Supersingular isogeny key exchange for beginners

Craig Costello

Microsoft Research, USA
craigco@microsoft.com

Abstract. This is an informal tutorial on the supersingular isogeny Diffie-Hellman protocol aimed at non-isogenists.

1 Introduction

A non-specialist seeking a basic understanding of the schemes remaining in the NIST post-quantum standardisation effort [15] would likely find that the SIKE protocol [8] has one of the highest barriers of entry. Indeed, it is occasionally an experience of the author that the sheer amount of background and jargon needed to get an isogeny-based talk off the ground is enough to overwhelm audience members into their laptops or underwhelm them to sleep. The purpose of this tutorial is to try and give a beginner's guide to Jao and De Feo's supersingular isogeny Diffie-Hellman (SIDH) protocol [9] by way of an illustrated toy example. Any reader that can grasp the toy example will find it trivial to extrapolate their understanding to the parameters of cryptographic size, e.g. those in the SIKE[1] proposal.

The aim is that this be a starting point for non-isogenists seeking a gentle introduction to the topic. As such, it is not intended to act as any kind of survey of the field of isogeny-based cryptography. The last few years have seen an avalanche of new constructions and protocols based on isogenies, and this tutorial will solely focus on the original Jao-De Feo SIDH protocol – the paper that triggered this avalanche. Furthermore, excellent surveys of the now broad field of isogeny-based cryptography already exist: the lecture notes of De Feo [4] and the Galbraith-Vercauteren [7] and Smith [13] surveys all give much more in-depth expositions. The hope is that potential newcomers to the field may dip their toes in here first, find (via an explicit toy example) that the isogeny waters are much less daunting than they seem, and then feel more comfortable diving into [4], [7], [13], or into the fast-expanding literature on the field. A recommended alternative starting point is Urbanik's friendly introduction [16].

[1] SIKE stands for supersingular isogeny key encapsulation, a variant of SIDH whose differences are mostly unimportant in this tutorial (further details are in Section 7).

2 The set of supersingular j-invariants

SIDH works in the quadratic extensions of large prime fields $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$, for which we typically choose the most convenient representation as $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ with $i^2 + 1 = 0$; elements are then of the form $u + vi$ where $u, v \in \mathbb{F}_p$.

Of the p^2 elements in $\mathbb{F}_{p^2}$, we are interested in a subset of size $\lfloor p/12 \rfloor + z$, where $z \in \{0, 1, 2\}$. The value of z depends on p mod 12 [12, Theorem V.4.1(c)], but it is unimportant; what is important to note is that as p grows exponentially large, so does the size of the subset we are interested in. This subset is precisely the set of supersingular j-invariants in $\mathbb{F}_{p^2}$ (we will describe what this terminology means in a moment).

For the purpose of being able to write this full set down, and in order to be able to carry out and visualise a mini SIDH protocol in full, herein we will focus on the toy example with

$$p := 431.$$

In this case there are $\lfloor p/12 \rfloor + 2 = 37$ supersingular j-invariants in $\mathbb{F}_{p^2}$, and they are depicted in Figure 1.

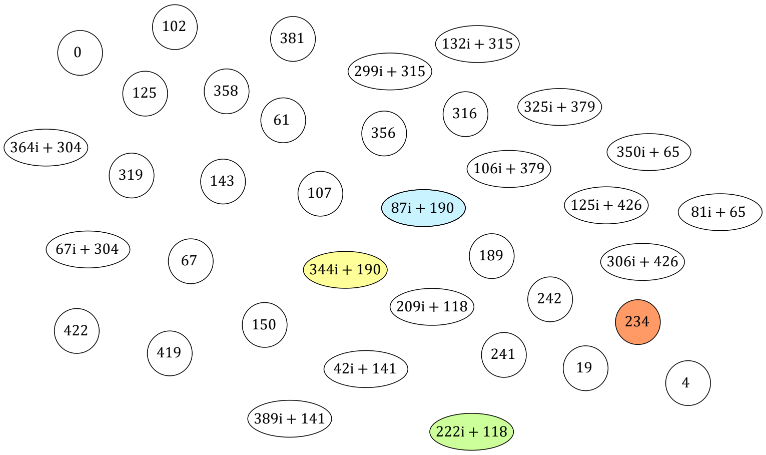

Fig. 1. The set of 37 supersingular j-invariants in $\mathbb{F}_{431^2}$.

Every elliptic curve has a unique j-invariant, and two elliptic curves are isomorphic to each other if and only if they have the same j-invariant. Each of the j-invariants in Figure 1 should therefore be thought of as representing any (and all) elliptic curve(s) with that j-invariant. For example, elliptic curves in

Montgomery form [11]

$$E_a \colon y^2 = x^3 + ax^2 + x$$

have j-invariant

$$j(E_a) = \frac{256(a^2-3)^3}{(a^2-4)}.$$

Taking $a_1 = 208i + 161$ and $a_2 = 172i + 162$ gives

$$j(E_{a_1}) = j(E_{a_2}) = 364i + 304,$$

so

$$E_{a_1} \colon y^2 = x^3 + (208i + 161)x^2 + x$$

and

$$E_{a_2} \colon y^2 = x^3 + (172i + 162)x^2 + x$$

are isomorphic, and both correspond to the leftmost j-invariant depicted in Figure 1. The isomorphisms are

$$\begin{aligned} \psi \colon \quad & E_{a_1} \to E_{a_2}, \\ & (x, y) \mapsto \big((66i + 182)x + (300i + 109), (122i + 159)y\big), \end{aligned}$$

and

$$\begin{aligned} \psi^{-1} \colon \quad & E_{a_2} \to E_{a_1}, \\ & (x, y) \mapsto \big((156i + 40)x + (304i + 202), (419i + 270)y\big). \end{aligned}$$

Write $\mathcal{O}_1$ and $\mathcal{O}_2$ for the identity elements (i.e. points at infinity) on E_{a_1} and E_{a_2}. Note that[2] $\psi(\mathcal{O}_1) = \mathcal{O}_2$ and $\psi^{-1}(\mathcal{O}_2) = \mathcal{O}_1$, and since the maps above do not have denominators, they are well-defined for all of the other (affine) points in $E_{a_1}(\mathbb{F}_{p^2})$ and $E_{a_2}(\mathbb{F}_{p^2})$. The composition of these two maps is the identity map on E_{a_1} or E_{a_2} (depending on the ordering).

An elliptic curve in characteristic p is either *supersingular*, or it is *ordinary.* In practice the supersingular case offers a number of advantages; instantiating efficient constructions is much easier, and the best known classical and quantum attacks against the related computational problems have exponential complexity[3].

The 37 j-invariants in Figure 1 are *all* of the supersingular j-invariants in characteristic p; supersingular curves always have j-invariants in $\mathbb{F}_{p^2}$ [12, Theorem V.3.1], so there are no more supersingular j-invariants to be found in higher extension fields.

[2] Those unfamiliar with projective space can take this at face value, while those in the know can substitute $x = X/Z$ and $y = y/Z$ to cast these equations into $\mathbb{P}^2$ and observe that $\psi((0\colon 1\colon 0) = (0\colon 1\colon 0)$, and vice versa.

[3] Note that this is the opposite of the situation for discrete logarithm-based ECC, where supersingular curves are avoided for security reasons. Discrete logarithms are no longer useful as hard underlying problems in the post-quantum setting, and they have no relevance to the security of SIDH.

SIDH in a nutshell. At this point it helps to see the high-level analogue between SIDH and the traditional Diffie-Hellman protocol in a generic, cyclic group G. Let g be the public generator with $\langle g \rangle = G$, and let Alice and Bob's respective public keys be g^a and g^b, so that g^{ab} is their shared secret. Referring back to Figure 1, the (blue) j-invariant $87i + 190$ is where Alice and Bob both begin; this is analogous to the generator g. In our example, Alice will choose a secret integer that, in turn, moves her around a subset of the 37 values until she arrives at the (green) j-invariant $222i + 118$; this will be the public key she sends to Bob, analogous to g^a. Bob will also choose a secret integer that moves him around some of the values in Figure 1, and he will eventually arrive at the yellow j-invariant $344i + 190$; he sends this as his public key to Alice, analogous to g^b. Together with Bob's public key and her secret integer, Alice then performs another sequence of moves to land at the (red) j-invariant 234; this acts as the analogue of g^{ab}, and it is the same j-invariant Bob will arrive at when he starts at Alice's public j-invariant and walks according to his secret integer.

The public keys contain additional information that is used to ensure that Alice and Bob can arrive at the same shared secret, but these details will come later. The purpose of the next section is to describe how both parties move between j-invariants: these moves are made with *isogenies*.

3 Isogenies

Just like the isomorphisms ψ and ψ^{-1} above, maps on a given elliptic curve, or between two elliptic curves, are written as $(x, y) \mapsto (f(x, y), g(x, y))$ for some functions f and g. The main reason Montgomery-form elliptic curves are often the preferred choice in both old-school ECC and in isogeny-based cryptography is that they facilitate very efficient x-only arithmetic, i.e. maps that ignore the y-coordinates entirely. In what follows we will also ignore the y-coordinates and simply write maps as

$$x \mapsto f(x),$$

but any reader wanting to complete the picture can recover the full maps by taking

$$(x, y) \mapsto (f(x), c \cdot y f'(x)),$$

where f' is the derivative of f and c is a fixed constant.

Consider the *multiplication-by-2* or *point doubling* map on a fixed Montgomery curve $E_a \colon y^2 = x^3 + ax^2 + x$, written as

$$[2]\colon \qquad E_a \to E_a, \qquad x \mapsto \frac{(x^2-1)^2}{4x(x^2+ax+1)}. \tag{1}$$

Observe that, unlike the isomorphisms in the previous section, the doubling map has a denominator that will create exceptional points. Viewing the curve equation, we see that these are the three points with $y = 0$, namely $(0, 0)$, $(\alpha, 0)$ and $(1/\alpha, 0)$, where $\alpha^2 + a\alpha + 1 = 0$. Indeed, these are the three points of order 2 on E_a, and together with the neutral element, $\mathcal{O}$, they are the entire kernel of the doubling map. This kernel forms a subgroup of the points in E_a, with group structure

$$\ker([2]) \cong \mathbb{Z}_2 \times \mathbb{Z}_2,$$

i.e. the *2-torsion* is precisely three cyclic subgroups of order 2. Each subgroup has one of the three points of exact order 2, together with the identity element $\mathcal{O}$. This is depicted in Figure 2.

Now consider the *multiplication-by-3* or *point tripling* map on $E_a \colon y^2 = x^3 + ax^2 + x$, written as

$$[3]\colon \qquad E_a \to E_a, \qquad x \mapsto \frac{x(x^4 - 6x^2 - 4ax^3 - 3)^2}{(3x^4 + 4ax^3 + 6x^2 - 1)^2} \tag{2}$$

Again, the denominator will give rise to exceptional points to the tripling map. Suppose its four roots are $\beta, \delta, \zeta, \theta$; each of these correspond to x-coordinates of points of order 3 in E_a, and this time there are two (non-zero) y-coordinates for each such x. Together with $\mathcal{O}$, there are then 9 points that are sent to $\mathcal{O}$ under [3], and this time we have

$$\ker([3]) \cong \mathbb{Z}_3 \times \mathbb{Z}_3,$$

i.e. the *3-torsion* is precisely four cyclic subgroups of order 3. This is depicted in Figure 3.

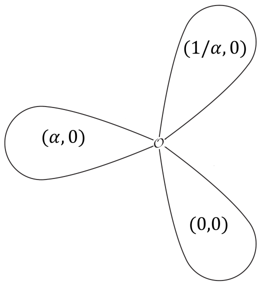

Fig. 2. The kernel $\ker([2]) \cong \mathbb{Z}_2 \times \mathbb{Z}_2$ of the doubling map; three cyclic subgroups of order 2.

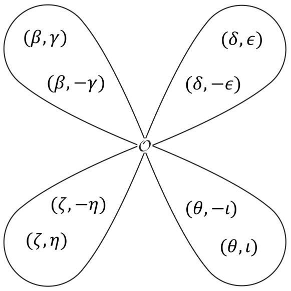

Fig. 3. The kernel $\ker([3]) \cong \mathbb{Z}_3 \times \mathbb{Z}_3$ of the tripling map; four cyclic subgroups of order 3.

It turns out that this pattern holds true for any ℓ where $p \nmid \ell$. The set of points in the ℓ-torsion, i.e. the set of points sent to $\mathcal{O}$ under the multiplication-by-ℓ map, is such that

$$\ker([\ell]) \cong \mathbb{Z}_\ell \times \mathbb{Z}_\ell,$$

forming $\ell + 1$ cyclic subgroups of order ℓ when ℓ is prime.

Both the doubling and tripling maps above are rather special cases of more general maps between elliptic curves that we call *isogenies*. In both of these instances, it just so happens that their domain and codomain are the same elliptic curve. In general, an isogeny is a map

$$\phi\colon E \to E'$$

from one elliptic curve to another. Isogenies are said to be either *separable* or *inseparable*; these definitions are not important here, but it should be said that we will only be interested in the former case.

A key fact to digest is that separable isogenies are in one-to-one correspondence with finite subgroups: every subgroup G of points on an elliptic curve E gives rise to a unique isogeny $\phi\colon E \to E'$ whose kernel is G, and vice versa. In this case we often see the codomain being written as E/G. Vélu's formulas [18] make this explicit: on input of (the curve constants defining) E and the points in G, these formulas output the constants defining $E' = E/G$ and the explicit maps for ϕ, i.e. the maps that move any points on E (except those in the kernel G) to their corresponding image on E'. Writing the general form of Vélu's formulas down is unnecessary here, as we will only require two instances. Nevertheless, it is worth noting that these formulas are simply rational functions of the inputs mentioned above, and the degrees of these functions are the same size as the size of the subgroup G.

Referring back to Figure 2, suppose we set $G = \{\mathcal{O}, (\alpha, 0), (1/\alpha, 0), (0, 0)\}$ and input it into Vélu's formulas, together with the curve E_a. The output would then be the unique map with kernel G, i.e. the doubling map in (1), together with E_a itself.

Suppose we instead choose our kernel to be one of the cyclic subgroups of order 2, e.g. set $G = \{\mathcal{O}, (\alpha, 0)\}$, and input this and E_a into Vélu's formulas. In this case the map we get is

$$\phi\colon \qquad E_a \to E_{a'}, \qquad x \mapsto \frac{x(\alpha x - 1)}{x - \alpha},$$

with

$$a' = 2(1 - 2\alpha^2). \tag{3}$$

This map can be used to compute 2-isogenies on any Montgomery curve. For example, recall one of the curves from Section 2 as

$$E_a\colon y^2 = x^3 + (208i + 161)x^2 + x, \qquad \text{with} \qquad j(E_a) = 364i + 304.$$

The point $(\alpha, 0) \in E_a$ with $\alpha = 350i + 68$ has order 2. Applying (3) yields the image curve

$$E_{a'}\colon y^2 = x^3 + (102i + 423)x^2 + x, \qquad \text{with} \qquad j(E_{a'}) = 344i + 190,$$

and the map

$$\phi\colon x \mapsto \frac{x((350i + 68)x - 1)}{x - (350i + 68)} \tag{4}$$

that will take (the x-coordinate of) any point not in $\{\mathcal{O}, (\alpha, 0)\}$ to the (x-coordinate of the) corresponding point on $E_{a'}$.

Now suppose we set the kernel as one of the subgroups of order 3 in Figure 3, e.g. $G = \{\mathcal{O}, (\beta, \gamma), (\beta, -\gamma)\}$. Vélu's formulas output

$$\phi\colon \qquad E_a \to E_{a'}, \qquad x \mapsto \frac{x(\beta x - 1)^2}{(x - \beta)^2},$$

with

$$a' = (a\beta - 6\beta^2 + 6)\beta \tag{5}$$

The point $(\beta, \gamma) = (321i + 56, 303i + 174)$ has order 3 on $E_a\colon y^2 = x^3 + (208i + 161)x^2 + x$. Applying (5) yields the image curve

$$E_{a'}\colon y^2 = x^3 + 415x^2 + x, \qquad \text{with} \qquad j(E_{a'}) = 189,$$

and the map

$$\phi\colon x \mapsto \frac{x((321i + 56)x - 1)^2}{(x - (321i + 56))^2}$$

that will move points from E_a to $E_{a'}$.

Unlike the isomorphisms in Section 2 which preserve the j-invariant, the isogenies in this section give image curves with different j-invariants; in this case the curves are no longer isomorphic, but are instead said to be *isogenous.*

The *degree* of a non-zero separable isogeny is the number of elements in its kernel [12, Theorem III.4.10], and it is also the degree of the isogeny as a rational map (in the sense of [12, p. 21]). For our purposes, one can see from eyeballing any of the examples above that the number of kernel elements match the degree of the corresponding map.

Isomorphisms are actually a special case of an isogeny where the kernel is trivial, i.e. the kernel is just $\{\mathcal{O}\}$ (as we saw in Section 2), so they are isogenies of degree 1. By definition, composing an isomorphism with its inverse gives the identity map. In general, however, isogenies do not have an inverse that behaves like this; instead, every isogeny has a unique *dual isogeny* [12, Theorem III.6.1], that almost behaves like an inverse. If $\phi : E \to E'$ is an isogeny of degree d, then the dual isogeny $\hat{\phi}$ is such that the composition $(\hat{\phi} \circ \phi) = [d]$, i.e. the multiplication-by-d map on E, and the composition $(\phi \circ \hat{\phi})$ is the multiplication-by-d map on E'. Just like inverse isomorphisms, the composition of dual isogenies lands us back on the same curve; the difference is that the kernel of this composition becomes the d-torsion in general (which is non-trivial when $d > 1$). As an example, composing the degree-2 isogeny $\phi : E_a \to E_{a'}$ in (3) with its degree-2 dual $\hat{\phi} : E_{a'} \to E_a$ gives the degree-4 doubling map in (1) (whose kernel is in Figure 2).

It is important to note that isogenies are well-behaved maps (the fancy word is morphisms [12, p. 12]) in both the geometric *and* algebraic sense. Our focus above has mostly been on the former side, where we have seen them map points between two geometric curves. But isogenies are also algebraic in that they are *group homomorphisms*: an isogeny $\phi\colon E \to E'$ satisfies

$$\phi(P+Q) = \phi(P) + \phi(Q)$$

for all $P, Q \in E$ [12, Theorem III.4.8]; the sum on the left corresponds to the elliptic curve group law on E, while the sum on the right is the group law on E'.

In terms of SIDH, it is helpful to see what this homomorphic behaviour means with some examples. Returning to the 2-isogeny in (4) from

$$E_a\colon y^2 = x^3 + (208i + 161)x^2 + x \qquad \text{to} \qquad E_{a'}\colon y^2 = x^3 + (102i + 423)x^2 + x,$$

recall that the kernel of ϕ was $\{\mathcal{O}, (\alpha, 0)\}$ with $\alpha = 350i + 68$.

We will now observe what becomes of various points on E as they move through ϕ, momentarily returning to both coordinates under the map $\phi\colon (x, y) \mapsto (f(x), c \cdot yf'(x))$, with $f(x)$ as above and with the constant c satisfying $c^2 = \alpha$. The points

$$P = (390i + 23, 104i + 7) \qquad \text{and} \qquad Q = (151i + 140, 110i + 136)$$

both have order 8 on E_a, but when ϕ is the 2-isogeny above, the points

$$\phi(P) = (23i + 231, 309i + 61) \qquad \text{and} \qquad \phi(Q) = (80i + 261, 192i + 259)$$

have orders 4 and 8 on $E_{a'}$, respectively. The reason the order of P decreased is because it *lies above* a non-trivial element in the kernel: $[4]P = (\alpha, 0) \in \ker(\phi)$, thus $\phi([4]P) = \mathcal{O}$ on $E_{a'}$, and since ϕ is a homomorphism, $\phi([4]P) = [4]\phi(P)$; given that $[2]P \notin \ker(\phi)$, it must therefore be that $\phi(P)$ has order 4. The same reasoning shows that, in general, a degree-d isogeny will decrease the order of any point lying above the kernel by a factor of d. On the other hand, other points preserve their orders when moving through isogenies, as we saw above for the point Q. As another example, the image of the point $R = (\beta, \gamma) = (321i{+}56, 303i{+}174)$ of order 3 on E_a is the point $\phi(R) = (102i{+}238, 346i{+}193)$ of order 3 on $E_{a'}$. In general, evaluating a degree-d isogeny at a point of order ℓ will preserve this order if d and ℓ are coprime; it is useful to keep this property in mind in the sections that follow.

A theorem of Tate [14] says that two elliptic curves $E/\mathbb{F}_q$ and $E'/\mathbb{F}_q$ are isogenous over $\mathbb{F}_q$ if and only if they have the same number of points over $\mathbb{F}_q$. Returning to our running example, all of the j-invariants in Figure 1 correspond to elliptic curves $E/\mathbb{F}_{431^2}$ with group orders $\#E(\mathbb{F}_{431^2}) = 432^2$. Moreover, they all have the same group structure $\mathbb{Z}_{432} \times \mathbb{Z}_{432}$. This is an instance of the general case in which supersingular curves $E/\mathbb{F}_{p^2}$ always have their full rational $(p-1)$- or $(p+1)$-torsion defined over $\mathbb{F}_{p^2}$. Moreover, in our case the elliptic curve group is precisely the $(p+1)$-torsion and, as we saw at the beginning of the section, we have $\ker([p+1]) \cong \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$, from which it follows that

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

Suppose $\phi\colon E \to E'$ is an $\mathbb{F}_q$-rational isogeny of degree $d > 1$, i.e. $\#\ker(\phi) = d$. One point of confusion that can sometimes arise for isogeny newcomers is how $E/\mathbb{F}_q$ and $E'/\mathbb{F}_q$ can have the same number of points. There are d elements mapped to $\mathcal{O}$ under ϕ, meaning there are $\#E(\mathbb{F}_q) - d$ elements in $E(\mathbb{F}_q)$ which are carried through to non-zero points in $E'(\mathbb{F}_q)$. If $d > 1$, then at first sight it can appear that the group orders should not match. However, the unbalance that seems to arise is resolved by points in higher extension fields that map down to $E'(\mathbb{F}_q)$ under ϕ, and the same thing happens in the reverse direction when considering the dual isogeny from E' back to E.

The *composition* of isogenies is as we might expect. Composing the two isogenies

$$\phi\colon E \to E' \qquad \text{and} \qquad \psi\colon E' \to E''$$

gives the isogeny

$$(\psi \circ \phi)\colon E \to E'',$$

whose degree is the product of the two individual degrees. We have already seen a special example of this above: when ϕ is the degree-2 isogeny in (3), then composing with the degree-2 dual isogeny $\hat{\phi}$ gave the degree-4 doubling map in (1).

As we will get a glimpse of in Section 5, this notion of composition is crucial to the practicality of SIDH. In real-world instantiations, we compute isogenies whose degrees are exponentially large, e.g. isogenies of degree 2^e for $e > 200$. Given that algorithms for general isogeny computation are linear in the degree of the isogeny, this would be out of the question if it was not for our being able to instead compute it as the composition of e individual 2-isogenies.

Finally, when speaking of isogenies, we are often implicitly speaking *up to isomorphism.* For example, in the fact we stated above whereby every subgroup of points gives rise to a unique isogeny, it would be more precise to state that this isogeny is unique up to isomorphism. We could always compose an isogeny with an isomorphism to get a different looking map, but for all intents and purposes these isogenies will be considered equivalent.

4 Isogeny graphs

Recall that, for each prime p, we are working with the set of all j-invariants in $\mathbb{F}_{p^2}$ that correspond to supersingular curves in characteristic p. We will continue with our example of $p = 431$, for which the 37 j-invariants are depicted back in Figure 1. The important notion illustrated in this section is that, when we introduce any other prime $\ell \neq p$, this set becomes a *graph* with surprising properties. The vertices of each graph remain fixed as the j-invariants themselves, but the edges between them correspond to ℓ-isogenies, and therefore the edges change for each ℓ. In SIDH we only need two of the graphs: one for Alice, for which we usually take $\ell = 2$, and one for Bob, for which we usually take $\ell = 3$ (these choices of the two smallest primes currently give the most efficient instantiation of SIDH).

To draw the graphs for our example, we proceed as in the previous section. Recall that the 2-isogeny in (4) was from the curve $E_a\colon y^2 = x^3+(208i+161)x^2+x$ with $j(E_a) = 364i + 304$ to the curve $E_{a'}\colon y^2 = x^3 + (102i + 423)x^2 + x$ with $j(E_{a'}) = 344i+190$; the kernel was generated by $(\alpha, 0) \in E_a$ with $\alpha = 350i+68$. Thus, we can draw an edge between these two j-invariants. Furthermore, recall (see Figure 2) that there are two other kernels of 2-isogenies on E_a: one generated by $(1/\alpha, 0)$ and the other generated by $(0, 0)$. These produce two more edges, one connecting $j = 364i + 304$ to $j = 67$ and one connecting $j = 364i + 304$ to $j = 319$. Continuing in this fashion, we visit every j-invariant in the graph until all three 2-torsion points have been used to generate three outgoing edges, and eventually we produce the full 2-isogeny graph that is depicted in Figure 4.

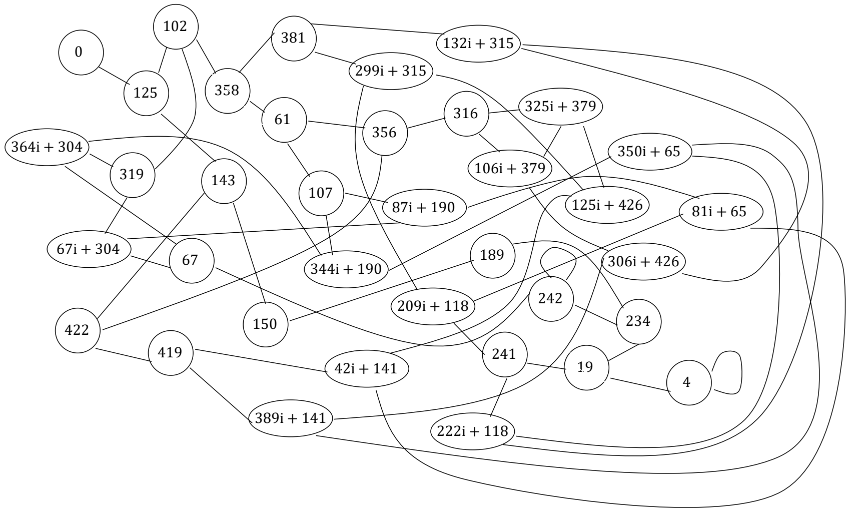

Fig. 4. The 2-isogeny graph for $p = 431$. The 37 nodes are the supersingular j-invariants and the edges between them correspond to 2-isogenies.

Observe that, for all $j \notin \{0, 4, 242\}$ (we will talk about these exceptions in Section 6), there are exactly 3 edges connecting a given node to other 2-isogenous j-invariants. Moreover, we have not written any directions on the arrows; the reasoning here is that, for any edge from $j(E)$ to $j(E')$ corresponding to an isogeny $\phi\colon E \to E'$, the dual isogeny gives an edge from $j(E')$ back to $j(E)$.

The 3-isogeny graph is drawn analogously, but recall (see Figure 3) that there are now four outgoing edges corresponding to every j-invariant (we again have a small number of exceptions for $j \in \{0, 4, 125, 242\}$). We saw explicitly in Section 3 that the point $(\beta, \gamma) = (321i + 56, 303i + 174)$ of order 3 on E_a (as above) was the kernel of an isogeny to the curve $E_{a'}\colon y^2 = x^3 + 415x^2 + x$ with $j(E_{a'}) = 189$. Inputting the three other 3-torsion subgroups into Vélu's formulas gives three image curves with $j = 19$, $j = 42i + 141$, and $j = 106i + 379$. Again, working through each node gives rise to the graph depicted in Figure 5.

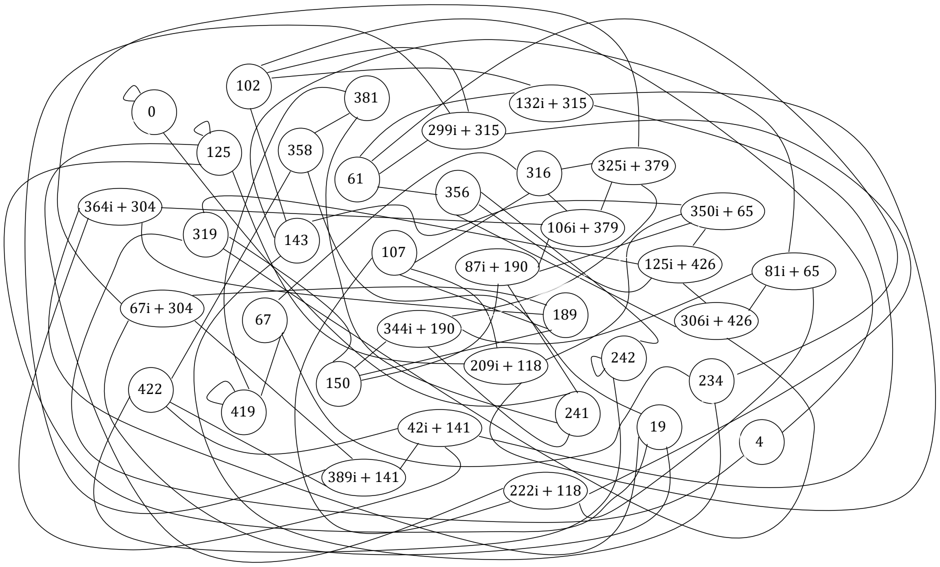

Fig. 5. The 3-isogeny graph for $p = 431$. The 37 nodes are the supersingular j-invariants and the edges between them correspond to 3-isogenies.

We are now in a position to discuss how SIDH primes are chosen. State-of-the-art instantiations of SIDH (including those in the SIKE specification [8]) always fix primes p of the form

$$p = 2^{e_A}3^{e_B} - 1,$$

where e_A and e_B are such that $2^{e_A} \approx 3^{e_B}$. In fact, SIDH actually allows more flexibility by introducing a cofactor f and permitting primes of the form $p = f \cdot 2^{e_A}3^{e_B} - 1$, but (as it currently stands) the case of $f = 1$ is flexible enough to find suitable primes at all of the interesting security levels.

The rationale behind choosing primes of this form ties back to the discussion in the previous section; we work with supersingular curves for which the elliptic curve group $E(\mathbb{F}_{p^2})$ is the full $(p+1)$-torsion, that which is isomorphic to $\mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$. Thus, for our choice of primes, the elliptic curve group is

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{(2^{e_A}3^{e_B})} \times \mathbb{Z}_{(2^{e_A}3^{e_B})}.$$

In other words, there are two points P and Q, both of order $2^{e_A}3^{e_B}$, that are a basis for the full elliptic curve group $E(\mathbb{F}_{p^2})$. Moreover, linear combinations $[\alpha]P + [\beta]Q$ with $\alpha, \beta \in \mathbb{Z}_{(2^{e_A}3^{e_B})}$ can be used to generate *all* of the points whose orders are *any* factor of $2^{e_A}3^{e_B}$. In particular, every point of order 2^{e_A} or of order 3^{e_B} lies in $E(\mathbb{F}_{p^2})$. We will see at the beginning of the next section that these are precisely the points that are used as secret generators of subgroups (i.e. isogenies) in the SIDH framework, so choosing primes in the above fashion ultimately means that Alice and Bob will only ever have to work with points inside $E(\mathbb{F}_{p^2})$.

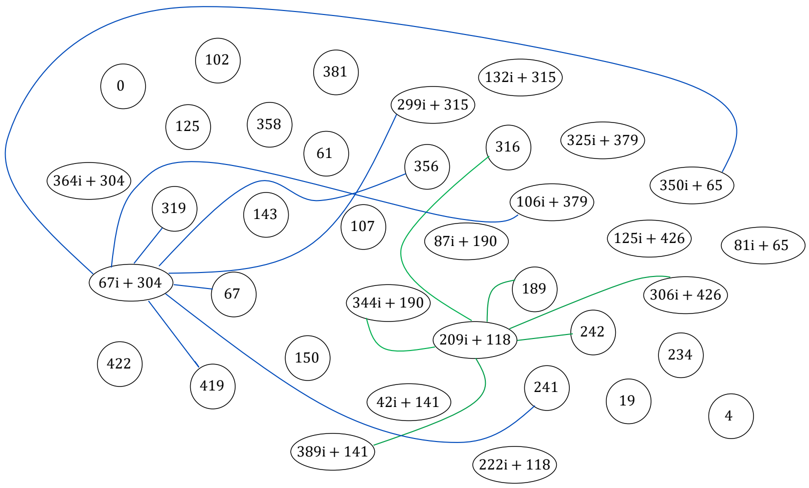

Fig. 6. The 6 edges at the node $j = 209i + 118$ corresponding to the 5-isogeny graph, and the 8 edges at the node $67i + 304$ corresponding to the 7-isogeny graph.

It is worth revisiting the fact that, although we only need to consider the graphs corresponding to $\ell = 2$ and $\ell = 3$, isogeny graphs exist for every prime ℓ for which $p \nmid \ell$. In Figure 6 we give the 6 edges corresponding to $\ell = 5$ at $j = 209i + 118$, and the 8 edges corresponding to $\ell = 7$ at $j = 67i + 304$ (these could be done at every node, but Figure 5 was already messy enough!). Although the j-invariants connected by these edges are always in $\mathbb{F}_{p^2}$, the isogenies corresponding to $\ell \notin \{2, 3\}$ would be computed using points that are not $\mathbb{F}_{p^2}$-rational. In other words, points of order ℓ^e for $\ell \notin \{2, 3\}$ are not found in $E(\mathbb{F}_{p^2})$ (with our prime p chosen as above), so to compute these isogenies we would have to perform computations in the larger extension fields of $\mathbb{F}_p$ where these points are found.

We will further discuss the properties of cryptographically sized supersingular isogeny graphs in Section 7.

5 Walking through the protocol

This section walks step by step through the entire toy example of the SIDH protocol. During both Alice and Bob's public key generation, every computation is provided explicitly so active readers (perhaps equipped with their favourite computer algebra package) can follow along; on the other hand, readers finding the repetition tedious can happily skip the details. The shared secret computations are analogous, so less details are given there. We start with a high-level and general description of the protocol, before returning to our concrete example.

A high-level overview. For the reasons detailed in Section 4, SIDH instantiations typically take $p = 2^{e_A}3^{e_B} - 1$ with $2^{e_A} \approx 3^{e_B}$. The protocol begins on an initial curve E in the corresponding supersingular isogeny graph.

Alice's secrets will be isogenies of degree (or, equivalently, subgroups of order) 2^{e_A}, and Bob's will be of degree 3^{e_B}. Recall from Section 3 that the ℓ-torsion is 2-dimensional, so Alice will compute generators of her secret subgroups by computing secret linear combinations of two public basis points, where

$$\langle P_A, Q_A \rangle = E[2^{e_A}] \cong \mathbb{Z}_{2^{e_A}} \times \mathbb{Z}_{2^{e_A}}.$$

Similarly, Bob will compute his secret generators as secret linear combinations of two public points, P_B and Q_B, where

$$\langle P_B, Q_B \rangle = E[3^{e_B}] \cong \mathbb{Z}_{3^{e_B}} \times \mathbb{Z}_{3^{e_B}}.$$

This necessarily means P_A and Q_A both have order 2^{e_A} and P_B and Q_B both have order 3^{e_B}. In practice (e.g. in SIKE [8]), they will typically choose their secret generator points, S_A and S_B, by simply taking

$$S_A = P_A + [k_A]Q_A \quad \text{with} \quad k_A \in [0, 2^{e_A}),$$

and

$$S_B = P_B + [k_B]Q_B \quad \text{with} \quad k_B \in [0, 3^{e_B}).$$

To compute her public key, Alice chooses $k_A \in \{0, 1, \ldots 2^{e_A} - 1\}$, computes S_A as above, and then computes the secret isogeny $\phi_A \colon E \to E_A$, where $E_A = E/\langle S_A \rangle$. She does this by composing e_A isogenies of degree 2, i.e. taking e_A steps in the 2-isogeny graph, which are defined by S_A. Her public key is then

$$\mathrm{PK}_A = (E_A, P'_B, Q'_B) = (\phi_A(E), \phi_A(P_B), \phi_A(Q_B)),$$

where the first element is the image *curve* $E_A = \phi_A(E)$, while the second and third elements are the images of Bob's public basis *points* (we will discuss why these points are needed in a moment).

Bob chooses $k_B \in \{0, 1, \ldots 3^{e_B} - 1\}$, computes S_B as above, and then computes his secret isogeny $\phi_B \colon E \to E_B$, where $E_B = E/\langle S_B \rangle$. He does this by composing e_B isogenies of degree 3, i.e. taking e_B steps in the 3-isogeny graph, which are defined by S_B. His public key is then

$$\mathrm{PK}_B = (E_B, P'_A, Q'_A) = (\phi_B(E), \phi_B(P_A), \phi_B(Q_A)).$$

On input of her secret integer k_A and Bob's public key, Alice computes the secret subgroup $S'_A = P'_A + [k_A]Q'_A$ on E_B, and then computes another secret isogeny $\phi'_A \colon E_B \to E_{AB}$, where $E_{AB} = E_B/\langle S'_A \rangle$. She then computes the shared secret as $j_{AB} = j(E_{AB})$.

On Bob's side, he takes $S'_B = P'_B + [k_B]Q'_B$ on E_A, computes the secret isogeny $\phi'_B \colon E_A \to E_{BA}$, where $E_{BA} = E_A/\langle S'_B \rangle$. He then computes the shared secret as $j_{BA} = j(E_{BA})$.

Why the image points? Before proceeding, it is important to discuss why Alice and Bob must move each other's basis points through their secret isogenies and include these images in their public keys. Unlike traditional Diffie-Hellman (see the end of Section 2) where exponents commute to give $(g^a)^b = (g^b)^a$, in the SIDH setting we do not have this property. Indeed, it does not even make sense to consider a composition of the isogenies $\phi_A \colon E \to E_A$ and $\phi_B \colon E \to E_B$, given their domains/codomains. Instead, Alice needs a way to describe an isogeny whose domain is E_B, and Bob needs an analogous way to start from E_A. Moving each other's basis points through the secret isogeny during key generation solves this problem; it allows both parties to essentially redo the same computations on the curves transmitted in the public keys and to ultimately arrive at the same j-invariant[4].

Public parameters. We now continue with our example of $p = 2^4 3^3 - 1$, and take the public starting curve as

$$E_{a_0} \colon y^2 = x^3 + a_0 x^2 + x, \quad \text{with} \quad a_0 = 329i + 423 \quad \text{and} \quad j(E_{a_0}) = 87i + 190.$$

We can take any four public basis points we like E_{a_0}, so long as $E[2^4] = \langle P_A, Q_A \rangle$ and $E[3^3] = \langle P_B, Q_B \rangle$. We fix

$$P_A := (100i + 248, 304i + 199) \quad \text{and} \quad Q_A := (426i + 394, 51i + 79),$$

together with

$$P_B := (358i + 275, 410i + 104) \quad \text{and} \quad Q_B := (20i + 185, 281i + 239).$$

Alice's public key generation. Suppose Alice chooses the secret

$$k_A := 11$$

from $\{0, 1, \ldots 15\}$. Her first step is to compute the secret generator corresponding to k_A, as

$$\begin{aligned} S_A &= P_A + [k_A]Q_A \\ &= (100i + 248, 304i + 199) + [11](426i + 394, 51i + 79) \\ &= (271i + 79, 153i + 430), \end{aligned}$$

[4] Astute readers wanting to prove that $j_{AB} = j_{BA}$ can argue that both j-invariants correspond to the isomorphism class of $E/\langle S_A, S_B \rangle$ by using the identity $E/\langle P, Q \rangle \cong (E/\langle P \rangle)/\langle \phi(Q) \rangle$ with $\phi \colon E \to E/\langle P \rangle$. Otherwise, see [9, §3].

which is a point of order 16 on the starting curve E_{a_0}. Alice now proceeds to compute her public key using only a combination of the point doubling operation in (1) and the 2-isogeny operation in (3). Below we will use prime superscripts to denote values that are updated/overwritten throughout the procedure, and again, though we refer to the x-only maps in Section 3, we will still write the points in full under the extension from $x \mapsto f(x)$ to $(x, y) \mapsto (f(x), cyf'(x))$ (where $c^2 = \alpha$ and $(\alpha, 0)$ is the kernel of the 2-isogeny at hand).

- *Compute* ϕ_0. Initialise $S'_A = S_A = (271i + 79, 153i + 430)$. Three repeated applications of the doubling in (1) produces the point $R'_A = [8]S'_A = (18i + 37, 0)$, which has order 2 on E_{a_0}. Inputting R'_A into (3) gives $\phi_0 \colon E_{a_0} \to E_{a_1}$, with $a_1 = 275i + 132$ and $j(E_{a_1}) = 107$. It also gives the map $\phi_0 \colon x \mapsto \frac{x((18i+37)x-1)}{x-(18i+37)}$, which is used to update $P'_B = \phi(P'_B) = (118i+85, 274i+150)$, $Q'_B = (62i+124, 64i+269)$, and $S'_A = (36i+111, 175i+67)$; the orders of P'_B and Q'_B on E_{a_1} are unchanged, but the order of the new S'_A has decreased from 16 to 8. Figure 7 shows ϕ_0 as Alice's first step in the 2-isogeny graph.

- *Compute* ϕ_1. Two repeated applications of the doubling in (1) produces the point $R'_A = [4]S'_A = (7i + 49, 0)$, which has order 2 on E_{a_1}. Inputting R'_A into (3) gives $\phi_1 \colon E_{a_1} \to E_{a_2}$, with $a_2 = 273i + 76$ and $j(E_{a_2}) = 344i + 190$. It also gives the map $\phi_1 \colon x \mapsto \frac{x((7i+49)x-1)}{x-(7i+49)}$, which is used to update $P'_B = \phi(P'_B) = (274i + 251, 316i + 59)$, $Q'_B = (214i + 94, 354i + 193)$, and $S'_A = (274i + 374, 84i + 77)$; the order of the new S'_A has decreased from 8 to 4. Figure 7 shows ϕ_1 as Alice's second step in the 2-isogeny graph.

- *Compute* ϕ_2. One doubling via (1) produces the point $R'_A = [2]S'_A = (245i + 27, 0)$, which has order 2 on E_{a_2}. Inputting R'_A into (3) gives $\phi_2 \colon E_{a_2} \to E_{a_3}$, with $a_3 = 93i + 136$ and $j(E_{a_3}) = 350i + 65$. It also gives the map $\phi_2 \colon x \mapsto \frac{x((245i+27)x-1)}{x-(245i+27)}$, which is used to update $P'_B = \phi(P'_B) = (77i + 209, 75i + 79)$, $Q'_B = (339i + 356, 12i + 419)$, and $S'_A = (227i + 150, 0)$; the order of the new S'_A has decreased from 4 to 2. Figure 7 shows ϕ_2 as Alice's third step in the 2-isogeny graph.

- *Compute* ϕ_3. $S'_A = (227i + 150, 0)$ already has order 2 on E_{a_3}, so no scalar multiplication is necessary in the final stage. Inputting S'_A into (3) gives $\phi_3 \colon E_{a_3} \to E_{a_4}$, with $a_4 = 423i + 179$ and $j(E_{a_4}) = 222i + 118$. It also gives the map $\phi_3 \colon x \mapsto \frac{x((227i+150)x-1)}{x-(227i+150)}$, which is used to update $P'_B = \phi(P'_B) = (142i + 183, 119i + 360)$, $Q'_B = (220i + 314, 289i + 10)$; the point S'_A is in the kernel and is not carried through. Figure 7 shows ϕ_3 as Alice's fourth and final step in the 2-isogeny graph.

Alice's secret 2^4-isogeny is the composition of the four 2-isogenies detailed above, i.e. $\phi_A \colon E_{a_0} \to E_{a_4}$, $Q \mapsto \phi_A(Q)$, with $\phi_A = (\phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$ - see Figure 7. Her public key, PK_A, is then

$$\begin{aligned} \mathrm{PK}_A &= (\phi_A(E_{a_0}), \phi_A(P_B), \phi_A(Q_B)) \\ &= (423i + 179, (142i + 183, 119i + 360), (220i + 314, 289i + 10)). \end{aligned} \tag{6}$$

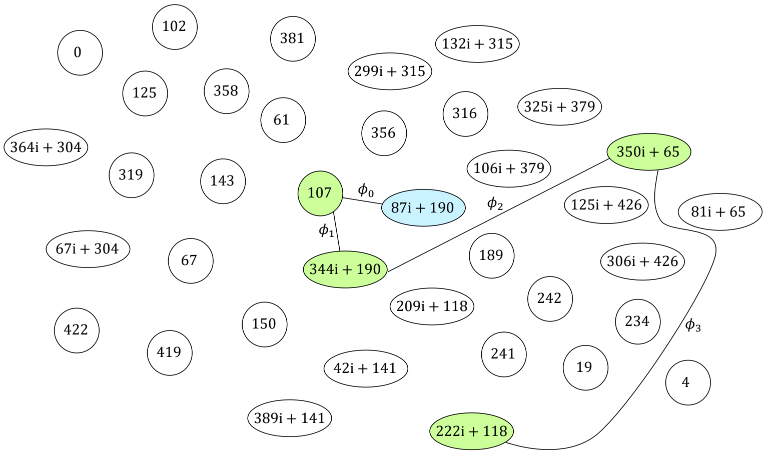

Fig. 7. Alice's key generation. She starts at the public curve corresponding to $j = 87i + 190$, her secret key is the isogeny $\phi_A = (\phi_3 \circ \phi_2 \circ \phi_1 \circ \phi_0)$, and the destination node $222i + 118$ becomes part of her public key.

Rather than send the j-invariant $j(E_{a_4}) \in \mathbb{F}_{p^2}$ as the value defining the curve, Alice can simply send $a_4 \in \mathbb{F}_{p^4}$, and Bob will do the same. The j-invariant function is only used during the shared secret computation to guarantee that Alice and Bob arrive at the same value.

Bob's public key generation. Suppose Bob chooses the secret

$$k_B := 2$$

from $\{0, 1, \ldots, 26\}$. His first step is to compute the secret generator corresponding to k_B, as

$$\begin{aligned} S_B &= P_B + [k_B]Q_B \\ &= (358i + 275, 410i + 104) + [2](20i + 185, 281i + 239) \\ &= (122i + 309, 291i + 374), \end{aligned}$$

which is a point of order 27 on E_{a_0}.

Bob proceeds to compute his public key using only a combination of the point tripling operation in (2) and the 3-isogeny operation in (5). Below we will use the same conventions as we did for Alice, overriding the notations for E_{a_i} and ϕ_i to save additional subscripts.

- *Compute* ϕ_0. Initialise $S'_B = S_B = (122i + 309, 291i + 374)$. Two repeated applications of the tripling in (2) produces the point $R'_B = [9]S'_B = (23i +$

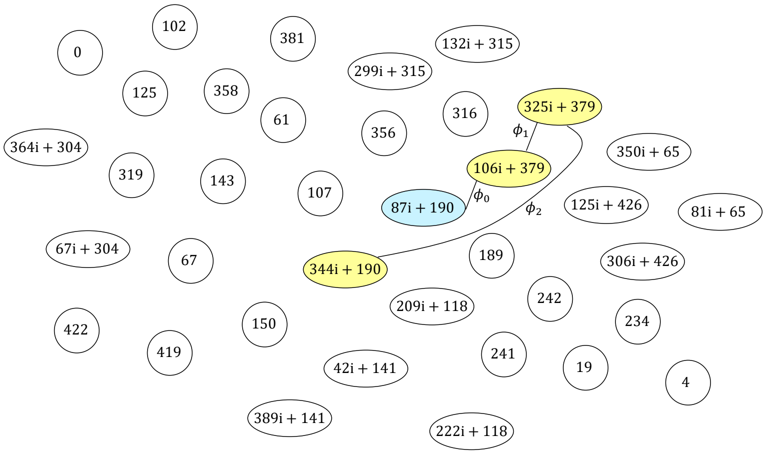

Fig. 8. Bob's key generation. He starts at the public curve corresponding to $j = 87i + 190$, his secret key is the isogeny $\phi_B = (\phi_2 \circ \phi_1 \circ \phi_0)$, and the destination node $344i+190$ becomes part of his public key.

$37, 4i+302)$, which has order 3 on E_{a_0}. Inputting R'_B into (5) gives $\phi_0\colon E_{a_0} \to E_{a_1}$, with $a_1 = 134i + 2$ and $j(E_{a_1}) = 106i + 379$. It also gives the map $\phi_0\colon x \mapsto \frac{x((23i+37)x-1)^2}{(x-(23i+37))^2}$, which is used to update $P'_A = \phi(P'_A) = (418i + 155, 288i+331)$, $Q'_A = (159i+242 : 310i+425)$, and $S'_B = (295i+256, 253i+64)$; the orders of P'_A and Q'_A on E_{a_1} are unchanged, but the order of the new S'_B has decreased from 27 to 9. Figure 8 shows ϕ_0 as Bob's first step in the 3-isogeny graph.

- *Compute* ϕ_1. One tripling via (2) produces the point $R'_B = [3]S'_B = ((98i + 36, 56i+155)$, which has order 3 on E_{a_1}. Inputting R'_B into (5) gives $\phi_1\colon E_{a_1} \to E_{a_2}$, with $a_2 = 117i + 54$ and $j(E_{a_2}) = 325i + 379$. It also gives the map $\phi_1\colon x \mapsto \frac{x((98i+36)x-1)^2}{(x-(98i+36))^2}$, which is used to update $P'_A = \phi(P'_A) = (252i + 425, 315i + 19)$, $Q'_A = (412i + 81 : 111i + 172)$, and $S'_B = (102i + 405, 375i+313)$; the order of the new S'_B has decreased from 9 to 3. Figure 8 shows ϕ_1 as Bob's second step in the 3-isogeny graph.

- *Compute* ϕ_2. $S'_B = (102i+405, 375i+313)$ already has order 3 on E_{a_2}, so no scalar multiplication is needed. Inputting R'_B into (5) gives $\phi_2\colon E_{a_2} \to E_{a_3}$, with $a_2 = 273i + 76$ and $j(E_{a_2}) = 344i + 190$. It also gives the map $\phi_1\colon x \mapsto \frac{x((98i+36)x-1)^2}{(x-(98i+36))^2}$, which is used to update $P'_A = \phi(P'_A) = (187i+226, 43i+360)$, $Q'_A = (325i + 415, 322i + 254)$, and $S'_B = (102i + 405, 375i + 313)$; the point S'_B is in the kernel and is not carried through. Figure 8 shows ϕ_2 as Bob's third and final step in the 3-isogeny graph.

Bob's secret 3^3-isogeny is the composition of the three 3-isogenies detailed above, i.e. $\phi_B \colon E_{a_0} \to E_{a_3}$, $Q \mapsto \phi_B(Q)$, with $\phi_B = (\phi_2 \circ \phi_1 \circ \phi_0)$ - see Figure 8. His public key, PK_B, is then

$$\begin{aligned}\mathrm{PK}_B &= (\phi_B(E_{a_0}), \phi_B(P_A), \phi_B(Q_A)) \\ &= (273i + 76, (187i + 226, 43i + 360), (325i + 415, 322i + 254)).\end{aligned} \tag{7}$$

Alice's shared secret computation. Alice starts from the curve output in Bob's public key in (7), taking her new starting curve E_{a_0} with $a_0 = 273i + 76$. Her first step is again to compute a secret generator on E_{a_0} corresponding to her secret k_A, as

$$\begin{aligned}S_A &= \phi_B(P_A) + [k_A]\phi_B(Q_A) \\ &= (187i + 226, 43i + 360) + [11](325i + 415, 322i + 254) \\ &= (125i + 357, 415i + 249).\end{aligned}$$

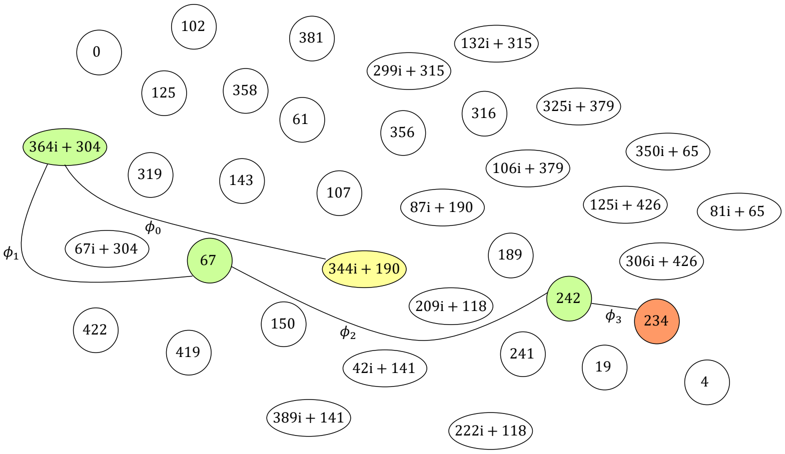

Fig. 9. Alice's shared secret computation. She starts at the curve from Bob's public key with $j = 344i + 190$, and uses her secret key to compute the walk to the node with $j = 234$, which is the shared secret.

She now proceeds exactly as before, performing the analogous sequence of operations as during key generation. The only difference is that she no longer needs to move any basis points through the isogeny, saving some computation because it is only the destination curve that she needs. The four 2-isogeny computations are summarised as

$$\begin{aligned}
&\phi_0\colon E_{a_0} \to E_{a_1}, &&\text{with} \quad a_1 = 289i + 341 &&\text{and} \quad j(E_{a_1}) = 364i + 304;\\
&\phi_1\colon E_{a_1} \to E_{a_2}, &&\text{with} \quad a_2 = 414i + 428 &&\text{and} \quad j(E_{a_2}) = 67;\\
&\phi_2\colon E_{a_2} \to E_{a_3}, &&\text{with} \quad a_3 = 246i &&\text{and} \quad j(E_{a_3}) = 242;\\
&\phi_3\colon E_{a_3} \to E_{a_4}, &&\text{with} \quad a_4 = 230 &&\text{and} \quad j(E_{a_4}) = 234.
\end{aligned}$$

Figure 9 depicts Alice's shared secret computation.

Bob's shared secret computation. Bob starts from the curve output in Alice's public key in (6), taking his new starting curve E_{a_0} with $a_0 = 423i+179$. His first step is again to compute a secret generator on E_{a_0} corresponding to her secret k_A, as

$$\begin{aligned}
S_B &= \phi_A(P_B) + [k_B]\phi_A(Q_B)\\
&= (142i + 183, 119i + 360) + [2](220i + 314, 289i + 10)\\
&= (393i + 124, 187i + 380).
\end{aligned}$$

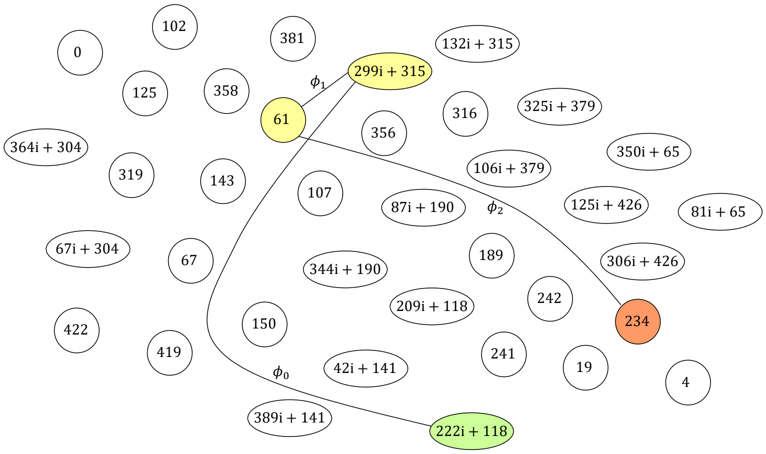

Fig. 10. Bob's shared secret computation. He starts at the curve from Alice's public key with $j = 222i + 118$, and uses his secret key to compute the walk to the node with $j = 234$, which is the shared secret.

Again, Bob proceeds exactly as he did during key generation, with the exception of moving Alice's basis points through the isogeny. His three 3-isogeny computations are summarised as

$$\begin{aligned}
&\phi_0\colon E_{a_0} \to E_{a_1}, && \text{with} \quad a_1 = 183i + 177 && \text{and} \quad j(E_{a_1}) = 299i + 315;\\
&\phi_1\colon E_{a_1} \to E_{a_2}, && \text{with} \quad a_2 = 31 && \text{and} \quad j(E_{a_2}) = 61;\\
&\phi_2\colon E_{a_2} \to E_{a_3}, && \text{with} \quad a_3 = 230 && \text{and} \quad j(E_{a_3}) = 234.
\end{aligned}$$

Figure 10 depicts Bob's shared secret computation. Referring back to Figure 9, we see that Alice and Bob have indeed arrived at the same shared value.

6 SIDH in practice

As promised in Section 1, extending one's understanding from the running toy example to parameters of cryptographic size is rather straightforward. The four sets of primes in the SIKE proposal [8] are defined in exactly the same way as the toy prime $p = 2^4 3^3 - 1$. They are also of the form $p = 2^{e_A} 3^{e_B} - 1$, but instead have

$$\begin{aligned}
&e_A = 216 && \text{and} && e_B = 137 && \text{in} \quad \texttt{SIKEp434};\\
&e_A = 250 && \text{and} && e_B = 159 && \text{in} \quad \texttt{SIKEp503};\\
&e_A = 305 && \text{and} && e_B = 192 && \text{in} \quad \texttt{SIKEp610};\\
&e_A = 372 && \text{and} && e_B = 239 && \text{in} \quad \texttt{SIKEp751}.
\end{aligned} \tag{8}$$

So, in `SIKEp434` for example, Alice will be computing 216 steps in the 2-isogeny graph and Bob will be computing 137 steps in the 3-isogeny graph. The name `SIKEp434` refers to the underlying prime p having 434 bits, and in this case the number of supersingular j-invariants (or nodes in the corresponding supersingular isogeny graphs) is between 2^{429} and 2^{430}.

The remainder of this section aims to highlight the more subtle differences that arise when ramping up from toy parameters to those of cryptographic size, and to bring the reader up to speed with the current state-of-the-art in SIDH.

Curse of the toy example. Of the 37 supersingular j-invariants over $\mathbb{F}_{431^2}$, 21 of them are defined over the ground field $\mathbb{F}_{431}$. This may seem to suggest that most (or at least a reasonable fraction) of the j-invariants will be defined over $\mathbb{F}_p$ in general, but it is important to stress that, as p grows large, this is not the case. As we saw in Section 2, the number of supersingular j-invariants in characteristic p is roughly $\lfloor p/12 \rfloor$, but it turns out that the number of supersingular j-invariants defined over $\mathbb{F}_p$ is in $\tilde{O}(\sqrt{p})$. In other words, for the types of supersingular isogeny graphs we work with in practice, the probability of a randomly chosen node (i.e. j-invariant) being defined over $\mathbb{F}_p$ is negligibly small. The reason that this is important is that the underlying problem of finding isogeny paths between two supersingular curves (more on this in Section 7) becomes much easier when both curves are defined over $\mathbb{F}_p$. For the problem to be as hard as possible, we need at least one of E and E' to have a j-invariant that is not in $\mathbb{F}_{p^2}$, and thus we want the chances of accidentally walking into such a j-invariant to be negligible.

Referring back to Figures 4 and 5, recall that the nodes corresponding to $j \in \{0, 4, 125, 242\}$ appeared to have unusual edge behaviour in one or both of the 2- and 3-isogeny graphs. In all of these cases, there is either an edge from the given node back to itself, there is not the usual number of edges connecting to this node, or both. These correspond to exceptional cases where either (i) isogenies degenerate into isomorphisms (e.g. the 2-isogeny with kernel $(0, 0)$ on E_0), or (ii) two or more kernel elements produce the same image j-invariant (e.g. the 3-isogenies emanating from the curve with $j = 4$). Again, with 4 out of 37 nodes being exceptional in our example, it may seem that the number of exceptional nodes is non-negligible as p grows large. However, the number of exceptional nodes stays this small as p grows larger, as can be seen by trying to force these exceptions in the isogeny formulas.

Finally, observe that Alice and Bob both walked to (or through) the node with $j = 344i + 190$ during key generation. This is another anomaly that occured because of the size of our example, which will be overwhelmingly unlikely for large parameter sets. Besides the exceptional j's mentioned above, the edges in Figures 4 and 5 appear to be completely unrelated, and indeed this lack of any meaningful connection between Alice and Bob's graphs is at the heart of the security underlying SIDH.

Efficient isogeny computations. The isogeny computation routines we walked through in Section 5 give the general picture of how we compute ℓ^e-isogenies for $\ell \in \{2, 3\}$. Namely, we start with a secret generator P of order ℓ^e, compute the point $[\ell^{e-1}]P$ of order ℓ by a scalar multiplication, and then use this point as the kernel of our first ℓ-isogeny $\phi_0 \colon E_0 \to E_1$. We then move the original point P through ϕ_0 to compute $\phi_0(P)$ as the point of order ℓ^{e-1} on E_1, and start iterating this process by beginning with another scalar multiplication $[\ell^{e-2}]\phi_0(P)$. This was the method used in the original Jao and De Feo paper [9], but in the extended De Feo-Jao-Plût paper [5] it was shown that it is possible to do much better.

To sketch the idea, note that during the first scalar multiplication $P \mapsto [\ell^{e-1}]P$, we can store an intermediate multiple of P, say $Q = [\ell^d]P$ for $1 \leq d \leq e-2$. Then, rather than moving P through $\phi_0(P)$ and computing the scalar multiplication $\phi_0(P) \mapsto [\ell^{e-2}]\phi_0(P)$, we can move Q through, and use it to compute the scalar multiplication $\phi_0(Q) \mapsto [\ell^{e-d-2}]\phi_0(Q)$ instead. The larger we choose d, the more we save on this second scalar multiplication. However, the larger we choose d, the fewer the number of iterations the point $Q = [\ell^d]P$ is of use; each time we compute an isogeny, the orders of its successive images decrease by a factor of ℓ, until it is eventually moved to the point at infinity and is of no use thereafter. One fix that may spring to mind is to store all of the intermediate multiples of P for $1 \leq d \leq e-2$, which is certainly plausible, but this then means we would have to pull all $e-3$ of these points through the first isogeny, $e-4$ through the second isogeny, and so on, which becomes cumbersome. Determining exactly which multiples to store in the first step, and every consecutive step thereafter, becomes a rather complicated looking combinatorial problem whose optimisation factors in the value of e, and the cost ratios of scalar multiplication and isogeny operations. Fortunately, De Feo, Jao and Plût provide a fully optimised solution to this problem [5, §4.2.2] that becomes relatively simple to implement in practice. Interested readers can consult the SIKE specification [8] and the source codes that accompany it for further details.

Note that, while the explicit formulas for point and isogeny operations in this tutorial have been presented in affine space, i.e. with inversions, all of them have projective analogues that avoid inversions entirely. Just like in old school ECC, every secret operation in SIDH can be performed projectively and avoid all but the final inversion, which is used to normalise the result into its unique affine representation.

Compressed public keys. For ease of exposition, in Section 5 we conformed to the original specification of public keys, where Alice and Bob both produce public keys of the form

$$\mathrm{PK} = \big(\phi(E), \phi(P), \phi(Q)\big), \tag{9}$$

with ϕ being their secret isogeny, E is the starting curve, and with P and Q being the basis points of the other party. We wrote the curve E using one element of $\mathbb{F}_{p^2}$ (the Montgomery a parameter), and the points P and Q using two elements of $\mathbb{F}_{p^2}$ each. In practice, however, we actually specify the public keys inside the x-only Montgomery framework as $\mathrm{PK} = [x(\phi(P)), x(\phi(Q)), x(\phi(Q-P)),$ where $x(\phi(P))$ is the x-coordinate of $\phi(P)$, etc, and where these three elements can be used to recover the Montgomery a coordinate in a handful of field operations [8, §1.1]. This requires only 3 elements of $\mathbb{F}_{p^2}$, is preferable to what we used in Section 5, and this is the state-of-the-art for "uncompressed" SIDH public keys.

It turns out that we can actually compress the public keys much further [2] by focussing on the second and third components of the public key in (9). The high level idea is that the $\mathbb{F}_{p^2}$ elements used to transmit the points $\phi(P)$ and $\phi(Q)$ are rather large compared to the size of the integer coefficients that are needed to represent them with respect to a given basis. We will return to our toy example to illustrate, recalling Alice's public key from (6) as

$$\begin{aligned}\mathrm{PK}_A &= (\phi_A(E_{a_0}), \phi_A(P_B), \phi_A(Q_B)) \\ &= (423i + 179, (142i + 183, 119i + 360), (220i + 314, 289i + 10)),\end{aligned}$$

where she had moved to the curve $E\colon y^2 = x^3 + (423i + 179)x^2 + x$, on which the points $\phi_A(P_B)$ and $\phi_A(Q_B)$ have order 3^3. For *any* basis of the 3^3-torsion on E, say the points R and S with $E[3^3] = \langle R, S\rangle$, we can write the points $\phi_A(P_B)$ and $\phi_A(Q_B)$ as

$$\phi_A(P_B) = [\alpha]R + [\beta]S \qquad \text{and} \qquad \phi_A(Q_B) = [\gamma]R + [\delta]S, \tag{10}$$

for some $\alpha, \beta, \gamma, \delta \in \{0, 1, 2, \ldots, 26\}$. If Alice can send $(\alpha, \beta, \gamma, \delta)$ instead of $\phi_A(P_B)$ and $\phi_A(Q_B)$, or even instead of $x(\phi_A(P_B))$ and $x(\phi_A(Q_B))$, this will be a significant decrease in bandwidth; 4 elements of $\mathbb{Z}_{27}$ is much smaller than the 4 elements of $\mathbb{F}_{431}$ that would be needed to send $x(\phi_A(P_B))$ and $x(\phi_A(Q_B))$. This whole idea hinges on Alice being able to indeed send $(\alpha, \beta, \gamma, \delta)$ to Bob without sending the basis points. This is achieved by Alice computing a basis $\{R, S\}$ deterministically from the curve $\phi_A(E_{a_0})$, in such a way that Bob will be able to start with her compressed public key

$$\big(\phi_A(E_{a_0}), (\alpha, \beta, \gamma, \delta)\big),$$

use $\phi_A(E_{a_0})$ to recover the same basis points R and S, and then recover $\phi_A(P_B)$ and $\phi_A(Q_B)$ exactly as in (10). Fortunately, it is relatively easy to devise methods to derive bases deterministically – see [8]. All that then remains is to show how Alice can decompose her points $\phi_A(P_B)$ and $\phi_A(Q_B)$ into $(\alpha, \beta, \gamma, \delta)$ with respect to this basis. This requires that Alice can solve two-dimensional discrete logarithm problems of the form in (10), which turns out to be possible in practice due to all of our supersingular elliptic curves having very smooth group orders[5], i.e. group orders whose largest prime factors are 3.

In general, both Alice and Bob can compress their public keys to be of the form $\big(\phi_A(E_{a_0}), (\alpha, \beta, \gamma, \delta)\big)$ and $\big(\phi_B(E_{a_0}), (\alpha', \beta', \gamma', \delta')\big)$, improving from three elements in $\mathbb{F}_{p^2}$ to one element of $\mathbb{F}_{p^2}$ and 4 elements of either $\mathbb{Z}_{2^{e_A}}$ or $\mathbb{Z}_{3^{e_B}}$. Furthermore, one of the integer components can be neglected in both cases by performing a normalisation across the bases. Assuming that $2^{e_A} \approx 3^{e_B} \approx p^{1/2}$, this shrinks both public keys to around 60% of their original size – see [8] for further details.

7 Security and cryptanalysis

We conclude the tutorial with a brief discussion on the security of SIDH. The first order of business is to address the difference between SIDH and SIKE.

SIDH versus SIKE. The computational problems related to the SIDH protocol are usually stated by listing a tuple corresponding to all of the information that a passive adversary would see, and then asking the attacker to either find the underlying secret isogeny (the search problem – see [7, Definition 2]), or to decide whether or not the tuple is indeed a well-formed instance of the SIDH problem (the decisional problem – see [7, Definition 3]). Given the four basis points in the public parameters, and their image points that are included in the public keys, these tuples contain substantially more information than just the preimage and image curves of the secret isogeny. One of the main assumptions made in the SIDH landscape is that all of these preimage and image points do

[5] We remind the reader that the security of SIDH/SIKE is unrelated to discrete logarithm problems!

not help a *passive* adversary in solving the more general problem which is to *compute the degree-ℓ^e isogeny*

$$\phi\colon E \to E', \tag{11}$$

that connects the curves E and E' in the supersingular isogeny graph. This assumption has remained valid to date, and there have been no known passive attacks that can make use of the auxiliary image points for the types of SIDH parameters used in practice.

However, the need to transmit these image points becomes a significant drawback when considering the combination of active adversaries and the use of static keys. Galbraith, Petit, Shani and Ti [6] showed that, if Alice or Bob is reusing a secret key for many protocol instances, a malicious party can actively learn their entire secret by performing as many interactions as the bitlength of their key.

We will return to our running example in Section 5 to illustrate the basic idea. Recall from (6) that Alice's public key was

$$\begin{aligned}\mathrm{PK}_A &= (\phi_A(E_{a_0}), \phi_A(P_B), \phi_A(Q_B))\\ &= (423i + 179, (142i + 183, 119i + 360), (220i + 314, 289i + 10)),\end{aligned}$$

which corresponded to her secret $k_A = 11$, and recall that Bob's public key was

$$\begin{aligned}\mathrm{PK}_B &= (\phi_B(E_{a_0}), \phi_B(P_A), \phi_B(Q_A))\\ &= (273i + 76, (187i + 226, 43i + 360), (325i + 415, 322i + 254)),\end{aligned}$$

which corresponded to his secret $k_B = 2$. Furthemore, recall that the first step in Alice's shared secret computation was to use Bob's public key and her secret to compute the new kernel as

$$\begin{aligned}S_A &= \phi_B(P_A) + [k_A]\phi_B(Q_A)\\ &= (187i + 226, 43i + 360) + [11](325i + 415, 322i + 254)\\ &= (125i + 357, 415i + 249).\end{aligned}$$

Now suppose that Bob wants to learn Alice's static secret. Following [6, §3.1], rather than sending his second image point $\phi_B(Q_A) = (325i + 415, 322i + 254)$, he will add it to a point of order 2 on $E_{273i+76}$, say $T_2 = (245i + 27, 0)$, to get $(325i + 415, 322i + 254) + (245i + 27, 0) = (76i + 247, 114i + 208)$, and instead send the malicious public key

$$\begin{aligned}\mathrm{PK}'_B &= (\phi_B(E_{a_0}), \phi_B(P_A), \phi_B(Q_A) + T_2)\\ &= (273i + 76, (187i + 226, 43i + 360), (76i + 247, 114i + 208)),\end{aligned}$$

where the first two components remain unchanged. Bob will proceed as usual, taking Alice's static key and his secret $k_B = 2$ to arrive at the shared secret $j_B = 234$ as in Figure 10. However, Alice will now compute

$$\begin{aligned}S'_A &= \phi_B(P_A) + [k_A](\phi_B(Q_A) + T_2)\\ &= (187i + 226, 43i + 360) + [11](90i + 354, 21i + 317)\\ &= (353i + 23, 152i + 277),\end{aligned}$$

which is a point of order 16 that is different from S_A. Moreover, $\langle S'_A \rangle \neq \langle S_A \rangle$, so the two isogenies generated by these two kernels will be different. Indeed, Alice computes the 16-isogeny from $E_{273i+76}$ with kernel $\langle S'_A \rangle$ and arrives at the node with $j_A = 242$.

The SIDH protocol will fail because $j_B \neq j_A$, at which point Bob immediately knows the final bit of Alice's secret; if Alice's secret were even, then it is easy to see that Bob's adding the point T_2 to $\phi_B(Q_A)$ would have produced $S'_A = S_A$, and the protocol would have gone through smoothly. Thus, whether or not the protocol succeeds, Bob learns a bit of Alice's secret. In [6, §3.2] the attack is continued *bit by bit* until Bob reconstructs all but the last two bits of Alice's secret, which can be brute forced with no further interaction – see [6, Algorithm 1].

Unfortunately there is currently no known way for Alice to check that Bob's public key is well-formed and non-malicious, and this is the reason that SIDH must either (i) insist that both parties use purely ephemeral secret keys, i.e. use each secret key once, or (ii) use a generic passive-to-active transformation (see [8]) to allow one of the two parties to reuse a long-term secret.

SIKE stands for supersingular isogeny key encapsulation, which applies such a generic transformation to SIDH in order to allow Alice to safely use a long-term static secret. The basic idea is that Bob must use Alice's fixed public key and his secret key to compute the true shared secret j before sending any information to Alice. Moreover, the secret key k_B he uses is computed as the output of a cryptographic hash function whose input is Alice's public key and a randomly chosen value m, i.e. he uses $k_B = H(\mathrm{PK}_A, m)$. Along with his usual public key, he *encapsulates* the random value m by XOR-ing it with a hash of the shared secret, $H'(j)$, and sends

$$(\mathrm{PK}_B, H'(j) \oplus m)$$

to Alice, where PK_B is as usual. She can now use PK_B and her secret key to compute j and then $H'(j)$, and this allows her to recover Bob's initial random value m. Alice can then recompute Bob's secret $k_B = H(\mathrm{PK}_A, m)$ and then check that PK_B is exactly as it should be, i.e. that Bob has not acted maliciously. If this check passes, Alice can carry on as usual, but if not, she can presume Bob is acting maliciously and can output garbage. This way, Bob will always get back garbage if he tampers with the protocol, and he will learn nothing about Alice's secret through doing so.

Basic properties of supersingular isogeny graphs. In discussing the security landscape of SIDH, it is important to understand some basic facts about supersingular isogeny graphs; these can be seen for our toy example by eyeballing Figures 4 and 5, but they hold true in general. Supersingular ℓ-isogeny graphs are both *connected*, meaning there is always a path between any two nodes, and $(\ell+1)$-*regular*, meaning every vertex has precisely $\ell + 1$ connected neighbouring vertices[6]. They are also instances of *expander graphs* which are said to have *rapid*

[6] Edges can have multiplicities greater than 1, which takes care of the tiny handful of anomalies we discussed in Section 6.

mixing. Very roughly speaking, this means that when starting at any node in the graph, only a relatively small number of steps (i.e. logarithmic in the number of nodes) is needed for a random walk to converge to a uniform distribution. In other words, we can pick any node in the graph, and so long as our walks are of a requisite (but relatively small) length, they can take us to any other node in the graph. Another way to state this is in terms of the graph's *diameter*, which is the maximum number of steps needed to connect any two nodes in the graph; an expander graph with N nodes has a diameter in $O(\log N)$. Astute readers should consult [5, §2] and the references therein for more precise statements[7].

The van Oorschot-Wiener collision finding algorithm. We now turn to describing the current state-of-the-art in cryptanalysis against SIDH. Assuming that both Alice and Bob use one-time ephemeral SIDH keys, or that SIKE is used to protect one side's static keys, we are now back in the situation where the best known attacks do not use any of the basis points, and where we are trying to solve the problem in (11): given two ℓ^e-isogenous curves E and E' in the supersingular isogeny graph, compute the corresponding isogeny.

For concreteness, consider the smallest set of SIKE parameters with $p = 2^{216}3^{137} - 1$, for which there are more than 2^{429} nodes in the corresponding supersingular isogeny graph, and suppose we are trying to find Alice's secret isogeny

$$\phi_A \colon E \to E_A$$

of degree 2^{216}.

Since the degree of our isogeny is fixed and known, this problem is a rather special instance of the general isogeny problem, which asks to find an isogeny connecting *any* two nodes in the graph. Our job is a lot easier than solving this general problem among the $\approx 2^{429}$ nodes, since the 216 steps taken by Alice falls significantly short of the diameter of the graph. To see this, we can count the number of possible destination nodes for Alice's 2^{216} by counting the number of distinct order-2^{216} subgroups on E; the number of order-ℓ^e subgroups is $\ell^{e-1}(\ell+1)$ in general, so in our case there are exactly $2^{216} + 2^{215}$ possibilities for E_A. We depict this property for our toy example in Figure 11, but emphasise that as p grows large the destination (green) nodes become exponentially sparse among all nodes; the graphs have $O(p)$ nodes and only $O(\sqrt{p})$ of them are possible destination nodes.

It is for this reason that random walk algorithms for solving the general isogeny problem are not the best known algorithms against SIDH. The 216 steps is much smaller than the average number of steps connecting two given nodes (across all pairs of nodes), so it is overwhelmingly likely that the walk Alice took from E to E_A is the shortest walk between those two nodes. Thus, the best algorithms for computing ϕ_A are *meet-in-the-middle* algorithms, whereby

[7] An even less formal but more visual interpretation of this property (with respect to Figures 4 and 5) would be to say that there is no way to position the nodes so that drawing the edges makes the pictures appreciably less messy.

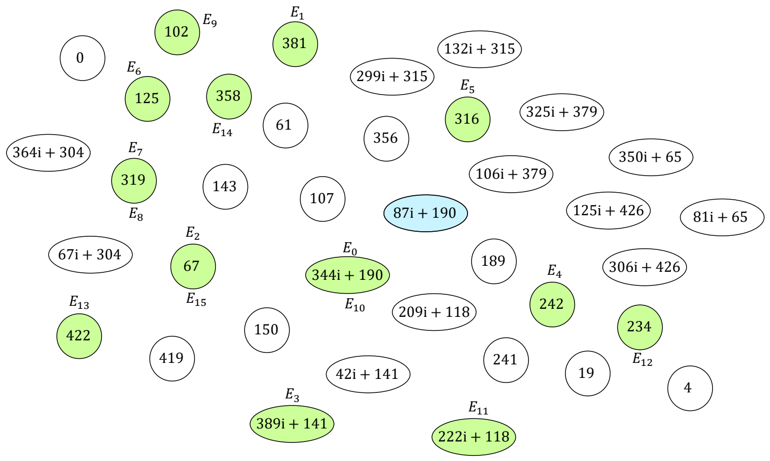

Fig. 11. Alice's possible destination nodes (in green) when starting at E with $j(E) = 87i + 190$, and computing 2^4-isogenies to the curves $E_k = E/\langle P_A + [k]Q_A\rangle$ with $k \in \{0, 1, \ldots, 15\}$.

we perform walks of 108 steps from both E and E_A until two of these walks reach *the* middle node; it is then overwhelmingly likely that these two walks connect to be exactly the walk that Alice took, and the problem is solved.

The generic version of meet in the middle builds a table of *all* of the curves that are 2^{108}-isogenous to E_A (of which there are $2^{108}+2^{107}$), and then computes each 2^{108}-isogeny from E, one at a time, until a match is found in the table. Since $2^{108} \approx p^{1/4}$, this gives an attack that runs in $O(p^{1/4})$ time and requires $O(p^{1/4})$ memory (so long as SIDH instances are constructed with $2^{e_A} \approx 3^{e_B}$, then these complexities remain the same when targeting either party's secret isogeny). Indeed, these were the asymptotic complexities originally stated by Jao and De Feo [9], and those that were used to analyse SIKE's security in the Round 1 submission.

A 2018 paper by Adj, Cervantes-Vázquez, Chi-Domínguez, Menezes and Rodríguez-Henríquez [1] analysed the security of SIDH and SIKE more concretely, and showed that the generic version of meet-in-the-middle is, overall, not the best algorithm for solving the underlying isogeny problems. More specifically, they argued that the exponential storage requirements (of more than 2^{108} bits in the smallest case of `SIKEp434`) essentially makes generic meet-in-the-middle irrelevant, particularly when comparing its runtime to the runtimes of attacks against other cryptographic primitives (as is required when positioning the security against NIST's target levels – see [15]). Adj *et al.* proposed that the correct way to analyse security is to fix an upper bound on the possible memory (they specified $w = 2^{80}$ as a possible yet still presently infeasible such bound), and to analyse the runtimes of the best known algorithms subject to this storage capacity. Their conclusion is that the van Oorschot-Wiener (vOW) parallel collision finding algorithm [17] is then the superior algorithm for finding ϕ.

We will briefly sketch the intuition behind the vOW algorithm in our context. Since all of the curves that are 2^{108}-isogenous to E_A can no longer fit in our table (of, say, size $w = 2^{80}$), we instead fill the table with curves from both sides, i.e. image curves that are either 2^{108}-isogenous to E or curves that are 2^{108}-isogenous to E_A. As before, we are still hoping for a collision in the table, when a walk from E produces the same image curve as a walk from E_A. However, given the storage capacity, even when we come across this middle curve from one side, the chances that it is already in the table are very small, and we have no way of knowing that this is the middle curve until we have run into it from the other side. Thus, we can either discard that image curve, or use it to replace an existing memory element. Either way, we could be discarding one side of the solution we seek, or the other side could have previously been overwritten.

Henceforth we will stop using the term (image) curves and instead refer to the elements of a set S, which we define to be the set of all j-invariants that are 2^{108}-isogenous to E or 2^{108}-isogenous to E_A. A better way to use the 2^{80} memory is to instead define a deterministic but pseudo-random walk on S, and to define a property that *distinguishes* certain elements of S; these elements are the only ones that we send to memory. In this case we have roughly 2^{109} elements in S, so we could define a property that distinguishes around 1 in every 2^{30} elements in order to fit all (if not close to all) of the distinguished elements in memory. Our pseudo-random function

$$f \colon S \to S$$

will take a j-invariant, feed it into a cryptographic hash function to produce a new string, use one bit of this string to decide whether we are to compute an isogeny from E or E_A, and then use the rest of the string to choose a subgroup/isogeny, which in turn produces a new j-invariant. To produce pseudo-random walks on S, we start with an initial element $x_0 \in S$ and use the function $f \colon x_i \mapsto x_{i+1}$ to iterate until we come across an element, x_n, that is distinguished[8]. When this happens, we store the distinguished element (together with the initial value x_0) in memory, otherwise we keep walking. We pick starting points at random, and perform many of these walks to collect pairs of initial and distinguished elements in memory; moreover, these walks are independent of each other so the process parallelises perfectly.

Let $x \in S$ and $y \in S$ be such that $x \neq y$, $f(x) = f(y) = z$, x gets mapped (via f) through a subgroup on E, and y gets mapped through a subgroup on E_A. It follows that z is the middle j-invariant that solves our problem. Now, it is very unlikely that z will be a distinguished element, but so long as we have a walk that finds x and another walk that finds y, f being deterministic guarantees that they will then walk on the same elements and thus into the same distinguished

[8] This distinguishing property can essentially be anything, so long as it yields the right fraction of elements; in our example, a good choice would be to hash the element and check for 30 leading zeros.

element. When the second of these two walks sends its distinguished element to memory, we will find that this distinguished element has already been found, and we will then be able to redo both walks and recover x and y.

Using such a function f and a distinguished element criterion solves our problem of not being able to store all of the elements corresponding to E_A in memory, but in doing so it introduces several new problems. Many of these are closely related to f behaving like a random function. For example, while we know that there is a unique subgroup on E and a unique subgroup on E_A that solves the problem, our particular choice of f (recall that it involves a cryptographic hash function) might be such that there is no x and/or y that pass through these subgroups under f, so that we will never terminate with the solution. Or, even if there are values of x and y that solve the problem under f, it could be that x and y themselves have no preimages under f. In this case our only chance of solving the problem is in the overwhelmingly unlikely case that we happen to pick both of them as starting points of our random walks. Furthermore, the solution we seek is no longer the only $x \neq y$ with $f(x) = f(y)$; in fact, we have introduced many, many more such collisions by virtue of f behaving like a random function.

What we would ultimately hope for is that we use a function f for which both x and y exist (ideally, several such x and y would exist), and for which there are many random walks that lead to x and y under f. The problem is, we have no way of knowing how good or bad a given random function f is for our particular problem instance. Thus, as is discussed by van Oorschot and Wiener [17], after filling and refilling/overwriting our entire memory under a given function f, the optimisation of this entire process forces us to abandon all of the prior computations and switch to an entirely new function, essentially restarting the attack.

Both the implementation and the concrete runtime analysis [17, §4.2] of this algorithm are non-trivial. In summary, a line of recent works studying and implementing the attack [1,10,3] all essentially confirm the original runtime analysis of van Oorshot and Wiener in the context of SIDH: if the available memory can hold w elements from the set of size S, and with m processors working in parallel, the runtime, T, of the algorithm is

$$T = \left(\frac{2.5}{m} \sqrt{\frac{|S|^3}{w}} \right) \cdot t,$$

where t is the time taken to compute one function iteration, which boils down to one $\ell^{e/2}$-isogeny computation. In SIDH we have $|S| \approx p^{1/4}$, so (with $w = 2^{80}$) we can see that the runtime of vOW becomes worse than the runtime of generic meet-in-the-middle for all of the SIKE parameters, and this gap widens as p grows larger. Nevertheless, this remains the best known concrete algorithm for solving the specialised isogeny problems underlying SIDH.

Finally, we remark that the vOW algorithm is entirely classical. An excellent recent paper by Jaques and Schanck [10] investigated a range of *quantum* attacks against SIDH, but ultimately showed that the concrete improvement across all relevant avenues of quantum attack is rather minimal, if at all.

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. 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.
4. L. De Feo. Mathematics of isogeny based cryptography. *CoRR*, abs/1711.04062, 2017. `https://arxiv.org/pdf/1711.04062.pdf`.
5. 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.
6. 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.
7. S. D. Galbraith and F. Vercauteren. Computational problems in supersingular elliptic curve isogenies. *Quantum Information Processing*, 17(10):265, 2018.
8. 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.
9. 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.
10. 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.
11. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
12. J. H. Silverman. *The Arithmetic of Elliptic Curves, 2nd Edition*. Graduate Texts in Mathematics. Springer, 2009.
13. B. Smith. Pre- and post-quantum Diffie-Hellman from groups, actions, and isogenies. In L. Budaghyan and F. Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields - 7th International Workshop, WAIFI 2018, Bergen, Norway, June 14-16,*

2018, Revised Selected Papers, volume 11321 of *Lecture Notes in Computer Science*, pages 3–40. Springer, 2018.

14. J. Tate. Endomorphisms of abelian varieties over finite fields. *Inventiones mathematicae*, 2(2):134–144, 1966.
15. 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`.
16. D. Urbanik. A friendly introduction to supersingular isogeny Diffie-Hellman, May, 2017. URL: `https://csclub.uwaterloo.ca/~dburbani/work/friendlysidh.pdf`.
17. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999.
18. J. Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris Sér. AB*, 273:A238–A241, 1971.