ is prime. Throughout the protocol, Alice and Bob perform secret walks in two different graphs that share precisely the same nodes: these nodes are the set of $\overline{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves in characteristic p, and there are close to $\lfloor p/12 \rfloor$ of them. Alice's secret walks take m steps in the 2-isogeny graph, i.e. the graph whose edges correspond to 2-isogenies, and Bob's secret walks take n steps in the 3-isogeny graph, i.e. the graph whose edges correspond to 3-isogenies. Both Alice and Bob begin at the same starting curve, E_0, and the public keys they output contain the respective image curves, E_A and E_B, that are the destination nodes of their secret walks. Upon receipt of Bob's public key, Alice computes a secret 2^m-isogeny from E_B to E_{AB}, and computes $j(E_{AB}) \in \mathbb{F}_{p^2}$ – the j-invariant representing the isomorphism class that is the shared secret. Analogously, Bob computes the 3^n-isogeny from E_A to his image curve E_{BA}, which is isomorphic to E_{AB}, so he arrives at the same shared value $j(E_{BA}) \in \mathbb{F}_{p^2}$ as Alice.

The best known attacks against SIDH try to either recover Alice's secret isogeny $\phi_A \colon E_0 \to E_A$, or Bob's secret isogeny $\phi_B \colon E_0 \to E_B$, and both of these problems are instances of the *supersingular isogeny problem*: given a finite field K and two supersingular elliptic curves E, E' defined over K such that $\#E = \#E'$, compute an isogeny $\phi \colon E \to E'$. In general, the best known classical algorithm for solving the supersingular isogeny problem is the Delfs-Galbraith algroithm [19], which requires $O(p^{1/2})$ isogeny operations to find a collision (of walks from E and E') in the graph of size $O(p)$. However, the special isogenies computed in SIDH above give rise to appreciably

easier instances of the supersingular isogeny problem; they are of a fixed, known degree close to $p^{1/2}$, and this allows for a classical meet-in-the-middle attack that, asymptotically, requires only $O(p^{1/4})$ isogeny operations [31, §5]. Roughly speaking, the difference between the difficulty of the isogeny problems that arise in SIDH and that of the general supersingular isogeny problem is due to the fact that Alice and Bob only take about half as many steps as the diameters of each of their graphs. Furthermore, these half-length walks fall short of the length needed to achieve uniform mixing in the supersingular isogeny graph [31, §5.1].

Jao and De Feo chose primes of the form $p = 2^m 3^n - 1$ and half-length walks so that Alice and Bob can both compute their isogenies using arithmetic in $\mathbb{F}_{p^2}$; they represent each isomorphism class by a supersingular elliptic curve $E/\mathbb{F}_{p^2}$ with group order $E(\mathbb{F}_{p^2}) = (p+1)^2 = (2^m 3^n)^2$, which facilitates a full $\mathbb{F}_{p^2}$-rational 2^m-torsion and full $\mathbb{F}_{p^2}$-rational 3^n-torsion. When all of the order 2^m and 3^n subgroups are $\mathbb{F}_{p^2}$-rational, so are the corresponding isogeny computations.

A first observation that sets the scene for this work is that there are actually two choices of supersingular curves over $\mathbb{F}_{p^2}$: those whose group orders are $(p+1)^2$, and those whose group orders are $(p-1)^2$. Moreover, this choice can be made at every node in the supersingular isogeny graph: although curves from these two sets are not isomorphic (or even isogenous!) to one another over $\mathbb{F}_{p^2}$, they do become isomorphic over $\mathbb{F}_{p^4}$, and therefore share the same j-invariant in $\mathbb{F}_{p^2}$ [49, Proposition III.1.4]. Indeed, these two sets of curves are the *quadratic twists* of one another.

The main point of this paper is to show that SIDH protocol does not have to restrict to working in one of the two sets of quadratic twists: it can stay in $\mathbb{F}_{p^2}$ while working in *both* the $(p+1)$-torsion *and* the $(p-1)$-torsion. This allows Alice and Bob to work in the torsion corresponding to opposite sets of quadratic twists with no change to the protocol. Optimised Montgomery arithmetic [39] in the SIDH setting only needs the x-coordinates of points [31] and the A coefficient of the curve [16], and as such is entirely *twist-agnostic*; in other words, the twisting morphism (which only alters y-coordinates and the B coefficient) leaves x-coordinates and A coefficients unchanged, so the lifting to $\mathbb{F}_{p^4}$ described above becomes a mere theoretical technicality that is not visible in cryptographic implementations (see Section 3).

The price to pay for working with both twists is that at least one of Alice or Bob must now perform walks comprised of steps in multiple ℓ-isogeny graphs, i.e. switching between multiple values of ℓ. This changes the underlying hardness assumption for one or both parties, but (as is discussed at length in Section 6) there is no known reason to believe that switching between many ℓ's makes the resulting SIDH problems any easier, so long as the number of destination nodes remains roughly the same size as the Jao-De Feo instantiation. On the other hand, the proposed approach gives at least one party the option of walking roughly as far as the diameter of the graph, heading toward the territory where the theory of expander graphs[1] provides concrete results about rapid mixing.

Allowing torsion from both sets of twists unlocks a number of new options and trade-offs for isogeny-based public key cryptography; many examples are given throughout this paper to illustrate these possibilities. At a high level, the options presented fall into two categories: the first is where Alice gets to computes significantly faster 2^m isogenies (than in existing SIDH/SIKE implementations) at the expense of a heavy slowdown on Bob's side; the second, and perhaps the more interesting, is the possibility of halving the sizes of the underlying fields and the SIDH/SIKE public keys at no known loss of security.

Accelerating Alice, burdening Bob. The Jao-De Feo construction allows Alice and Bob to achieve roughly the same performance in SIDH and SIKE; the cost of a 2^m-isogeny and 3^n-isogeny are similar when $2^m \approx 3^n$, and the benchmarks for optimized implementations of SIKE are very similar for key encapsulation and key decapsulation – see [30, Table 2.1]. However, there are two real-world scenarios where the performance of one side is much more crucial than the performance on the other. The first is the *client-server* scenario, where servers are oftentimes performing many orders of magnitude more runs of the protocol than any individual client(s). For example, Cloudflare recently announced [34] a large-scale experiment on the performance

[1] See [21, Proposition 2.1], [24, Theorem 1], or [28].

of post-quantum cryptography in the BoringSSL implementation of the TLS handshake, with SIKEp434 being one of their two chosen key encapsulation mechanisms. At the recent NIST post-quantum standardization workshop, Sullivan remarked that Google did not clear SIKE for their side of the experiment due to a "denial of service risk" [51]. While the rationale here seems far-fetched (it would be worrying if increasing the TLS handshake by a few milliseconds would really threaten the stability of Google's online services), it does showcase one example scenario where server-side performance is the priority. In the SIKE protocol, the decapsulator must compute both isogenies [30], so it is only the encapsulator that can be sped up using the instantiations below. In TLS 1.3, the client sends its public key in the initial flow, and as such the server becomes the encapsulator; thus, if SIKE were to be dropped in to TLS 1.3 using the examples below, servers would indeed be the ones seeing the Alice's benefits while clients would be paying the penalty of being Bob.

An example of the opposite scenario, i.e. when the priority becomes the client's performance, is in the arena of lightweight cryptography [55, §1]. Here there are many example scenarios where resource-constrained devices are communicating with a (relatively unconstrained) server, and affording the client(s) a speedup at the expense of the server would be a welcomed trade-off.

Example 1 (Mersenne-p521). As an illustrative example, let p be the Mersenne prime $p = 2^{521} - 1$, and consider the prime factorization of $p - 1$ as

$$\begin{aligned} p - 1 = 2 \cdot 3 \cdot 5^2 \cdot 11 \cdot 17 \cdot 31 \cdot 41 \cdot 53 \cdot 131 \cdot 157 \cdot 521 \cdot 1613 \cdot 2731 \cdot 8191 \cdot 42641 \cdot 51481 \cdot 61681 \\ \cdot 409891 \cdot 858001 \cdot 5746001 \cdot 7623851 \cdot 34110701 \cdot 308761441 \cdot 2400573761 \cdot 65427463921 \\ \cdot 108140989558681 \cdot 145295143558111 \cdot 173308343918874810521923841. \end{aligned}$$

Suppose this prime is chosen as the basis of an implementation geared to match the security of SIKEp503 from the SIKE submission [30], which has $p = 2^{250}3^{159} - 1$. Alice can work in the $(p + 1)$-torsion as usual, computing 2^{250}-isogenies as she would in SIKEp503. Bob's torsion must be coprime to Alice's, and thus he must work entirely in the $(p - 1)$-torsion. To achieve close to the same level of conjectured security (see Section 6), Bob must compute isogenies of degree

$$N = 3 \cdot 5^2 \cdot 11 \ldots 5746001 \cdot 7623851 \cdot 34110701.$$

Compared to the optimal strategy for computing 159 chained 3-isogenies in the SIKEp503 implementation, Bob's isogeny computation has now become *a lot* more expensive. Computing a degree-ℓ isogeny requires $O(\ell)$ field operations (see Section 2), so the complexity of Bob's degree-N isogeny will be dominated by the largest prime(s); the 34110701-isogeny alone will be more expensive than Alice's entire 2^{250}-isogeny. Although this example is about as bad as the situation can get for Bob in this paper, it is not prohibitively bad. The formulas for arbitrary degree Montgomery isogenies [14] allow for managable isogeny computations, even up to degrees this large. Furthermore, as is detailed in Section 3, Bob has a number of additional options at his disposal for handling large ℓ, including extensive precomputation, parallelizing the algorithm from [14], or avoiding the large primes altogether by instead opting to find more smaller order torsion points at the cost of moving to a higher extension field.

On the other side, Alice's situation has significantly improved due to her being able to take advantage of the Mersenne prime. While her speedup (somewhere between a factor 1.4x and a factor 2x, based solely on optimised underlying prime field arithmetic in both scenarios) is nowhere near enough to counteract Bob's slowdown for a single run of the protocol (at least one order of magnitude), this trade-off may still be preferred in the types of real-world scenarios mentioned above.

In this example, observe that the same underlying field could be used to scale between more than one of the proposed SIKE parameter sets. To match SIKEp434 – the smallest SIKE parameter set with $p = 2^{216}3^{137} - 1$ – Alice could compute a 2^{216}-isogeny as usual, and Bob's largest prime could now go down to $\ell = 7623851$. Or, to move up to match SIKEp610, Alice would compute her 2^{305}-isogeny and Bob's largest prime would now be $\ell = 2400573761$; this size of ℓ is well into the range of Bob's slowdown being a deal-breaker, so here the alternatives described at the end

of Section 3 would be imperative. On the other hand, observe that this up-and-down scaling of security over the same field would (1) be welcome news to aggressive software implementers who are aware that the bulk of the effort in highly-optimised, curve-based cryptographic software goes into hand-written assembly for the underlying field arithmetic, and (2) allow for easy parameter shifts to all the intermediate possible parameterizations as well – Alice's changes are trivial for implementers and Bob could wrap (tailored) routines for individual large ℓ-isogenies around the optimised routine for the smallest parameter set.

Lastly, it is worth pointing out that Alice has the flexibility to scale her own security for any given run of the protocol, independent of, and unbeknownst to, Bob. This may be of interest in scenarios where the security of a long-term static key is paramount, e.g. in certain applications of the SIKE protocol. The torsion basis $\{P_A, Q_A\}$, which is part of the public parameters (see Section 2), could be specified once-and-for-all such that $\langle P_A, Q_A \rangle = E[2^{521}]$. For any run of the protocol, Alice could choose to work in the 2^m-torsion by instead working with $\{P'_A, Q'_A\} = \{[2^{521-m}]P_A, [2^{521-m}]Q_A\}$, and choosing the length of her secret scalar(s) accordingly. On the other side, Bob does not need to know Alice's choice here; the public key obtained under his secret isogeny ϕ_B will contain the basis $\{\phi_B(P_A), \phi_B(Q_A)\}$ that generates the 2^{521}-torsion on $\phi_B(E)$, and again Alice can simply modify the basis to generate torsion of her chosen degree, taking $\{[2^{521-m}]\phi_B(P_A), [2^{521-m}]\phi_B(Q_A)\}$ instead. This comes at no additional cost to Bob and a very minor additional cost to Alice. The only additional security caveat to keep in cases like this is Petit's attack on unbalanced parameters [42], but current knowledge says this is not applicable if Alice keeps the bitlength of her isogeny degrees within a factor two of Bob's, or vice versa.

Bob on both sides. Another possibility explored in this paper is the ability for either (or both) parties to use isogenies from both the $(p+1)$-torsion *and* the $(p-1)$-torsion.

Example 2 (Goldilocks prime). Consider the prime $p = 2^{448} - 2^{224} - 1$ underlying Hamburg's Goldilocks curve [27]. Alice could exceed the security offered by SIKEp434 by computing 2^{224}-isogenies solely inside the $(p+1)$-torsion as before. If Bob was to take his kernel subgroups entirely from the $(p-1)$-torsion (as was the case for the Mersenne prime above), he would have to choose a combinations of the odd primes in the factorization

$$\begin{aligned} p - 1 = 2 \cdot 641 \cdot 18287 \cdot 196687 \cdot 1466449 \cdot 2916841 \cdot 6700417 \cdot 1469495262398780123809 \\ \cdot 167773885276849215533569 \cdot 596242599987116128415063 \\ \cdot 37414057161322375957408148834323969. \end{aligned}$$

To match the conjectured security of SIKEp434, Bob would need to compute an ℓ-isogeny with $\ell = 167773885276849215533569$, which is far beyond the practical ranges. However, in this case Bob can also cherry-pick primes that remain in the $(p+1)$-torsion, taking the odd primes in the factorization

$$\begin{aligned} p + 1 = 2^{224} \cdot 3 \cdot 5 \cdot 17 \cdot 29 \cdot 43 \cdot 113 \cdot 127 \cdot 257 \cdot 449 \cdot 2689 \cdot 5153 \cdot 65537 \cdot 15790321 \\ \cdot 183076097 \cdot 54410972897 \cdot 358429848460993. \end{aligned}$$

The largest ℓ-isogeny Bob would now need to compute to match SIKEp434 security is $\ell = 15790321$, which is less than that which was needed in the Mersenne prime setting above. In this example working on both sides is the only real option for Bob, but this does come with a caveat – see Section 3.

Hybrid-friendly fields. In [16] it was argued that SIDH is particularly amenable to being paired with classical ECDH, because one can benefit from choosing a non-supersingular elliptic curve with a strong ECDLP over the same prime field that underlies SIDH. Moreover, if these curves are chosen using the best known modern practices for ECDH (cf. [35]), then much of the Montgomery-style arithmetic needed within SIDH can be reused in the classical ECC computations, or vice

versa. However, defining a new, non-standard classical curve over an arithmetically suboptimal prime field with $p = 2^m 3^n - 1$ is not as attractive (on the ECC side) as the hybrids that would result from some of the primes explored in this paper. In particular, both of the primes above (and several of those explored in Section 4) already support deployed elliptic curves in the current or future NIST standards, so high-speed libraries for all of the ECDH (and thus the shared field and curve) arithmetic are already widespread.

Smaller primes. Let M and N be the coprime degrees of Alice and Bob's respective secret isogenies. In both the original Jao-De Feo proposal with $M = 2^m$ and $N = 3^n$, and in the two scenarios above where N was instead the product of multiple primes, M and N are both approximately $p^{1/2}$. This is the best case scenario if both M and N need to divide $p+1$, but it is rather wasteful once the $(p-1)$-torsion is brought into play. The new requirement is that M and N need to instead divide $p^2 - 1$, which (assuming they remain balanced in size) means they can now lie anywhere in the range of around $p^{1/2}$ to around p. In Section 4 a number of examples are given that place M and N somewhere in the middle of this range, i.e. where $p^{1/2} \ll M, N \ll p$. Based on the current literature, this paper conjectures that these examples achieve the same asymptotic security as the Jao-De Feo construction with primes that are significantly smaller.

The optimal case. In terms of the ratio of isogeny degrees to the prime size, the optimal case is when $M \approx N \approx p$. Here Alice can compute isogenies of degree dividing $M = p + 1$, and Bob can compute coprime[2] isogenies of degree dividing $N = p - 1$. From a constructive point of view, realising this optimal scenario in practice boils down to being able to find primes p for which $p-1$ and $p+1$ are *smooth* enough to allow efficient isogeny computations. At first glance this may seem like a specific and/or overly restrictive requirement, but finding instances of such primes is made easier thanks to a beautiful theorem of Størmer from the late 1800's [50]. For the purposes herein (see Theorem 1), it says that two numbers differing by 2 are B-smooth if and only if the number between them, x, is found as a solution (x, y) of the Pell equation $x^2 - Dy^2 = 1$, where D is squarefree, and both D and y are B-smooth. Moreover, Størmer showed that the number of such x is finite, and he gave a method to find all of them by solving a system of Pell equations. Thus, applying his method to find primes p for which $p - 1$ and $p + 1$ are B-smooth simplifies to further stipulating that any solution satisfying the above requirements must also have x as a prime.

As an illustrative example, when $B = 53$, the largest number x such that $x - 1$ and $x + 1$ are B-smooth is $x = 2907159732049$, for which

$$\begin{aligned} x - 1 &= 2^4 \cdot 3 \cdot 11^2 \cdot 13^2 \cdot 29 \cdot 41 \cdot 47 \cdot 53, \quad \text{and} \\ x + 1 &= 2 \cdot 5^2 \cdot 7^6 \cdot 19^2 \cdot 37^2. \end{aligned}$$

However, the largest such number that is prime is $p = 187753824257$, for which

$$\begin{aligned} p - 1 &= 2^{24} \cdot 19^2 \cdot 31, \quad \text{and} \\ p + 1 &= 2 \cdot 3 \cdot 7 \cdot 11^2 \cdot 13 \cdot 29 \cdot 43^2 \cdot 53. \end{aligned}$$

In this toy example, Alice could compute M-isogenies with $M = 2^{24} \cdot 19^2 \cdot 31$, and Bob could compute N-isogenies with $N = 3 \cdot 7 \cdot 11^2 \cdot 13 \cdot 29 \cdot 43^2 \cdot 53$.

Størmer's algorithm was improved by Lehmer [36], who showed that if there are t primes less than or equal to B, all of the solutions can be found by solving the $2^t - 1$ Pell equations that correspond to the $2^t - 1$ possibilities of D being squarefree and B-smooth. Thus, to exhaustively list the full set of solutions for a given B gets increasingly difficult as B grows large, and indeed finding attractive examples of cryptographic size becomes a challenging balancing act. It is well known (see Section 5) that solving the Pell equation generally becomes harder as D grows large,

[2]Plainly, 2 is the only common factor of $p + 1$ and $p - 1$, but once a factor of 2 is removed from the side that is not divisible by 4, the remaining factors are immediately coprime.

but fortunately imposing an upper bound on the size of the desired solutions (since they become the field characteristic) gives rise to an effective early abort strategy for the simple continued fraction (SFC) method. The problem then becomes one of searching over B-smooth values of D until the SCF terminates with a solution (x, y) to the corresponding Pell equation before x grows too large; furthermore, the desired solutions need to (i) be sieved such that y is B-smooth to satisfy the requirements of Størmer's theorem, and (ii) find x as both prime and of the right size for a target security level.

This search is discussed in detail in Section 5, together with a large-sized example that has so far been found with it. At present, this example is provided merely as a proof of concept in order to illustrate that such examples exist. The search has not yet been streamlined and is far from optimal, and as such, so is the example. In the coming months, as the search is optimised, more exhaustive, and resources are ramped up, it is expected that many better examples will supercede that in the current version. The quality of an example is judged solely by the smoothness bound; if a prime p of a given size is such that $p - 1$ and $p + 1$ are B-smooth, then another example will be deemed better if it is also around the same size but achieves a smoothness bound less than B; it is hopeless to try adding any (e.g. arithmetically favourable) properties of the prime p to the wishlist, beyond the size of p itself.

In terms of classical security, having both M and N around the same size as p brings the asymptotic complexity of the Delfs-Galbraith algorithm to be the same as the asymptotic complexity of the claw-finding algorithm; both run in $O(p^{1/2})$ time. However, Delfs-Galbraith does not have large storage requirements, so when the $O(p^{1/2})$ storage requirements for the claw-finding algorithm are factored into the analysis (as in [1]), Delfs-Galbraith becomes the superior algorithm for breaking the underlying isogeny problem. An analogous situation happens in the quantum security analysis; the $O(p^{1/4})$ Biasse-Jao-Sankar algorithm [7] for the general supersingular isogeny problem (which is essentially the quantum version of Delfs-Galbraith) becomes far superior to Tani's quantum claw-finding algorithm [53], and it also has low storage requirements. The full security analysis is in Section 6.

Naming. The instantiation proposed in this paper is dubbed B-SIDH[3] in order to distinguish it from the original Jao-De Feo SIDH instantiation, and to avoid muddying the waters in the case that future cryptanalysis weakens any variants described herein. Although Section 6 conjectures that switching between multiple ℓ-isogeny graphs during one secret isogeny computation does not decrease security in any known way, it may turn out that using torsion with many prime factors is a bad idea, or that decreasing p relative to the degrees of the secret isogenies is a bad idea. Of course, it may also turn out that the one (or both) of the converse statements is true, but in any case it must be emphasised that the instantiations proposed in this paper rely on *different security assumptions* than SIDH and SIKE – see Section 6.

Security of non-commutative vs. commutative schemes. At a high level, there are currently two main umbrellas of isogeny-based public-key cryptography under public scrutiny: those like SIDH [31] and SIKE [30] where the curves involved have non-commutative endomorphism rings, and those like CRS [18,48] and CSIDH [10] where the associated endomorphism rings are commutative. It is important to note that, while there are similarities between the instantiations herein and CSIDH (like the use of many different prime isogeny degrees in the same secret computation), this paper falls entirely under the non-commutative umbrella.

At present, the security landscape of these two areas of isogeny interest is very different. In terms of the known difficulty of the underlying problems, the non-commutatitve endomorphism ring is what seemingly makes SIDH and SIKE stubborn to quantum adversaries, who currently

[3]Pronounced "B-side", in reference to the analogy between the set of supersingular curves of cardinality $(p-1)^2$ and the less popular, sometimes forgotten 'flip-side' of a record. Additionally, unlike the original set of supersingular curves of cardinality $(p+1)^2$, writing a supersingular curve from the B-side in Montgomery form cannot be done without specifying a non-trivial value of the B coefficient– see Section 3.

have to resort to generic algorithms with exponential complexity. Conversely, Childs, Jao and Soukharev [13] showed that commutative endomorphism rings allow the application of the Kuperberg [33] and Regev [46] subexponential algorithms to CRS and CSIDH. Both areas of cryptanalysis are currently very active, and there have been a number of very recent papers analysing the concrete security of SIDH/SIKE and of CSIDH. In December 2018, Lange speculated that "At this moment I think actually CSIDH has a better chance than SIKE of surviving, but who knows" [4]; however, recent cryptanalytic efforts seem to paint the opposite picture. A line of papers cryptanalysing SIDH and SIKE [1,32,17] have shown that the concrete complexity of the best known attacks is significantly worse than the original asymptotic analyses suggest, and thus the size of the parameters in the SIKE submission were decreased in the Round 2 update [30]. On the other hand, the orthogonal line of works cryptanlysing CRS and CSIDH [6,8,5,41] mostly come to different conclusions. Peikert's most recent word on the topic concludes that the original CSIDH analysis was too optimistic, and that "the proposed CSIDH-512 parameters provide relatively little quantum security beyond what is given by the cost of quantumly evaluating the CSIDH group action itself", and that CSIDH-512 "falls well short of the claimed NIST security level 1"[41], so it is reasonable to predict that future instantiations of CSIDH will shift the parameters in the other direction.

There are two other important real-world security considerations that arise for which the commutative and non-commutative isogeny-based schemes come with different pros and cons. The first is active security, which is obtained out-of-the-box by the commutative CRS and CSIDH schemes. Validation of public keys is highly efficient in these settings, and this is the main strength of CSIDH; it allows the use of long-term static keys without any passive-to-active transformations, and thus serves as a drop-in replacement for existing (EC)DH key exchange. On the other hand, the non-commutativity of SIDH and SIKE means that auxiliary points are needed (see Section 2) in the public keys to make the protocol work, and these auxiliary points are what active adversaries can use to prey on long-term secrets [24]. This is why the original SIDH scheme needed to be transformed into SIKE, the actively secure variant [30]. The transformation of all of the proposed instantiations in this paper would be analogous.

In regards to the ease of implementing side-channel resistant implementations of isogeny-based schemes, the main loop for the isogeny computation in the non-commutative schemes follows a regular, uniform execution that does not depend on the secret key. Beyond the arithmetic in the underlying field, the bulk of the work in achieving *constant-time* software implementations of SIDH/SIKE is in protecting the initial scalar multiplication that chooses the secret subgroup (see Section 2); methods for doing this are well-known from the classical ECC literature, and a variant of the Montgomery ladder [39] – arguably the most simple scalar multiplication to implement in constant time – is indeed what is used in SIKE [30]. On the other hand, achieving efficient constant-time implementations of CRS and CSIDH is non-trivial (see [10, Remark 15]), and this remains an active area of investigation. It must be emphasised that, while the instantiations in this paper compute chains of isogenies of varying degrees, the chain itself is fixed once-and-for-all and does not depend on secret key material; thus, all issues concerning side-channel protection in this paper are also completely analogous to the SIDH and SIKE settings.

Performance vs. SIDH. There is no performance claims made in this paper, except in the scenarios where Alice's performance will clearly be improved thanks to a faster underlying prime, but where it should be reiterated that Bob will almost always suffer a collossal slowdown. The main performance takeaway of this paper is that the primes and the public keys in the optimal scenario of Section 5 are significantly smaller than the SIDH/SIKE counterparts. Moreover, the public keys in Section 5 will remain smaller even when compression techniques [2,15,60,40] are applied to the SIDH and SIKE public keys. If the ECC+SIDH/SIKE hybrid is used as in [16], these gaps will widen further.

Time will tell whether the much smaller primes afforded in Section 5 can be exploited enough to make up for the larger degree isogenies; the first order of business is to find examples where the smoothness bound is optimised for a given security level (see Section 5). Other issues, such

as trying to then find an optimal strategy for the isogeny computation in Algorithm 1 over a particular example, are left for future work.

2 Twist-agnostic SIDH

The parameter that governs the security of Jao and De Feo's supersingular isogeny Diffie-Hellman (SIDH) protocol is the large prime p. As soon as p is chosen, a set of roughly $\lfloor p/12 \rfloor$ elements is defined: these are the entire set of supersingular j-invariants over $\overline{\mathbb{F}}_p$, and they are the nodes on the graphs that Alice and Bob walk on during the protocol. Alice and Bob share this set of nodes, but their graphs have different edges that depend on the degrees of their individual secret isogenies. Following [31], for any prime $\ell \nmid p$, there are $\ell + 1$ isogenies (counting multiplicities, and up to isomorphism) of degree ℓ that eminate from a given supersingular isomorphism class. Moreover, Pizer [43,44] showed that this gives rise to a connected $(\ell+1)$-regular multigraph that satisfies the Ramanujan property and thus has optimal expansion properties (see Section 6).

The prime p also governs the efficiency of SIDH. Alice and Bob both compute prime power isogenies whose degrees are of the form ℓ^e, where the only restrictions are that their individual values of ℓ are coprime and that the size(s) of ℓ^e is large enough that recovering the secret isogeny walks is hard. In theory, Alice and Bob could choose any value of ℓ they like, but it is more efficient if the ℓ^e-torsion (which is where their secret kernels are chosen from) is defined over $\mathbb{F}_{p^2}$ – the same field where all of the supersingular j-invariants are defined. Observing that the smallest primes ℓ give rise to the most efficient ℓ^e-isogenies, Jao and De Feo construct the prime p to guarantee this rationality condition by setting $p = f \cdot 2^m 3^n - 1$ (allowing for a small *cofactor* f), and representing nodes in the graph by elliptic curves $E/\mathbb{F}_{p^2}$ with

$$E(\mathbb{F}_{p^2}) \;\cong\; \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}. \tag{1}$$

For any $r \in \mathbb{Z}$ with $r \mid p+1$, the entire r-torsion $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$ is then contained in $E(\mathbb{F}_{p^2})$. By choosing p as above, this ensures that the full 2^m-torsion $E[2^m] \cong \mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m}$, and the full 3^n-torsion $E[3^n] \cong \mathbb{Z}_{3^n} \times \mathbb{Z}_{3^n}$, are both $\mathbb{F}_{p^2}$-rational. Since every (seperable) isogeny $\phi\colon E \to E'$ of degree d is in one-to-one correspondence with a kernel subgroup of order d [49, Proposition III.4.12], and each such isogeny is computed using rational functions of the input curve and the given kernel subgroup [58], it follows that if both of these inputs are $\mathbb{F}_{p^2}$-rational, then so is the isogeny computation.

SIDH. With $p = f \cdot 2^m 3^n - 1$ as above, the SIDH protocol specifies the following public parameters: a starting supersingular curve $E_0/\mathbb{F}_{p^2}$, a basis $\{P_A, Q_A\}$ for $E[2^m] \cong \mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m}$, and a basis $\{P_B, Q_B\}$ for $E[3^n] \cong \mathbb{Z}_{3^n} \times \mathbb{Z}_{3^n}$. To generate her public key, Alice chooses two secret integers $(\alpha_A, \beta_A) \in \mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m}$ such that her secret point $S_A = [\alpha_A]P_A + [\beta_A]Q_A$ is of order 2^m. She then computes m chained 2-isogenies that are composed to give her secret 2^m-isogeny $\phi_A\colon E_0 \to E_A$, where $E_A = E_0/\langle S_A \rangle$. Along the way, she moves the basis points P_B and Q_B through the isogeny computation, eventually obtaining their images under ϕ_A. Her public key is then $\mathrm{PK}_A = (\, E_A\,, \phi_A(P_B)\,, \phi_A(Q_B))$. On Bob's side, he chooses $(\alpha_B, \beta_B) \in \mathbb{Z}_{3^n} \times \mathbb{Z}_{3^n}$, computes his secret point $S_B = [\alpha_B]P_B + [\beta_B]Q_B$, and then uses it to compute his secret 3^n-isogeny $\phi_B\colon E_0 \to E_B$ (via n consecutive 3-isogenies), such that $E_B = E_0/\langle S_B \rangle$. His public key is $\mathrm{PK}_B = (\, E_B\,, \phi_B(P_A)\,, \phi_B(Q_A))$.

Upon receiving PK_B, Alice uses her secret integers to compute a new secret point $S'_A = [\alpha_A]\phi_B(P_A) + [\beta_A]\phi_B(Q_A)$ of order 2^m on E_B, and then uses it to compute the 2^m-isogeny $\phi'_A\colon E_B \to E_B/\langle S'_A \rangle$. Bob uses his secret integers and PK_A to compute the point $S'_B = [\alpha_B]\phi_A(P_B) + [\beta_A]\phi_A(Q_B)$ of order 3^n on E_A, and then uses it to compute the 3^n-isogeny $\phi'_B\colon E_A \to E_A/\langle S'_B \rangle$. Both parties then compute the same shared secret as the j-invariant of their respective image curves $E_B/\langle S'_A \rangle$ and $E_A/\langle S'_B \rangle$, since $E_B/\langle S'_A \rangle \cong E_A/\langle S'_B \rangle$ [31].

Twist-agnostic 2- and 3-isogenies in $\mathbb{P}^1$. Jao and De Feo exploited the fact that all of the arithmetic in the above computations can be performed on the x-line of the associated curves, i.e. in $E/\{\pm 1\}$ rather than E, and furthermore that this arithmetic is particularly efficient if the curves are in Montgomery form [39]

$$E_{A,B}\colon By^2 = x^3 + Ax^2 + x.$$

Henceforth, $E_{A,B}$ or E will be used instead of $E_{A,B}/\{\pm 1\}$ or $E/\{\pm 1\}$ for simplicity; unless explicitly stated, y-coordinates will be ignored (using a '—') since all computations will ultimately take place on x-lines.

Beyond the three point ladder (see [31]) that is used to compute the secret subgroup generators, Alice's 2^m-isogenies can be computed using only Montgomery doubling operations [39] and Montgomery 2-isogeny operations [47]. As such, the only operations she needs in the main loop are

$$\begin{aligned}
[2]\colon\quad & E_{A,B} \to E_{A,B},\\
& (X\colon —\colon Z) \mapsto \left((X^2 - Z^2)^2\colon —\colon 4XZ(X^2 + AXZ + Z^2)\right), \qquad \text{and}\\
\phi\colon\quad & E_{A,B} \to E_{A',B'},\\
& (X\colon —\colon Z) \mapsto \left(X(X_2X - Z_2Z)\colon —\colon Z(Z_2X - X_2Z)\right), \qquad \text{and}
\end{aligned}$$

$$(A'\colon 1) = (2Z_2^2 - 4X_2^2\colon Z_2^2), \tag{2}$$

where $(X_2\colon —\colon Z_2) \neq (0\colon —\colon 1)$ is a point of order 2 on $E_{A,B}$, where $E_{A',B'} = E_{A,B}/\langle (X_2\colon —\colon Z_2)\rangle$, and where $(B\colon 1) = (BX_2\colon Z_2)$ in $\mathbb{P}^1$.

Similarly, the only functions Bob needs in the main loop are Montgomery tripling operations and Montgomery 3-isogeny operations [21], given as

$$\begin{aligned}
[3]\colon\quad & E_{A,B} \to E_{A,B},\\
& (X\colon —\colon Z) \mapsto \big(X(4(A-2)XZ^3 - (X-3Z)(X+Z)^3)^2\colon —\colon\\
& \qquad\qquad Z(4(A-2)X^3Z + (3X-Z)(X+Z)^3)^2\big), \qquad \text{and}\\
\phi\colon\quad & E_{A,B} \to E_{A',B'},\\
& (X\colon —\colon Z) \mapsto \left(X(X_3X - Z_3Z)^2\colon —\colon Z(Z_3X - X_3Z)^2\right),
\end{aligned} \tag{3}$$

where $(X_3\colon —\colon Z_3)$ is a point of order 3 on $E_{A,B}$, and where $E_{A',B'} = E_{A,B}/\langle (X_3\colon —\colon Z_3)\rangle$ with $(A'\colon 1) = (X_3(AX_3Z_3 + 6(Z_3^2 - X_3^2))\colon Z_3^3)$ and $(B\colon 1) = (BX_3^2\colon Z_3^2)$ in $\mathbb{P}^1$.

In the same way that Jao and De Feo observed that the y-coordinate can be ignored when computing the images of points under isogenies, Costello, Longa and Naehrig [16] further observed that the Montgomery B coefficient can also be ignored when computing the image curves. It is not used in (2) or (3), so the above formulas work correctly on both twists.

General ℓ-degree isogenies. Costello and Hisil [14] generalised the 3-isogeny formulas of De Feo, Jao and Plût [21] to give formulas for arbitrary degree ℓ-isogenies in Montgomery form, given as

$$\begin{aligned}
\phi\colon\quad & E_{A,B} \to E_{A',B'},\\
& (X\colon —\colon Z) \mapsto \left(X\left(\prod_{i=1}^{d} X_iX - Z_iZ\right)^2\colon —\colon Z\left(\prod_{i=1}^{d} Z_iX - X_iZ\right)^2\right),
\end{aligned} \tag{4}$$

where $(X_i\colon —\colon Z_i)$ is the i-th multiple of the point $P \in E_{A,B}$ of exact order $\ell = 2d+1$, and where $E_{A',B'} = E_{A,B}/\langle P\rangle$ (the computation of the image curve $E_{A',B'}$ will be discussed in a moment).

Unlike the 2- and 3-isogenies in (2) or (3), for which there is only one x-coordinate found in the set of kernel points, the formulas in (4) require the coordinates of more than one kernel point in

general; for $d > 2$, the d projective tuples of the form $(X_i : — : Z_i)$ can be computed by using the Montgomery ladder [39] at a cost of $2(d-1)$ field squarings and $4(d-1)$ field multiplications [5, §3.4]. This is a one-off cost that is independent of how many points are to be pushed through ϕ. Each such point incurs an additional $(4d+2)$ multiplications, $2d+2$ additions and 2 squarings [14, Algorithm 3].

In [14, §4] it was pointed out that the formulas in (4) could also be used to compute the image curve $E_{A',B'}$: under ϕ, a 2-torsion point $(X_2 : — : Z_2) \neq (0 : — : 1)$ on $E_{A,B}$ will map to a 2-torsion point $(X_2' : — : Z_2') \neq (0 : — : 1)$ on $E_{A',B'}$, from which A' can be recovered via $(A' : 1) = (X_2^2 + Z_2^2 : — : X_2 Z_2)$. Meyer and Reith [38] showed that using the birational equivalence to the twisted Edwards form allows for a significantly faster computation of the isogenous curve; the isogenous Montgomery curve coefficient can be projectively computed as

$$(A' : 2) = \big(\ (A+2)^\ell \pi_+^8 + (A-2)^\ell \pi_-^8 \quad : \quad (A+2)^\ell \pi_+^8 - (A-2)^\ell \pi_-^8 \ \big), \qquad \text{where}$$
$$\pi_+ = \prod_{i=1}^{d} (X_i + Z_i) \qquad \text{and} \qquad \pi_- = \prod_{i=1}^{d} (X_i - Z_i), \tag{5}$$

where the X_i and Z_i (and their sums and differences) are already computed during the optimised computation of (4). Thus, computing π_+^8 and π_-^8 incurs an additional $(2d-2)$ field multiplications and 6 field squarings, and computing $(A+2)^\ell$ and $(A-2)^\ell$ incurs an additional $2 \cdot \log_2(\ell)$ field squarings and around $\log_2(\ell)$ field multiplications.

3 Using torsion from the quadratic twists

Let $E/\mathbb{F}_{p^n}$ be an elliptic curve, let t_n be the trace of the p^n-power Frobenius endomorphism, and recall that (i) E is supersingular if and only if t_n is a multiple of p [49, Exercise V.5.10(a)], and that (ii) $\#E(\mathbb{F}_{p^n}) = p^n + 1 - t_n$ with $|t_n| \leq 2\sqrt{p^n}$ [49, Theorem V.1.1]. When $n = 1$, there is only one possible value of t_1 that is a multiple of p such that $|t_1| \leq 2\sqrt{p}$, i.e. $t_1 = 0$, and thus it follows that $E/\mathbb{F}_p$ is supersingular if and only if $\#E(\mathbb{F}_p) = p+1$. In other words, there is only one possible group order for supersingular elliptic curves over $\mathbb{F}_p$.

The first observation that sets the scene for this work is that there are actually two possible values for t_2 that correspond to $E/\mathbb{F}_{p^2}$ being supersingular: $t_2 = -2p$ and $t_2 = 2p$ both satisfy (i) and (ii). All known instantiations of SIDH and SIKE fall into the former case by default. They define a starting supersingular curve $E_0/\mathbb{F}_p$ and lift to work in $E_0(\mathbb{F}_{p^2})$; since $E_0(\mathbb{F}_p) \mid E_0(\mathbb{F}_{p^2})$ and $\#E_0(\mathbb{F}_p) = p+1$, it must be that $\#E_0(\mathbb{F}_{p^2}) = p^2+1+2p = (p+1)^2$ and hence that $t_2 = -2p$.

Upon starting on a curve with $t_2 = -2p$, a choice has been made between the two possibilities for t_2; two elliptic curves are $\mathbb{F}_{p^2}$-isogenous if and only if they have the same group order over $\mathbb{F}_{p^2}$ [54, Theorem 1(c)], so computing $\mathbb{F}_{p^2}$-rational isogeny walks means walking on curves with the same number of points as $E_0/\mathbb{F}_{p^2}$. However, these two possibilities for t_2 correspond to sets of curves that are the *quadratic twists* of one another, meaning that they not only become isogenous over $\mathbb{F}_{p^4}$, they become isomorphic over $\mathbb{F}_{p^4}$. Moreover, optimised isogeny arithmetic works correctly independently of the quadratic twist (see Section 2), so the explicit formulas that are used on the curves with $t_2 = -2p$ can also be used to work on the curves with $t_2 = 2p$.

It is crucial to note that even though two quadratic twists are not isomorphic over $\mathbb{F}_{p^2}$, they will still have the same j-invariant in $\mathbb{F}_{p^2}$ [49, Proposition 1.4(b)]. Put another way, every node (i.e. j-invariant) in the supersingular isogeny graph can actually be represented by two different $\mathbb{F}_{p^2}$-isomorphism classes: those with $t_2 = -2p$ and the same group structure as $E/\mathbb{F}_{p^2}$ in (1), or those with $t_2 = 2p$ and with group structure

$$E^t(\mathbb{F}_{p^2}) \ \cong \ \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}.$$

Every such supersingular curve with group structure $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$ is the *quadratic twist* of a supersingular curve with group structure $\mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$, and vice versa. Moreover, in the same way that any factor r of $p+1$ gave rise to a full rational r-torsion in $E(\mathbb{F}_{p^2})$, any factor s of $p-1$ gives rise to a full rational s-torsion in $E^t(\mathbb{F}_{p^2})$.

For Alice and Bob to freely work with points coming from the $(p+1)$-torsion and the $(p-1)$-torsion, it appears that the entire protocol must be lifted to $\mathbb{F}_{p^4}$. While this is technically true, the lifting will ultimately not be visible in an optimised implementation[4]. As was discussed in Section 2, the point and isogeny-formulas ignore the y-coordinates of points and the B coefficients of Montgomery curves, and this is where all the twisting arithmetic happens. The upshot is that while the protocol will be lifted to $\mathbb{F}_{p^4}$, where $E(\mathbb{F}_{p^4}) \cong E^t(\mathbb{F}_{p^4}) \cong \mathbb{Z}_{p^2-1} \times \mathbb{Z}_{p^2-1}$, Alice and Bob are still in a position to work entirely in $\mathbb{F}_{p^2}$ as usual. They can then choose a secret kernel point whose order divides $p+1$, or whose order divides $p-1$, or (more generally) whose order divides the product p^2-1.

To make this concrete, let B be a square in $\mathbb{F}_{p^2}$, let γ be a non-square in $\mathbb{F}_{p^2}$, take $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}(\delta)$ with $\delta^2 = \gamma$, and write

$$E_{A,B}\colon By^2 = x^3 + Ax^2 + x \qquad \text{and} \qquad E^t_{A,\gamma B}\colon \gamma By^2 = x^3 + Ax^2 + x$$

as models[5] for $E/\mathbb{F}_{p^2}$ and $E^t/\mathbb{F}_{p^2}$. The map

$$\sigma\colon E_{A,\gamma B}(\mathbb{F}_{p^4}) \to E_{A,B}(\mathbb{F}_{p^4}), \qquad (x,y) \mapsto (x, \delta y) \tag{6}$$

is a group isomorphism that leaves x-coordinates unchanged.

Write $f(x) = x^3 + Ax^2 + x$. For any $u \in \mathbb{F}_{p^2}$, either (1) $f(u)$ is a square in $\mathbb{F}_{p^2}$, in which case $(u, \sqrt{f(u)/B})$ is a point in $E_{A,B}(\mathbb{F}_{p^2})$, (2) $f(u)$ is a non-square in $\mathbb{F}_{p^2}$, in which case $f(u)/(\gamma B)$ is a square, and $(u, \sqrt{f(u)/(\gamma B)})$ is a point in $E_{A,\gamma B}(\mathbb{F}_{p^2})$, or (3) $f(u) = 0$, in which case $(u, 0)$ is one of the three 2-torsion points (on both $E_{A,B}$ and $E_{A,uB}$).

Let $P_1 = (u_1, —)$ be a point corresponding to case (1), let $P_2 = (u_2, —)$ be a point corresponding to case (2), and suppose $\phi_1\colon E_{A,B} \to E_{A,B}/\langle P_1\rangle$ and $\phi_2\colon E_{A,\gamma B} \to E_{A,\gamma B}/\langle P_2\rangle$. It does not make sense to evaluate ϕ_1 at P_2 or ϕ_2 at P_1 (these points do not even lie on $\mathbb{F}_{p^2}$-isogenous curves, let alone the same curve), but this is fixed by lifting to $\mathbb{F}_{p^4}$ and precomposing with the twisting morphisms. Setting $\phi_1' = \phi_1 \circ \sigma$ and $\phi_2' = \phi_2 \circ \sigma^{-1}$ gives the isogenies

$$\phi_1'\colon \quad E_{A,\gamma B} \to E_{A,\gamma B}/\langle \sigma(P_2)\rangle, \qquad P \mapsto (\phi_1 \circ \sigma)(P)$$

and

$$\phi_2'\colon \quad E_{A,B} \to E_{A,B}/\langle \sigma^{-1}(P_1)\rangle, \qquad P \mapsto (\phi_2 \circ \sigma^{-1})(P),$$

which are well-defined over $\mathbb{F}_{p^4}$.

The key observation is that $\sigma\colon (x, —) \mapsto (x, —)$ and $\sigma^{-1}\colon (x, —) \mapsto (x, —)$ behave like the identity map when working on the corresponding Kummer lines, so the twisting morphisms can simply be ignored in the implementation. Thus, if both parties operate within the x-coordinate- and A-coefficient-only framework, Alice can take her secret points from the $(p+1)$-torsion of $E_{A,B}(\mathbb{F}_{p^2})$ and Bob can take his secret points from the $(p-1)$-torsion of $E_{A,\gamma B}$, and the implementation of the SIDH protocol can otherwise remain unchanged.

Computing $(\prod \ell_i^{e_i})$-isogenies. Unlike the original Jao-De Feo construction, where Alice computes 2^m-isogenies and Bob computes 3^n-isogenies, all of the instantiations proposed herein require that at least one of the two parties computes isogenies whose degree has multiple prime factors. The general setting is that Alice computes M-isogenies and Bob computes N-isogenies, where $M = \prod_{i=1}^{a} m_i^{\mu_i}$ and $N = \prod_{i=1}^{b} n_i^{\nu_i}$ are coprime, with m_i prime for all $1 \le i \le a$, and with n_i

[4]This is reminiscent of Bernstein's twist-agnostic Curve25519 construction. He also uses a quadratic extension field in the specification of the Curve25519 function [3, Theorem 2.1], but this extension is a technicality that is not seen in the implementation.

[5]The idea works analogously for more general (i.e. short Weierstrass) elliptic curves, but all of the instantiations discussed in this paper allow for Montgomery form.

prime for all $1 \leq i \leq b$. In what follows, both of these will be treated as instances of the general situation in which a party is required to compute an isogeny of degree

$$L = \prod_{i=1}^{k} \ell_i^{e_i},$$

where the ℓ_i are all distinct primes.

For now, assume that either $L \mid p+1$ so that $E[L]$ is $\mathbb{F}_{p^2}$-rational, or that $L \mid (p-1)$ so that $E^t[L]$ is $\mathbb{F}_{p^2}$-rational (the more general situation where $L \mid p^2-1$ will be discussed in the sequel). The number of cyclic subgroups of order ℓ^e in $E[\ell^e]$ is $(\ell+1)\ell^{e-1}$ [21, §3], from which it follows that the number of cyclic subgroups of order L in $E[L]$ is

$$\prod_{i=1}^{k} (\ell_i + 1)\ell_i^{e_i - 1}. \tag{7}$$

Let P and Q be points of order L such that the Weil pairing $e_L(P,Q)$ is of exact order L, i.e. such that $\{P,Q\}$ is a basis for the full L-torsion. To compute secret subgroups, the same simplification will be made herein as is made in the state-of-the-art SIDH and SIKE implementations; rather than computing the secret generators as $S = [\alpha]P + [\beta]Q$ (for combinations of $(\alpha,\beta) \in \mathbb{Z}_L \times \mathbb{Z}_L$ that give rise to S being of order L), it is preferable to take

$$S = Q + [\alpha]P \qquad \text{for} \qquad \alpha \in \{0, 1, 2, \ldots, L-1\}, \tag{8}$$

since this guarantees that S is of exact order L, that each α corresponds to a unique kernel subgroup, and it allows S to be computed using the simple three point ladder [21].

Jao and De Feo gave two simple algorithms for computing ℓ^e-isogenies: 'multiplication based' and 'isogeny based'. In their construction, when ℓ is fixed as 2 or 3, ℓ-isogenies and the multiplication-by-ℓ map are only a handful of field operations each, so both of these algorithms have roughly the same complexity. Opting for one over the other will depend on the specific value of ℓ, the explicit formulas used, and the implementation [31, §4.2]. For general ℓ, however, ℓ-isogenies require $O(\ell)$ field operations (see Section 2), while the multiplication-by-ℓ map only requires $O(\log \ell)$ field operations, meaning that the 'multiplication based' approach becomes much more efficient.

Algorithm 1 adopts the multiplication based approach to compute cyclic isogenies of degree $L = \ell_1^{e_1} \ldots \ell_k^{e_k}$. Since Alice and Bob must push each other's basis points through their secret isogenies during public key generation, it also takes as input a list of points $T_1, \ldots, T_z$, and outputs their image under the isogeny.

Algorithm 1 Computing a cyclic $(\ell_1^{e_1} \ldots \ell_k^{e_k})$-isogeny.

Input: Elliptic curve E, a point $S \in E$ of order $L = \ell_1^{e_1} \ldots \ell_k^{e_k}$, and (optionally) a list of points $T_1, \ldots, T_z \in E$ to be pushed through the isogeny.
Output: $E/\langle S \rangle = \phi(E)$ and $\phi(T_1), \ldots, \phi(T_z) \in \phi(E)$.

1: Initialise $E' \leftarrow E, \quad S' \leftarrow S, \quad (T_1', \ldots, T_z') \leftarrow (T_1, \ldots, T_z)$
2: **for** $i = 1$ **to** k **do**
3: **for** $e = 0$ **to** $e_i - 1$ **do**
4: Compute $R = [\ell_i^{e_i - e - 1}]S'$
5: Compute $\psi : E' \to E'/\langle R \rangle$
6: Evaluate $(E', S', (T_1', \ldots, T_z')) \leftarrow (\psi(E'), \psi(S'), (\psi(T_1'), \ldots, \psi(T_z')))$
7: **end for**
8: **end for**
9: **return** E', $(T_1', \ldots, T_z')$

Algorithm 1 works inside the x-coordinate-only Montgomery framework as follows. In the examples that arise where Alice or Bob has $k > 1$ and values of $\ell_i > 3$, Step 4 of Algorithm 1

amounts to calling the Montgomery ladder with the scalar ℓ_i, and Steps 5 and 6 are computed using both (4) and (5). Otherwise, if $\ell_i \in \{2, 3\}$, then the explicit formulas for multiplication-by-ℓ_i and the ℓ_i-isogenies are given in (2) and (3).

For all of the scenarios proposed herein, Algorithm 1 is immediately amenable to *constant-time* implementations. For a fixed set of system parameters, it will compute the same sequence of isogenies for a given party that is independent of their secret. It is during the computation of the secret (input) point, S, that care needs to be taken to resist simple side-channel attacks.

For prime power isogenies of the form ℓ^e, De Feo, Jao and Plût [21, §4.2.2] give *optimal strategies* that perform significantly better than the multiplication- or isogeny-based approaches. If a given example has $e_i \gg 1$ for some i, then their treatment can be directly applied to replace Steps 3 to 7 in Algorithm 1 and call an optimised subroutine for the $\ell_i^{e_i}$-isogeny. However, extending their approach to compute the full cyclic $(\ell_1^{e_1} \dots \ell_k^{e_k})$-isogeny is non-trivial. Every ℓ_i will produce a different ratio between the cost of the multiplication-by-ℓ_i map and the cost of evaluating ℓ_i-isogenies, so there are $2k$ costs (rather than just 2) that must be factored into the analysis. For examples where k is large and $e_i = 1$ for most $i \in \{1, \dots k\}$, implementers will likely find benefits by deriving ad-hoc strategies that aim to pseudo-optimise according to the $2k$ costs that are specific to the example and to the implementation itself. In any case, implementers applying Algorithm 1 as is should be made aware that it is suboptimal in general.

On a related note, it is worth noting that Algorithm 1 permits any (re)ordering of the ℓ_i. For example, if extensive precomputations are employed to deal with large values of ℓ_i (more on this below), then implementers will choose for these isogenies to come first. Conversely, if a more optimised strategy requires many temporary points to be moved through intermediate isogenies (as is the case in [21]), then it may make sense to arrange some of the larger primes to be at the end of the computation, where these points tend to be dropped.

Bob on both sides: a caveat. In scenarios like Example 2, where Bob working entirely in the $(p-1)$-torsion would make for unreasonably large ℓ-isogenies, Bob can instead build his N-isogeny by taking any product $N \mid p^2 - 1$ where N is coprime to Alice's M (the most common scenario being $M = 2^m$ so that any odd N suffices). However, in this case points of exact order N will no longer have their x-coordinates in $\mathbb{F}_{p^2}$ in general; this only happens when the points are of order dividing $p-1$, or they are of order dividing $p+1$. Bob has two options, both of which add overhead to his and Alice's computations, and both of which inflate the size of Alice's public keys.

The first option involves defining the system parameters using two points $P_B, Q_B \in E(\mathbb{F}_{p^4})$ such that $\langle P_B, Q_B \rangle = E(\mathbb{F}_{p^4})[N]$, and then proceeding as usual. Bob's elliptic curve and isogeny arithmetic will now take place over $\mathbb{F}_{p^4}$, and he will incur a significant performance penalty. However, the underlying curves are always minimally defined over $\mathbb{F}_{p^2}$ [20], and Alice's basis points will remain in $\mathbb{F}_{p^2}$ even as they are dragged through computations involving elements in $\mathbb{F}_{p^4}$. Thus, his public keys will stay the same size (the three elements of $\mathbb{F}_{p^2}$ that are the x-coordinates of $\phi_B(P_A)$, $\phi_B(Q_A)$, and $\phi_B(Q_A - P_A)$), while Alice's will double (they are now three x-coordinates in $\mathbb{F}_{p^4}$).

The second option is to split P_B and Q_B into sums of two points whose x-coordinates are in $\mathbb{F}_{p^2}$. If $N' \mid p+1$ and $N'' \mid p-1$ are such that $N' \cdot N'' = N$ with N' and N'' coprime, then it is straightforward to find the points P'_B, P''_B, Q'_B and Q''_B such that $P_B = P'_B + P''_B$ and $Q_B = Q'_B + Q''_B$, with $E(\mathbb{F}_{p^2})[N'] = \langle P'_B, Q'_B \rangle$ and $E^t(\mathbb{F}_{p^2})[N''] = \langle P''_B, Q''_B \rangle$. Rather than taking secrets $\alpha \mathbb{Z}_N$, Bob could now define his secrets as $(\alpha', \alpha'') \in \mathbb{Z}_{N'} \times \mathbb{Z}_{N''}$, and split his secret isogeny computations into two components: the first would take $S'_B = P'_B + [\alpha']Q'_B$, compute $\phi' \colon E \to E'$ with $E' = E/\langle S'_B \rangle$, while also computing $\phi'(P_A)$, $\phi'(Q_A)$, $\phi'(P''_B)$ and $\phi'(Q''_B)$; the second would compute $S''_B = \phi'(P''_B) + [\alpha'']\phi'(Q''_B)$, then $\phi'' \colon E' \to E''$ with $E'' = E'/\langle S''_B \rangle$, together with $\phi''(\phi'(P_A))$ and $\phi''(\phi'(Q_A))$. Setting $\phi = \phi'' \cdot \phi'$, Bob can then output his public key as $\phi(E)$, $\phi(P_A)$ and $\phi(Q_A)$. The only additional overhead Bob incurs in this case is having to push P''_B and Q''_B through the first isogeny ϕ', but all of his arithmetic is now in $\mathbb{F}_{p^2}$. On Alice's side, she would need to push the four $\mathbb{F}_{p^2}$-rational points through her secret isogeny, but this is again likely to be

preferred over the $\mathbb{F}_{p^4}$ arithmetic in the first option above. Her public key would again double in size.

Even if the system parameters are defined as in the first option, Bob can do the splitting in the second option privately to speed up his computations and keep the arithmetic in $\mathbb{F}_{p^2}$. Note that this caveat only applies to some of the examples in Section 4, but not to any instances of the *optimal scenario* in Section 5.

Handling large ℓ-degree isogenies Recall from above that, unlike the multiplication-by-ℓ map which can be computed in $\log(\ell)$ field operations, the computation of a prime degree cyclic ℓ-isogeny requires $O(\ell)$ field operations. If $\ell_1 < \cdots < \ell_k$, this can make an ℓ_k-isogeny the overwhelming bottleneck of a full $(\ell_1^{e_1} \dots \ell_k^{e_k})$-isogeny routine. The purpose of this subsection is to describe some ways that ℓ-isogenies can either be (i) accelerated in practice, or (ii) traded off for smaller degree isogenies that are defined in extension fields.

Parallelisation. Let P be a point of order $\ell = 2d+1$, write $[i]P = (X_i : — : Z_i)$, and recall from (4) and (5) that the ℓ-isogeny with kernel $\langle P \rangle$ requires $(X_i : — : Z_i)$ for $1 \leq i \leq d$. Computing *all* d multiples of the input point is what makes ℓ-isogeny computations become rapidly expensive. However, this process parallelises almost perfectly: for large ℓ, t processors working in parallel can reduce the number of (differential) additions required by close to a factor of t. A straightforward way to achieve this starts by using $\lceil t/2 \rceil$ steps of the Montgomery ladder to compute $[i]P$ for $1 \leq i \leq t$. Three values are then passed to the first $t-1$ processors to kickstart the differential additions: the i-th processor is passed $[i]P$, $[t]P$ and $[t-i]P$, for $1 \leq i \leq t-1$. The last processor only needs $[t]P$. The i-th processor can then compute $[i+jt]P$ as the differential sum of $[i+(j-1)t]P$, $[t]P$, and $[i+(j-2)t]P$ for $1 \leq j \leq \lceil d/t \rceil$ (the only exception is the t-th processor which kickstarts with a doubling to compute $[2t]P$ from $[t]P$, but thereafter carries on with differential additions in unison). Each time a new value $(X_{i+jt} : — : Z_{i+jt})$ is computed, it can be used to update an accumulated subproduct (or subproducts, if multiple ℓ-isogeny images are required) that belongs to the i-th processor. After the initial phase that assigns the three values to each processor, no communication is required between the processors until the end, where the subproducts can all be collected and multiplied together. In the case of computing image points as in (4), then one final squaring and one final multiplication are used to finish the routine; in the case of computing image curves as in (5), then $\log(\ell)$ final multiplications and squarings are required. Note that this parallelisation can be exploited across all of the prime degree isogenies (comprising one full secret isogeny computation) that are large enough to make it worthwhile.

Precomputation. Assume Bob is tasked with large prime degree isogenies and he is the one generating ephemeral public keys. The runtime of his public key generation procedure can be improved if storage permits a significant offline precomputation. Recall from Section 3 that he computes his secret generator point S_α as $S_\alpha = Q_B + [\alpha]P_B$, where $\alpha \in \{0, 1, 2, \dots, L-1\}$ for $L = \prod_{i=1}^{k} \ell_i^{e_i}$, and where $\{P_B, Q_B\}$ is a basis for $E_0[L]$. Let ℓ_k be the largest prime dividing L and assume that $e_k = 1$ (which is the most common scenario when ℓ_k is large). Bob can arrange the ℓ_k-isogeny to come first, and can precompute all of the possible first steps by first setting $Q'_B = [L']Q_B$, $P'_B = [L']P_B$ for $L' = L/\ell_k$. For each $\alpha' \in \{0, 1, 2, \dots \ell_k - 1\}$, he then computes $S'_\alpha = Q'_B + [\alpha']P'_B$ before computing the isogeny $E_{\alpha'} = E_0/\langle S'_\alpha \rangle = \phi_{\alpha'}(E_0)$, together with the images of Alice's points, $\phi_{\alpha'}(P_A)$ and $\phi_{\alpha'}(Q_A)$. The list of ℓ_k triples $(E_{\alpha'}, \phi_{\alpha'}(P_A), \phi_{\alpha'}(Q_A))$ is stored in memory and he can now avoid the ℓ_k-isogeny at runtime by simply choosing the triple corresponding to his secret key (or defining the secret key corresponding to his chosen triple).

More torsion from above. There may be scenarios where there are large enough isogeny degrees that it becomes preferable to instead use torsion points whose coordinates are no longer in $\mathbb{F}_{p^2}$. Suppose that Alice takes $\{P_A, Q_A\}$ as a basis for $E_0(\mathbb{F}_{p^2})[M]$ to work over $\mathbb{F}_{p^2}$ as usual (e.g. with $M = 2^m$), but that Bob, faced with the predicament of at least one large n_i in $N = \prod_{i=1}^{b} n_i^{\nu_i}$, would prefer to work with $\tilde{N} = \prod_{i=1}^{\tilde{b}} \tilde{n}_i^{\tilde{\nu}_i}$, where $E_0[\tilde{N}] \not\subset E_0(\mathbb{F}_{p^2})$, but rather $E_0[\tilde{N}] \subset \mathbb{F}_{p^{2k}}$; his rationale

being that the $\tilde{n}_i$ are small enough to make computing the $\tilde{N}$-isogeny in $\mathbb{F}_{p^{2k}}$ faster overall. So long as $\tilde{N} \approx N$ (to maintain the requisite security), the SIDH framework allows for this, but with some caveats that apply to both sides. Bob's basis points are now $P_B, Q_B \in E_0(\mathbb{F}_{p^{2k}})[\tilde{N}]$, and though the inflated public parameters are not a problem in most practical scenarios, the main drawback is that Alice's performance suffers from having to drag larger coordinates through her secret isogenies during the generation of her public key. Her secret kernel, image curve, and indeterminate isogeny computations are all in $\mathbb{F}_{p^2}$, but she must also operate in $\mathbb{F}_{p^{2k}}$ when evaluating her isogenies at Bob's points during public key generation. Moreover, her uncompressed public keys are now three x-coordinates in $\mathbb{F}_{p^{2k}}$, and the public key compression techniques she would need to perform to reverse the inflation[6] would likely be unweildly inside $E(\mathbb{F}_{p^k})[\tilde{N}]$. Nevertheless, Bob's public keys would stay the same size; the three x-coordinates he produces during key generation are still in the $\mathbb{F}_{p^2}$-rational M-torsion, so there may be scenarios (e.g. where a particularly nice prime p produces an unfortunately large factor of $p-1$) where the overall trade-off becomes worthwhile. The group orders to examine for smaller factors, obtained by lifting up to $E(\mathbb{F}_{p^{2k}})$, are $(p^k \pm 1)^2$; again, the sign choice here corresponds to the two choices of quadratic twists over $\mathbb{F}_{p^{2k}}$.

Finally, it is worth noting that there are other twists available for the special curves with $j = 0$ and $j = 1728$ [49, Proposition X.5.4], and depending on the prime, one or both of these curves are often supersingular [49, p. 152]. Though these twists can also be used to obtain points with new orders, unlike σ in (6), the twisting morphisms now move the x-coordinates to the correpsonding extension field. Nevertheless, it could be that simplifications are possible in the isogeny formulas, or perhaps that these twists behave more favourably in other curve models, but even if this is the case it appears that these morphisms can only be exploited during key generation.

4 Accelerating Alice, burdening Bob

This section presents some example instantiations where Alice can compute her 2^m-isogenies over faster primes. These primes are sampled from the pre-existing ECC literature (cf. [27, §3.1]), and imposing these faster shapes is what ultimately slows Bob down; he is forced to compute isogenies whose degrees are whatever the other prime factors divide $p^2 - 1$. The list below is far from exhaustive and it is likely that there are prime shapes offering better trade-offs. Implementers searching for their own primes should be reminded that $p^2 - 1$ need not be entirely smooth. So long as Alice and Bob can both compute coprime isogenies of large enough degrees to resist the known attacks, large prime factors of $p^2 - 1$ can be treated as cofactors within the SIDH/SIKE frameworks. In what follows, underlines are used to distinguish factors of $p+1$ and factors of $p-1$ in examples that are subject to the caveat at the end of Section 3.

The Mersenne prime(s). Example 1 already studied the most attractive example of Mersenne prime in the interesting ranges: with $p = 2^{521} - 1$, the main takeaways were Alice being solidly faster at the cost of Bob being (probably orders of magnitude) slower, Alice being able to easily scale her security both with or without Bob, Alice having the option of walking the diameter of the 2-isogeny graph, and a nice classical/post-quantum hybrid.

There are only two other Mersenne primes worth mentioning. The next largest Mersenne prime is $p = 2^{607} - 1$, for which $p^2 - 1 = 2^{608} \cdot 3^2 \cdot 7 \cdot 607 \cdot \ell_{27} \cdot \ell_{43} \cdot \ell_{59} \cdot \ell_{100} \cdot \ell_{174} \cdot \ell_{190}$, where the ℓ_b represent b-bit primes; this is clearly an unlucky factorisation offers Bob no combination of reasonable computability and security. The next smallest Mersenne prime is $p = 2^{127} - 1$, which is far too small to offer any reasonable security in the elliptic curve setting, but it *may* become interesting in light of the recent work on genus 2 isogenies. In [22, §4.1], Flynn and Ti conjecture that a prime of roughly 171 bits would meet the 128-bit security level, but this is under the same construction as Jao and De Feo, i.e. $p = f \cdot 2^m 3^n - 1$, with f a small cofactor and $M = 2^m$ and

[6] Compression afficianados should note that the j-invariant is still in $\mathbb{F}_{p^2}$, and that scalars in $\mathbb{Z}_{\tilde{N}}$ are no larger than p.

$N = 3^n$ both around 85 bits. If, instead, the prime $p = 2^{127} - 1$ were to be used[7], then Alice could compute 2^m-isogenies (using any m up to 127, if she liked) and Bob could compute secret isogenies of odd degree dividing

$$2^{127} - 2 = 2 \cdot 3^3 \cdot 7^2 \cdot 19 \cdot 43 \cdot 73 \cdot 127 \cdot 337 \cdot 5419 \cdot 92737 \cdot 649657 \cdot 77158673929.$$

Taking $N = 3^3 \cdot 7^2 \cdot \cdots \cdot 92737 \cdot 64965$ gives $\lceil \log_2(N) \rceil = 90$. Moreover, Flynn and Ti describe their algorithms on the Kummer surfaces where arithmetic is also twist-agnostic.

The Ridinghoods. Example 2 studied the prime $p = 2^{448} - 2^{224} - 1$ underlying Hamburg's Goldilocks curve, which is an example of a Ridinghood prime of the form $p = 2^{2k} - 2^k - 1$; Hamburg [27] shows that these primes facilitate arithmetic that is particularly Karatsuba-friendly. The other values of $k \in \{161, 208, 224, 225, 240, 354\}$ give examples that are largely similar to Example 2, where Alice works in the 2^k-torsion and Bob can choose a product of primes from the remaining odd factors of $p^2 - 1$. The example below is very close to the Goldilocks prime, and although the prime is not *word-aligned*, it does give Bob significantly smaller isogeny degrees.

Example 3 (Ridinghood-p450). With $p = 2^{450} - 2^{225} - 1$, Alice can compute 2^{225}-isogenies to match the security of SIKEp434, and Bob can compute N-isogenies, where

$$\begin{aligned} N = \underline{3^4} \cdot \underline{5} \cdot 7 \cdot \underline{11} \cdot \underline{17} \cdot \underline{19} \cdot \underline{29} \cdot 31 \cdot \underline{43} \cdot 73 \cdot \underline{113} \cdot \underline{127} \cdot 151 \cdot \underline{251} \cdot \underline{257} \cdot \underline{331} \cdot \underline{449} \cdot 601 \\ \cdot 631 \cdot 1801 \cdot \underline{2689} \cdot \underline{4051} \cdot \underline{5153} \cdot 23311 \cdot \underline{65537} \cdot 100801 \cdot 115201. \end{aligned}$$

Here $\lceil \log_2 N \rceil = 228$ and the largest isogeny Bob would need to compute has $\ell = 115201$, which is much smaller than the largest isogeny with $\ell = 15790321$ in Example 2.

Solinas primes. It may be also fruitful to search over primes of the form $p = 2^\alpha - 2^\beta \pm \cdots \pm 2^\kappa \pm 1$, particularly if κ is large enough that Alice can take $M = 2^\kappa$. As usual, the search is looking for a large odd factor, N, of $p^2 - 1$ that is smooth enough for Bob to be able to compute N-isogenies. The only such instances that were checked in the preparation of this paper were the six primes underlying the NIST curves, none of which had κ large enough and favourable factorisation of $p^2 - 1$.

Montgomery-friendly primes. While there are only a handful of Mersenne and Ridinghood primes, there are vastly more Montgomery-friendly primes, which are usually defined (see [9, §3.2] or [27]) as being of the form $p = 2^k \cdot c - 1$. For a fixed k and an upper bound on the bitlength of p, this gives many values of c that can be searched over until $N \mid c(p-1)$ is both large enough for Bob to be secure, and smooth enough that he can compute the isogenies. A quick search with $k = 216$ (to match SIKEp434 exactly) and $1 \leq c < 2^{40}$ found the following two examples.

Example 4. Over the 254-bit Montgomery-friendly prime $p = 2^{216} \cdot c - 1$ with $c = 137439113067$, Alice can compute 2^{216}-isogenies as usual and Bob can compute N-isogenies, with

$$\begin{aligned} N = 3^3 \cdot \underline{5} \cdot 167 \cdot \underline{613} \cdot \underline{2543} \cdot \underline{5167} \cdot \underline{96443} \cdot \underline{204509} \cdot \underline{778769} \cdot \underline{3357979} \cdot 30481063 \\ \cdot \underline{172664567} \cdot \underline{955640417}. \end{aligned}$$

Example 5. Over the 254-bit Montgomery-friendly prime $p = 2^{216} \cdot c - 1$ with $c = 1030791469290$, Alice can compute 2^{216}-isogenies as usual and Bob can compute N-isogenies, with

$$\begin{aligned} N = 3 \cdot 5 \cdot \underline{7} \cdot 11 \cdot \underline{31} \cdot \underline{137} \cdot 7411 \cdot 421483 \cdot \underline{467431} \cdot \underline{765031} \cdot \underline{1511737} \cdot \underline{119055253} \\ \cdot \underline{179824133} \cdot \underline{332548273} \cdot \underline{617693863}. \end{aligned}$$

[7] Here the starting curve has been changed from their choice to $C/\mathbb{F}_{p^2} : y^2 = x^5 + x$, the Jacobian of which is supersingular with $\#J_C(\mathbb{F}_{p^2}) = (p+1)^4$.

Note that both of these examples have Bob taking torsion from both sides, taking factors from both c and $p-1$. To avoid the caveat in Section 3, similar searches could disallow factors of c so that Bob can work solely with torsion from the twists. Note that the search that found the above examples was far from exhaustive; there are likely to be much better examples with $k = 216$ and almost certainly better examples if k is varied.

On the other hand, allowing torsion from both sides and searching over larger (smooth) c can give rise to much smoother N-isogenies.

Example 6. With $c = 48339122132979549776760416299303657 7$, $p = 2^{256} \cdot c - 1$ is a 375-bit prime p. Here Alice can take $M = 2^{216}$ to exactly match the 2^m-torsion used in SIKEp434, and Bob can take

$$N = \underline{3} \cdot \underline{5} \cdot \underline{107} \cdot 263 \cdot 401 \cdot \underline{409} \cdot 1249 \cdot \underline{1451} \cdot 1493 \cdot \underline{1693} \cdot \underline{8011} \cdot 8423 \cdot \underline{10331} \cdot 10501 \cdot 11519 \cdot 12829 \cdot 13049 \cdot \underline{13477} \cdot \underline{14149} \cdot 14411,$$

which has $\lceil \log_2(N) \rceil = 213$, roughly matching the 3^n-torsion used in SIKEp434. Again, this would found with a naive search that impatiently doubled the smoothness bound B until an example with a large enough N was found within a few minutes; this example was found with $B = 16384$.

Example 7. The previous value of $B = 8192$ found a 383-bit prime $p = 2^{256} \cdot c - 1$ with $c = 149129193885612563981396585707851113249$. In this case Alice can again take $M = 2^{216}$, and Bob can take

$$N = \underline{3} \cdot \underline{7} \cdot 401 \cdot \underline{431} \cdot \underline{557} \cdot 1117 \cdot 1151 \cdot \underline{2027} \cdot 2801 \cdot \underline{2861} \cdot 2879 \cdot \underline{3623} \cdot 3803 \cdot 4271 \cdot \underline{5393} \cdot 6301 \cdot 6343 \cdot 7013 \cdot \underline{7039} \cdot 7879,$$

which has $\lceil \log_2(N) \rceil = 209$, sacrificing a few bits of security over the previous example for what would be a much faster N-isogeny computation.

In Examples 4–7, Alice will benefit from smaller primes that come with additional arithmetic advantages over the SIKEp434 setting. Bob's isogeny would be much more expensive than Alice's (and his 3^{137}-isogeny in SIKEp434), but he too gains some performance back from the underlying field. Note that the public keys in these examples would be smaller than those offered by uncompressed SIKEp434.

In all of the above examples, the prime p is still significantly larger than the degree of Alice and Bob's secret isogenies. This means that, asymptotically, the $O(M^{1/2}) \approx O(N^{1/2})$ claw-finding algorithm would still be preferred over the $O(p^{1/2})$ Delfs-Galbraith algorithm to find Alice and Bob's secret isogenies. When the sizes of M and N start to get closer to the size of p, as in Examples 4 and 5, determining the best attack may require a closer examination of the concrete complexity in (11) compared to the concrete complexity of the Delfs-Galbraith algorithm – see Section 6.

5 Optimal public keys

In terms of the size of isogeny degrees compared to the size of the prime p, the optimal scenario is when $M \approx N \approx p$. In theory, the construction described in Section 3 allows Alice to compute isogenies inside the $(p+1)$-torsion and Bob to compute isogenies inside the $(p-1)$-torsion using only arithmetic in $\mathbb{F}_{p^2}$, for *any* prime p. For this to work in practice, however, both $(p+1)$ and $(p-1)$ need to be smooth enough that these isogenies are computable. Recall that once a factor 2 is removed from whichever of $(p+1)$ and $(p-1)$ is not divisible by 4, the remaining two values are immediately coprime.

Størmer's theorem. The naive way to look for instances where this is possible is by searching through primes p of the appropriate size (more on this in a moment) until $p^2 - 1$ is smooth enough; note that non-smooth cofactors are tolerable in the SIDH framework but this sacrifices optimality. It is possible to do much better than the naive search thanks to a beautiful theorem of Størmer from 1897 [50] that provides exactly what is needed. Given an upper smoothness bound B, Størmer's theorem describes the full set of integers that are sandwiched between two B-smooth numbers. Moreover, he shows that this set is finite, and gives a way to find it via solving a system of Pell equations. In 1964, Lehmer [36] improved Størmer's algorithm to minimize the number of Pell equations that need solving and presented a version of the theorem specific to the situation at hand.

Theorem 1 ([36]). *Let*

$$q_1 < q_2 < \cdots < q_t$$

be a given set of t primes, and let Q be the set of numbers generated by them. Let Q' be the subset of all square-free members of Q. Let S be a number such that S and $S+2$ belong to Q. Then $S = x_n - 1$ where (x_n, y_n) is a solution of the Pell equation

$$x^2 - Dy^2 = 1 \tag{9}$$

in which

$$1 < D \in Q', \qquad 1 \le n \le \max\{3\,,\,(q_t+1)/2\}\,, \qquad y_n \in Q. \tag{10}$$

Conversely. if (x_n, y_n) is a solution of (9) subject to (10), then both $S = x_n - 1$ and $S+2$ belong to Q.

In other words, Theorem 1 says that two integers S and $S+2$ are B-smooth if and only if the number between them, $x_n = S+1$, is a solution (x_n, y_n) to the Pell equation in (9), where D is squarefree, and both D and y_n are B-smooth. Only one additional stipulation is needed in order to apply this theorem to the present context: that the solution $x_n = S+1$ is a prime.

Theorem 1 allows an arbitrary set of primes q_i. However, it is immediately clear that if $S+1$ is to be prime, then $q_1 = 2$ and $q_2 = 3$ must be included (since S and $S+2$ must both belong to Q). Moreover, if q_t is the largest prime up to B, then the main bottleneck in finding applicable solutions is that y_n factors over Q; thus, it does not help to exclude any primes below q_t in the search below, so henceforth q_j will always be the j-th prime.

It is helpful to see Theorem 1 in action by means of a toy example. Let $B = 19$ so that $t = 8$ and $\{q_1, q_2, \ldots, q_7, q_8\} = \{2, 3, \ldots, 17, 19\}$. The set Q is the infinite set of 19-smooth numbers, while the set Q' is the squarefree subset of Q, containing the $2^t - 1 = 255$ elements of the form $q_1^{\beta_1} q_2^{\beta_2} \ldots q_8^{\beta_8}$ with $\beta_i \in \{0, 1\}$. To collect all of the integers x such that $x-1$ and $x+1$ are B-smooth, there are 255 Pell equations to solve. For each $D \in Q'$, Equation (9) must be solved for the *fundamental solution* (x_1, y_1) [29], which can then be used to generate the solutions (x_n, y_n) up to $n = (q_t+1)/2 = 10$, via the recurrence $(x_{m+1}, y_{m+1}) = (2x_1x_m - x_{m-1}, 2x_1y_m - y_{m-1})$, with $(x_0, y_0) = (1, 0)$ [36, §6].

Whenever a solution (x_n, y_n) to any of the Pell equations is found, if y_n is B-smooth then x_n is such that $x_n - 1$ and $x_n + 1$ are also B-smooth. With $D = 176358 = 2 \cdot 3 \cdot 7 \cdot 13 \cdot 17 \cdot 19$, the first 10 solutions are

$$\begin{aligned}
(x_1, y_1) &= (8399, 20),\\
(x_2, y_2) &= (141086401, 335960),\\
(x_3, y_3) &= (2369969355599, 5643456060),\\
&\vdots\\
(x_{10}, y_{10}) &= (150246143559426357708268378140491262955498163 99,\\
&\qquad 35777150761840526645502984218838604303680220).
\end{aligned}$$

Sieving the y_i yields that y_1 is the only B-smooth number, and hence $x_1 = 8399$ is the only value found with $D = 176358$ such that $x_1 - 1 = 2 \cdot 13 \cdot 17 \cdot 19$ and $x_1 + 1 = 2^4 \cdot 3 \cdot 5^2 \cdot 7$ are B-smooth. In reference to the present context, x_1 is not prime, so this D provides no solutions to the problem at hand.

When $D = 9690 = 2 \cdot 3 \cdot 5 \cdot 17 \cdot 19$, $(x_1, y_1) = (330751, 3360)$ has $y_1 = 2^5 \cdot 3 \cdot 5 \cdot 7$ as B-smooth, so that $x_1 = 330751$ has $x_1 - 1 \in Q$ and $x_1 + 1 \in Q$; this is the largest x_1 found over all of the $D \in Q'$, and is thus the largest such integer that is sandwiched between two 19-smooth numbers.

The largest such prime integer is $p = 4751$, found as the fundamental solution $(x_1, y_1) = (4751, 60)$ to (9) with $D = 6270 = 2 \cdot 3 \cdot 5 \cdot 11 \cdot 19$, which is such that $p - 1 = 2 \cdot 5^3 \cdot 19$ and $p + 1 = 2^4 \cdot 3^3 \cdot 11$.

Searching for prime sandwiches on smooth buns. For the primes from $B = 19$ to $B = 43$, Table 1 lists both the largest x_n such that $x_n - 1$ and $x_n + 1$ are B-smooth, and the largest such x_n that is a prime, as well as their bitlengths.

Table 1. The largest integers x_n that are sandwiched between two B-smooth numbers, together with the largest prime such integers, for various B.

B	largest x_n	bits	largest p	bits
19	330751	19	4751	13
23	657019	20	22541	15
29	3704401	22	2470337	22
31	740512499	30	2470337	22
37	740512499	30	14209999	24
41	127855050751	37	61889827	26
43	842277599279	40	842277599279	40

For such small B, searches that solve all of the required Pell equations can be done in a matter of seconds or minutes. In looking for cryptographically sized p, however, exhaustively solving the full set of $2^t - 1$ Pell equations becomes infeasible. Lehmer points out that the upper bounds obtained for the sizes of the x_n one can expect to find for a given B are "very weak" [36, §5], so there is no real gauge on how large B needs to become to expect to find a prime p of a given size.

As was mentioned in Section 1, a naive and very unoptimised search was performed on a desktop computer to obtain a placeholder example as a proof of concept. In the coming months, as this search is optimised and resources are ramped up, it is expected that much better examples will be more readily found at the target security levels. As is detailed in the next section, the best known classical attacks over optimal parameters with $M \approx N \approx p$ run in $O(p^{1/2})$ time and the best known quantum attacks run in $O(p^{1/4})$ time. Thus, as a very rough gauge on the size of the desired parameters, a prime geared towards NIST's security level 1 [56] should lie somewhere in the range from 200 to 300 bits. For a given size of p, the quality of an example is judged solely by the smoothness bound; if a prime p is such that $p-1$ and $p+1$ are B-smooth, then another example will be deemed better if it is also around the same size but achieves a smoothness bound less than B; beyond the size of p itself, it is hopeless to try adding any (e.g. arithmetically favourable) properties of the prime p to the wishlist.

Two excellent references for methods of solving the Pell equation are the book by Jacobson and Williams [29] and the survey of Lenstra [37]. It is well known that solving the Pell equation generally becomes harder as D grows large; Lenstra notes that *any* method that produces the fundamental solution (x_1, y_1) is exponentially slow for infinitely many D. Fortunately, the *simple continued fraction* (SCF) method is all that is needed for the application at hand; moreover, imposing an upper bound on the sizes of the primes p means that this method can be aborted rapidly once the SCF exceeds this bound.

The example below was found with the SCF exactly[8] as described in [29, p. 59], with an upper bound of 2^{256} imposed on the running value G_{i-1}. The search impatiently doubled the smoothness bound B and, noting that smaller values of D tend to give smaller SCF periods and thus produce solutions, searched through values of $D = 2 \cdot 3 \cdot q_3 \cdot q_4 \cdot q_5$ with $q_3, q_4, q_5 < 2^{10}$ (and where q_3, q_4 and q_5 are allowed to be 1) until a solution was found. With $B = 2^{22}$, a solution was found with $D = 5960598 = 2 \cdot 3 \cdot 7 \cdot 139 \cdot 1021$. In this case, the fundamental solution of the Pell equation is

$$(x_1, y_1) = (2000839684562944911084089536 01, 819534755505253055240719 20),$$

for which y_1 is B-smooth but x_1 is not prime. However, the subsequent values (x_2, y_2) and (x_3, y_3) were also returned as both have $x_i < 2^{256}$. The value of y_3 is not B-smooth, but the solution

$$(x_2, y_2) = (800671888664388978464540516446276703083486508057275417344 01,$$
$$3279515323387001406912201773220406152305636352773396784 0)$$

does have y_2 being B-smooth, and moreover, $p = x_2$ is a 196-bit prime. In this case

$$p - 1 = 2^{10} \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 17^2 \cdot 41^2 \cdot 43^2 \cdot 53^2 \cdot 139 \cdot 523^2 \cdot 1021 \cdot 24547^2 \cdot 95651^2 \cdot 175061^2, \quad \text{and}$$
$$p + 1 = 2 \cdot 11^2 \cdot 31^2 \cdot 2011^2 \cdot 7207^2 \cdot 22709^2 \cdot 23041^2 \cdot 42257^2 \cdot 1831021^2.$$

In this example the 175061- and 1831021-isogenies are managable but still highly suboptimal. Over the coming months, exhaustive searches over much smaller smoothness bounds are expected to supercede this one, and to produce much faster isogenies in the scenario where the sizes of public keys are optimsed.

Note that, once a 256-bit example p is found, the uncompressed public keys will be three elements of $\mathbb{F}_{p^2}$ (as in uncompressed SIDH/SIKE), i.e. around 192 bytes. This is significantly less than the 330-byte public keys of uncompressed SIKEp434, and is even less than the 196-byte *compressed* public keys of SIKEp434.

6 Security analysis

At a high level, there are two main changes to the usual computational isogeny problems underlying SIDH and SIKE [21, Problems 5.1–5.4] that are implicit in this paper. The first is that the isogeny walks now use multiple values of ℓ; the vertex set of a given graph stays fixed, but the edges now change between successive steps. The second is that the walks are no longer *half-length* (i.e. around half the bitlength of p); lowering the size of the primes relative to the length of the walks means that other avenues of attack become relevant with respect to the usual meet-in-the-middle attacks. This section studies the implications of these changes with respect to known attacks from the literature.

Before focusing on the underlying computational problems, the following remark briefly revisits the two other practical security notions discussed in Section 1.

Remark 1 (Active adversaries). Recall from Section 3 that protecting Algorithm 1 implementations against side-channel adversaries is analogous to protecting pre-existing SIDH/SIKE algorithms for 2^m- and 3^n-isogenies – see [30, §5.3]. While it is stressed that static keys in the proposed instantiations should be safeguarded analogously to SIDH static keys (i.e. by using SIKE), it is worth pointing out a peculiarity that arises when large prime ℓ-isogenies are on the side of the party re-using a static key. In the case of actively attacking a static secret 2^m-isogeny, the Galbraith-Petit-Shani-Ti attack [24] recovers one bit of the key during every protocol interaction; it only requires around m malicious interactions with the static key holder to compute their entire secret key. However, an active adversary seems to get far less *bang for their buck* when applying this attack to the case where the computation of the static key involves large prime isogenies. Let

[8] Note that the errata of [29] replaces Q_i with Q_{i+1} in the line that assigns q_{i+1}.

$L = \prod_{i=1}^{k} \ell_i^{e_i}$ with $\ell_1 < \cdots < \ell_k$. At each point of the attack, the adversary maliciously chooses points $\tilde{P}$ and $\tilde{Q}$, sends the basis $\{P + \tilde{P}, Q + \tilde{Q}\}$ instead of the honestly generated basis $\{P, Q\}$, and then checks (i.e. queries the oracle – see [24, §3]) to see if $\langle Q + \tilde{Q} + [\alpha](P + \tilde{P})\rangle \stackrel{?}{=} \langle Q + [\alpha]P\rangle$. If the final isogeny in the computation of the static public key is an ℓ_k-isogeny, then the adversary's first step involves trying $\tilde{P} = [\prod_{i=1}^{k-1} \ell_i^{e_i}]P$ with $\tilde{Q} = [-x \cdot \prod_{i=1}^{k-1} \ell_i^{e_i}]P$ for $0 \leq x < \ell_k$ until the key exchange succeeds (i.e. the oracle in [24, §3] returns "1") , at which point he has found $0 \leq \alpha_\ell < \ell_k$ such that that $\alpha_\ell \equiv \alpha \bmod \ell_k$, and he can then move onto solving $\alpha \bmod \ell_{k-1}$ (or $\alpha \bmod \ell_k^2$ if $e_k > 1$). On average, the attacker would need $\ell_k/2$ oracle queries to find α_ℓ, so when $\ell_k \gg m$ (as it is for most of the examples in this paper) he is using many more queries to recover what is only a small fraction of the entropy he can recover in the 2- or 3-power case (note that the attacker learning less information per oracle query for larger ℓ was already pointed out in [24, Remark 2]). Although it is both outside the scope of this paper and not recommended without a more detailed analysis, the types of large ℓ that arise in some of the examples do hint towards a potential trade-off where SIDH keys could be reused for a small number of protocol runs that is upper-bounded by some tolerable security loss. In the case of the original SIDH protocol reusing 2^m-isogenies, one bit of security can be presumed lost every time a static key is (re)used, but in the case of a secret L-isogeny with $\ell_k > 2^\kappa$ and $\kappa = \kappa_1 + \kappa_2 + 1$, one might be willing to trade off the $2^{-\kappa_2}$ probability of losing κ bits of security in order to reuse a single key up to 2^{κ_1} times, particularly if the parameters are chosen to account for this trade-off.

Multiple edge sets. Based on current knowledge, there is no reason to believe that a walk consisting of many different prime degree isogenies makes the underlying problem appreciably easier than that of a walk in a fixed ℓ-isogeny graph, provided the number of possible destination nodes is around the same size. When computing L-isogenies with $L = \prod \ell_i^{e_i}$, the number of cyclic subgroups of order L inside any given group $E(\mathbb{F}_{p^2})$ is $\prod(\ell_i + 1)\ell_i^{e_i - 1}$, and so long as this is around the same size as $(\ell + 1)\ell^{e-1}$, the difficulty of recovering an L-isogeny appears to be no easier than that of recovering an ℓ^e-isogeny. The generalisation of the problems underlying SIDH to isogenies of multiple degrees has already been considered in prior works (e.g. [42] and [25, §2.3]), where the same conclusion was drawn (or the same assumption was made).

Classical cryptanalysis. The remainder of this section focusses on the upshot of computing secret L-isogenies, where L is much larger than $p^{1/2}$. When $L \approx p^{1/2}$, as in the original SIDH proposal, the *meet-in-the-middle* or *claw-finding* algorithms [21, §5.3] stand alone as the best known attacks against SIDH and SIKE. However, the most interesting instantiations proposed in this paper have $L \gg p^{1/2}$, and as L tends towards p, algorithms other than the meet-in-the-middle attacks become relevant.

In what follows it will be assumed that $L = \ell_i^{e_i} \approx p$, as was the case in Section 5, since this is the extreme case where the alternative attack avenues are most relevant. The underlying problem is to find the isogeny

$$\phi \colon E_1 \to E_2$$

of degree L, where $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$ are supersingular. Let $\{P_1, Q_1\}$ be a basis for $E_1[L]$, and recall from (8) that the number of distinct order L subgroups that can be generated by points of the form $S_1 = Q_1 + [\alpha]P_1$ is exactly L.

Meet-in-the-middle algorithms. Let $L_1 \approx L_2 \approx p^{1/2}$ with $L_1 L_2 = L$. The claw-finding algorithm cited by Jao and De Feo [31, §5.2] uses $O(L_1)$ time to compute a table of all of the curves L_1-isogenous to E_1, and stores them using $O(L_1)$ memory. It then proceeds by trying one L_2-isogeny at a time, this time emanating from E_2, until a match is found in the table and the problem is solved; this stage requires $O(L_2)$ time and essentially no memory. It follows that the claw-finding algorithm runs in $O(p^{1/2})$ time and requires $O(p^{1/2})$ memory.

Adj, Cervantes-Vázquez, Chi-Domínguez, Menezes and Rodríguez-Henríquez [1] proposed that the van Oorschot-Wiener (vOW) parallel collision finding algorithm [57] has a lower overall cost for

finding ϕ, and thus should be used to assess the security of SIDH and SIKE. Their implementation confirmed that the original vOW runtime analysis [57, Equation (4)] is sharp in the context of finding the isogeny ϕ. If w is the number of entries that can be stored in the table above, m is the number of processors running in parallel, and t is the time taken to compute L_1 and L_2 isogenies, then the vOW algorithm finds ϕ in expected runtime

$$T = \frac{2.5}{m} \cdot \left(\frac{p^{3/4}}{w^{1/2}} \right) \cdot t. \tag{11}$$

Adj et al. conclude that $w > 2^{80}$ is infeasible, so conduct their analysis by setting $w = 2^{80}$. In the discussion below it helps to observe that (with this choice of w) the runtime in (11) becomes $T = 2.5 \cdot t \cdot p^{1/2}$ when $p = 2^{160}$. For p larger than this size (as it is in all of the proposed examples) and with w fixed, the runtime becomes significantly larger than $p^{1/2}$. Following the work of Adj et al., two subsequent papers [32,17] conducted separate analyses that confirmed the runtime in (11).

When the degree of the isogenies is much less than p (as in SIDH and SIKE), the above claw-finding meet-in-the-middle algorithms are the superior classical algorithms; although their asymptotic complexities are often cited (or rewritten) in terms of p, they are actually dependent on the length of the walk, so (as was done for (11)) this length is typically used in the derivation of the runtime before it is replaced with $p^{1/2}$. However, when $L \approx p$, Galbraith's original meet-in-the-middle algorithm [23] becomes asymptotically competitive with the claw-finding algorithms. Just like claw-finding, it requires $O(p^{1/2})$ memory and runs in $O(p^{1/2})$ time.

It is important to note that when $L \approx p$ it is possible that the secret isogeny ϕ is no longer the shortest path between E_1 and E_2. It is reasonable to assume that the table collision found via the claw-finding algorithm will correspond to ϕ, but running Galbraith's algorithm (as it stands) may return a different path between E_1 and E_2. Nevertheless, if this path (or the algorithm itself) cannot be modified to find the correct path, other paths can presumably be discarded at no loss of asymptotic efficiency. For the examples in this paper where there is still a comfortable margin between the size of L and the size of p, the correct path is again likely to be the shortest, so Galbraith's (unmodified) algorithm is likely to terminate with ϕ.

Random walk algorithms for any path. There are two styles of applicable random walk algorithms that can be used to solve the general supersingular isogeny problem: both Pollard rho [45] and Delfs-Galbraith [19] find *some* path between E_1 and E_2. The former finds an isogeny between E_1 and E_2 by taking two pseudo-random walks in the graph of size $O(p)$; the number of steps required until these two walks collide is $O(p^{1/2})$ by the birthday paradox. The latter algorithm, which is preferred in practice (see [19, §4] or [7]), uses two self-avoiding random walks to find paths from each curve to two subfield curves, $\tilde{E}_1/\mathbb{F}_p$ and $\tilde{E}_2/\mathbb{F}_p$, and then connects these two subfield curves. Since there are $O(p^{1/2})$ subfield curves in the graph of size $O(p)$, the first step requires $O(p^{1/2})$ steps, and since connecting the two subfield curves requires $O(p^{1/4})$ steps [19], the entire algorithm takes $O(p^{1/2})$ steps to find an isogeny connecting E_1 and E_2. Like vOW, the Delfs-Galbraith algorithm parallelises perfectly, but unlike vOW, it does not have large storage requirements.

Both of these algorithms are likely to terminate with a path that is not the secret path corresponding to ϕ. However, since E_1 is typically a special curve with a known endomorphism ring $\mathrm{End}(E_1)$, it is prudent to assume that this can be used to modify the path into the correct one via the techniques discussed at length in [25, §4].

Quantum cryptanalysis. The best known quantum algorithm for solving SIDH and SIKE instances is, asymptotically, Tani's algorithm [53]. Roughly speaking, as $p \to \infty$, Tani's algorithm solves the claw-finding problem for secret isogenies of degree $O(p^{1/2})$ in time $O(p^{1/6})$ on a quantum computer. Translating to the setting of isogenies of degree $L \approx p$, this would give an $O(p^{1/3})$ quantum claw-finding algorithm; note that recent work of Jaques and Schanck [32] shows that (even under the assumption of a large amount of quantum resources) the concrete complexity of Tani's algorithm is much closer to the classical claw-finding complexity. Nevertheless, when $L \approx p$, Tani's

algorithm is no longer than superior algorithm for solving the corresponding isogeny problem. In [7], Biasse, Jao and Sankar give a quantum algorithm for the general supersingular isogeny problem (in characteristic p) that runs in time $O(p^{1/4})$. Their algorithm is essentially the Delfs-Galbraith algorithm (from above) ported to the quantum setting; they use Grover's algorithm [26] to get a quadratic speedup from $O(p^{1/2})$ to $O(p^{1/4})$ on the phase that finds the two supersingular subfield curves $\tilde{E}_1/\mathbb{F}_p$ and $\tilde{E}_2/\mathbb{F}_p$, and then develop a subexponential algorithm (based on the Childs-Jao-Soukharev subexponential algorithm [13] for the ordinary case) to connect the subfield path. The memory requirements of this algorithm are small; Biasse, Jao and Sankar define a set of N isogenies of degree 3^λ, where $\lambda \in O(\log(p))$ is chosen large enough so that this set contains a walk that passes through a subfield curve with probability $1/2$. As long as there are enough (i.e. $O(\log(p))$) qubits to encode such a path, then this algorithm succeeds with probability $1/4$ [7, Proposition 2].

As in the classical algorithms, since $\mathrm{End}(E_1)$ is typically known, the path obtained by the above process can presumably be modified into the path corresponding to ϕ at no additional asymptotic cost.

Security summary. When $\phi\colon E_1 \to E_2$ is an isogeny between two supersingular curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$ of degree $L = \prod_{i=1}^{k} \ell_i^{e_i} \approx p$, the best known classical algorithm for finding ϕ is the Delfs-Galbraith algorithm; it runs in $O(p^{1/2})$ time and (unlike claw-finding or vOW) does not have large storage requirements. Applying Grover's speedup to the Delfs-Galbraith algorithm also gives the best known quantum algorithm; it requires $O(\log(p))$ qubits, run in time $O(p^{1/4})$, and does not have large storage requirements. In the classical case, Delfs-Galbraith parallelises perfectly, where as Grover's algorithm is well-known to give a $\sqrt{m}$ speedup when parallelised across m quantum processors [59].

References

1. G. Adj, D. Cervantes-Vázquez, J. Chi-Domínguez, A. Menezes, and F. Rodríguez-Henríquez. On the cost of computing isogenies between supersingular elliptic curves. In C. Cid and M. J. Jacobson, editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 322–343. Springer, 2018.
2. R. Azarderakhsh, D. Jao, K. Kalach, B. Koziel, and C. Leonardi. Key compression for isogeny-based cryptosystems. In K. Emura, G. Hanaoka, and R. Zhang, editors, *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, AsiaPKC@AsiaCCS, Xi'an, China, May 30 - June 03, 2016*, pages 1–10. ACM, 2016.
3. D. J. Bernstein. Curve25519: new Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.
4. D. J. Bernstein and T. Lange. Towards Post-Quantum Cryptography in TLS. The year in post-quantum crypto, December 2018. Slides: `https://cr.yp.to/talks/2018.12.28/slides-dan+tanja-20181228-pqcrypto-16x9.pdf`, Video: `https://www.youtube.com/watch?v=ZCmnQR3_qWg&t=3010s`.
5. D. J. Bernstein, T. Lange, C. Martindale, and L. Panny. Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 409–441. Springer, 2019.
6. J. Biasse, A. Iezzi, and M. J. Jacobson. A note on the security of CSIDH. In Chakraborty and Iwata [11], pages 153–168.
7. J. Biasse, D. Jao, and A. Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. In W. Meier and D. Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2014.
8. X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH and ordinary isogeny-based schemes. Cryptology ePrint Archive, Report 2018/537, 2018. `https://eprint.iacr.org/2018/537`.

9. J. W. Bos and P. L. Montgomery. *Montgomery Arithmetic from a Software Perspective*, page 10–39. Cambridge University Press, 2017.
10. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: an efficient post-quantum commutative group action. In T. Peyrin and S. D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
11. D. Chakraborty and T. Iwata, editors. *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, volume 11356 of *Lecture Notes in Computer Science*. Springer, 2018.
12. S. Chapman. Fredrik Carl Mälertz Størmer, 1874-1957. *Biographical memoirs of fellows of the Royal Society*, 1958. `https://doi.org/10.1098/rsbm.1958.0021`.
13. A. M. Childs, D. Jao, and V. Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Mathematical Cryptology*, 8(1):1–29, 2014.
14. C. Costello and H. Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In Takagi and Peyrin [52], pages 303–329.
15. C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik. Efficient compression of SIDH public keys. In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 679–706, 2017.
16. C. Costello, P. Longa, and M. Naehrig. Efficient algorithms for supersingular isogeny Diffie-Hellman. In M. Robshaw and J. Katz, editors, *Advances in Cryptology — CRYPTO 2016 — 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 572–601. Springer, 2016.
17. C. Costello, P. Longa, M. Naehrig, J. Renes, and F. Virdia. Improved classical cryptanalysis of the computational supersingular isogeny problem. *IACR Cryptology ePrint Archive*, 2019:298, 2019.
18. J. M. Couveignes. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, 2006:291, 2006.
19. C. Delfs and S. D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Designs, Codes and Cryptography*, 78(2):425–440, 2016.
20. M. Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 14, pages 197–272. Springer, 1941.
21. L. De Feo, D. Jao, and J. Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 8(3):209–247, 2014.
22. E. V. Flynn and Y. B. Ti. Genus two isogeny cryptography. In J. Ding and R. Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 286–306. Springer, 2019.
23. S. D. Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics*, 2:118–138, 1999.
24. S. D. Galbraith, C. Petit, B. Shani, and Y. B. Ti. On the security of supersingular isogeny cryptosystems. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 63–91, 2016.
25. S. D. Galbraith, C. Petit, and J. Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.
26. L. K. Grover. A fast quantum mechanical algorithm for database search. In G. L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.
27. M. Hamburg. Ed448-Goldilocks, a new elliptic curve. *IACR Cryptology ePrint Archive*, 2015:625, 2015.
28. Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
29. M. J. Jacobson and H. C. Williams. *Solving the Pell equation*. Springer, 2009.

30. D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, V. Soukharev, and D. Urbanik. SIKE: Supersingular Isogeny Key Encapsulation. Manuscript available at `sike.org/`, 2017.
31. D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In B. Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011.
32. S. Jaques and J. M. Schanck. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 32–61. Springer, 2019.
33. G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005.
34. K. Kwiatkowski. Towards post-quantum cryptography in TLS. The Cloudflare blog, June 2019 `https://blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/`.
35. A. Langley, M. Hamburg, and S. Turner. Elliptic Curves for Security. Internet Research Task Force (IRTF) RFC7748, January 2016 `https://tools.ietf.org/html/rfc7748`.
36. D. H. Lehmer. On a problem of Störmer. *Illinois Journal of Mathematics*, 8(1):57–79, 1964.
37. H. W. Lenstra. Solving the Pell equation. *Notices of the AMS*, 49(2):182–192, 2002.
38. M. Meyer and S. Reith. A faster way to the CSIDH. In Chakraborty and Iwata [11], pages 137–152.
39. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
40. M. Naehrig and J. Renes. Dual isogenies and their application to public-key compression for isogeny-based cryptography. In *To appear in Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings. Preprint:* `https://eprint.iacr.org/2019/499.pdf`, 2019.
41. C. Peikert. He gives C-Sieves on the CSIDH. *IACR Cryptology ePrint Archive*, 2019:725, 2019.
42. C. Petit. Faster algorithms for isogeny problems using torsion point images. In Takagi and Peyrin [52], pages 330–353.
43. A. K. Pizer. Ramanujan graphs and Hecke operators. *Bulletin of the American Mathematical Society*, 23(1):127–137, 1990.
44. A. K. Pizer. Ramanujan graphs. *AMS/IP Stud. Adv. Math.*, 7:159–178, 1998.
45. J. M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.
46. Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. *arXiv preprint* `https://arxiv.org/abs/quant-ph/0406151`, 2004.
47. J. Renes. Computing isogenies between Montgomery curves using the action of $(0, 0)$. In T. Lange and R. Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*, pages 229–247. Springer, 2018.
48. A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `https://eprint.iacr.org/2006/145`.
49. J. H. Silverman. *The Arithmetic of Elliptic Curves, 2nd Edition.* Graduate Texts in Mathematics. Springer, 2009.
50. C. Størmer. Quelques théorèmes sur l'équation de Pell $x^2 - dy^2 = \pm 1$ et leurs applications. *Christiania Videnskabens Selskabs Skrifter, Math. Nat. Kl*, (2):48, 1897.
51. N. Sullivan. Measuring TLS key exchange with post-quantum KEM. Talk at the NIST Second PQC Standardization Conference, August 2019 `https://csrc.nist.gov/Presentations/2019/measuring-tls-key-exchange-with-post-quantum-kem`.
52. T. Takagi and T. Peyrin, editors. *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*. Springer, 2017.
53. S. Tani. Claw finding algorithms using quantum walk. *Theor. Comput. Sci.*, 410(50):5285–5297, 2009.
54. J. Tate. Endomorphisms of abelian varieties over finite fields. *Inventiones mathematicae*, 2(2):134–144, 1966.
55. The National Institute of Standards and Technology (NIST). Submission requirements and evaluation criteria for the lightweight cryptography standardization process, August, 2018. URL: `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf`.

56. The National Institute of Standards and Technology (NIST). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, December, 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.
57. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999.
58. J. Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris Sér. AB*, 273:A238–A241, 1971.
59. C. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.
60. G. Zanon, M. A. Simplício Jr., G. C. C. F. Pereira, J. Doliskani, and P. S. L. M. Barreto. Faster key compression for isogeny-based cryptosystems. *IEEE Trans. Computers*, 68(5):688–701, 2019.