Towards practical key exchange from ordinary isogeny graphs

Luca De Feo[1,3][0000−0002−9321−0773], Jean Kieffer[2], and Benjamin Smith[3]

[1] Université Paris Saclay, UVSQ, LMV, Versailles, France
[2] École Normale Supérieure, Paris, France
[3] Inria and École polytechnique, Université Paris Saclay, Palaiseau, France

Abstract. We revisit the ordinary isogeny-graph based cryptosystems of Couveignes and Rostovtsev–Stolbunov, long dismissed as impractical. We give algorithmic improvements that accelerate key exchange in this framework, and explore the problem of generating suitable system parameters for contemporary pre- and post-quantum security that take advantage of these new algorithms. We also prove the session-key security of this key exchange in the Canetti–Krawczyk model, and the IND-CPA security of the related public-key encryption scheme, under reasonable assumptions on the hardness of computing isogeny walks. Our systems admit efficient key-validation techniques that yield CCA-secure encryption, thus providing an important step towards efficient post-quantum non-interactive key exchange.

Keywords: post-quantum cryptography · key exchange · elliptic curves · isogenies

1 Introduction

Isogeny-based protocols form one of the youngest and least-explored families of post-quantum candidate cryptosystems. The best-known isogeny-based protocol is Jao and De Feo's SIDH key exchange [35], from which the NIST candidate key-encapsulation mechanism SIKE was derived [3,53]. SIDH was itself inspired by earlier key-exchange constructions by Couveignes [18] and Rostovtsev and Stolbunov [57,61,62], which were widely considered unwieldy and impractical.

Indeed, the origins of isogeny-based cryptography can be traced back to Couveignes' seminal work "Hard Homogeneous Spaces", that went unpublished for ten years before appearing in [18]. A *principal homogeneous space* (PHS) for a group G is a set X with an action of G on X such that for any $x, x' \in X$, there is a unique $g \in G$ such that $g \cdot x = x'$. Equivalently, the map $\phi_x : g \mapsto g \cdot x$ is a bijection between G and X for any $x \in X$. Couveignes defines a *hard homogeneous space* (HHS) to be a PHS where the action of G on X is efficiently computable, but inverting the isomorphism φ_x is computationally hard for any x.

Any HHS X for an *abelian* group G can be used to construct a key exchange based on the hardness of inverting φ_x, as shown in Algorithms 1 and 2. If Alice

Algorithm 1: Key generation for cryptosystems in an HHS X for a group G, with a fixed "base point" x_0 in X.

Input: ()
Output: A private-public keypair $(g, x) \in G \times X$ s.t. $x = g \cdot x_0$

```
function KeyGen()
    g ← RANDOM(G)          // g is sampled uniformly at random from G
    x ← g · x_0
    return (g, x)
```

and Bob have keypairs (g_A, x_A) and (g_B, x_B), respectively, then the commutativity of G lets them derive a shared secret

$$\mathtt{DH}(g_A, x_B) = g_A \cdot g_B \cdot x_0 = g_B \cdot g_A \cdot x_0 = \mathtt{DH}(g_B, x_A)\,.$$

The analogy with classic group-based Diffie–Hellman is evident.

Algorithm 2: Diffie–Hellman in an HHS X for a group G

Input: A private key $g_A \in G$ and a public key $x_B \in X$, each generated by calls to `KeyGen`
Output: A shared secret value $k \in X$

```
function DH(g_A,x_B)
    k ← g_A · x_B
    return k
```

For instance, if X is a cyclic group $\langle x \rangle$ of order p, and $G = (\mathbb{Z}/p\mathbb{Z})^*$ acts on $X \setminus \{1\}$ by $g \cdot x = x^g$, then inverting φ_x is the discrete logarithm problem (DLP) in X. But for other homogeneous spaces, inverting φ_x may have no relationship with any DLP, and might resist attacks based on Shor's algorithm in the quantum setting. Similar ideas have occasionally appeared in the literature in different forms [39,47].

Couveignes viewed HHS chiefly as a general framework, encompassing various flavors of Diffie–Hellman-like systems. Nevertheless, he suggested using a specific HHS based on the theory of complex multiplication of elliptic curves, in a sense generalizing the class-group-based Diffie–Hellman key exchange of Buchmann and Williams [9]. Independently, Rostovtsev and Stolbunov proposed in [57] a public key encryption scheme based on the same HHS. Later, Stolbunov [62] derived more protocols from their primitive, including an interactive key exchange scheme similar to Algorithm 2. Rostovtsev and Stolbunov's proposal crucially deviates from the HHS paradigm in the way random elements of G are sampled, as we will explain in Section 3. This makes the primitive less flexible, but also (relatively) more practical.

Rostovtsev and Stolbunov advertised their cryptosystems as potential post-quantum candidates, leading Childs, Jao and Soukharev to introduce the first

subexponential quantum algorithm capable of breaking them [12]. Hence, being already slow enough to be impractical in a classical security setting, their primitive appeared even more impractical in a quantum security setting.

The aim of the present paper is to improve and modernize the Couveignes–Rostovtsev–Stolbunov (CRS) construction, borrowing techniques from SIDH and point-counting algorithms, to the point of making it usable in a post-quantum setting. Our main contributions are in Sections 3–4, where we present a new, more efficient way of computing the CRS group action, and in Section 5 where we give precise classic and quantum security estimates, formalize hardness assumptions, and sketch security proofs in stronger models than those previously considered. We also discuss the cryptographic advantages that this primitive has over SIDH. Finally, in Section 6 we present a proof-of-concept implementation and measure its performance. While the final result is far from being competitive, we believe it constitutes progress in the direction of having a valid isogeny-based alternative to SIDH.

CSIDH. While preparing this paper we were informed of recent work by Castryk, Lange, Martindale, Panny, and Renes, introducing CSIDH, an efficient post-quantum primitive very similar to CRS [11]. Their work builds upon the ideas presented in Sections 3–4, and uses them in a different homogeneous space where they apply effortlessly. Their breakthrough confirms that, if anything, our techniques were a fundamental step towards the first practical post-quantum non-interactive key exchange protocol.

Side channel awareness. The algorithms we present here are not intended to provide any protection against basic side-channel attacks. Uniform and constant-time algorithms for arbitrary-degree isogeny computations are an interesting open problem, but they are beyond the scope of this work.

Acknowledgments. We would like to thank Wouter Castryk, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes for sharing a draft of their paper with us, and Alexandre Gélin and François Morain for fruitful discussions.

2 Isogenies and complex multiplication

We begin by recalling some basic facts on isogenies of elliptic curves over finite fields. For an in-depth introduction to these concepts, we refer the reader to [59]. For a general overview of isogenies and their use in cryptography, we suggest [20].

2.1 Isogenies between elliptic curves

In what follows $\mathbb{F}_q$ is a finite field of characteristic p with q elements, and $\overline{\mathbb{F}}_q$ is its algebraic closure. Let E and E' be elliptic curves defined over $\mathbb{F}_q$. A homomorphism $\phi : E \to E'$ is an algebraic map sending 0_E to $0_{E'}$; it induces a group homomomorphism from $E(\overline{\mathbb{F}}_q)$ to $E'(\overline{\mathbb{F}}_q)$ [59, III.4]. An *endomorphism* is

a homomorphism from a curve to itself. The endomorphisms of E form a ring $\mathrm{End}(E)$, with the group law on E for addition and composition for multiplication. The simplest examples of endomorphisms are the scalar multiplications $[m]$ (mapping P to the sum of m copies of P) and the *Frobenius* endomorphism

$$\begin{aligned} \pi : E &\longrightarrow E\,, \\ (x, y) &\longmapsto (x^q, y^q)\,. \end{aligned}$$

As an element of $\mathrm{End}(E)$, Frobenius satisfies a quadratic equation $\pi^2 + q = t\pi$. The integer t (the *trace*) fully determines the order of E as $\#E(\mathbb{F}_q) = q + 1 - t$. A curve is called *supersingular* if p divides t, *ordinary* otherwise.

An *isogeny* is a non-zero homomorphism of elliptic curves. The degree of an isogeny is its degree as an algebraic map, so for example the Frobenius endomorphism π has degree q, and the scalar multiplication $[m]$ has degree m^2. Isogenies of degree ℓ are called ℓ-isogenies. The kernel $\ker\phi$ of ϕ is the subgroup of $E(\overline{\mathbb{F}}_q)$ that is mapped to $0_{E'}$. An isogeny ϕ is *cyclic* if $\ker\phi$ is a cyclic group.

An *isomorphism* is an isogeny of degree one. An *isomorphism class* of elliptic curves is fully determined by their common *j-invariant*, which is an element of $\overline{\mathbb{F}}_q$. If any curve in the isomorphism class is defined over $\mathbb{F}_q$, then its j-invariant is in $\mathbb{F}_q$.

Any isogeny can be factored as a composition of a *separable* and a *purely inseparable* isogeny. *Purely inseparable* isogenies have trivial kernel, and degree a power of p. *Separable* isogenies include all isogenies of degree coprime to p. Up to isomorphism, separable isogenies are in one-to-one correspondence with their kernels: for any finite subgroup $G \subset E$ of order ℓ there is an elliptic curve E/G and an ℓ-isogeny $\phi : E \to E/G$ such that $\ker\phi = G$, and the curve and isogeny are unique up to isomorphism. In particular, if ϕ is separable then $\deg\phi = \#\ker\phi$. It is convenient to encode $\ker\phi$ as the polynomial whose roots are the x-coordinates of the points in $\ker\phi$, called the *kernel polynomial* of ϕ.

For any ℓ-isogeny $\phi : E \to E'$, there is a unique ℓ-isogeny $\hat{\phi} : E' \to E$ such that $\phi \circ \hat{\phi} = [\ell]$ on E' and $\hat{\phi} \circ \phi = [\ell]$ on E. We call $\hat{\phi}$ the *dual* of ϕ. This shows that being *ℓ-isogenous* is a symmetric relation, and that being isogenous is an equivalence relation. Further, a theorem of Tate states that two curves are isogenous over $\mathbb{F}_q$ if and only if they have the same number of points over $\mathbb{F}_q$.

2.2 Isogeny graphs

Isogeny-based cryptosystems are based on *isogeny graphs*. These are (multi)-graphs whose vertices are elliptic curves up to isomorphism, and whose edges are isogenies between them (again up to isomorphism). The use of isogeny graphs for algorithmic applications goes back to Mestre and Oesterlé [48], followed notably by Kohel [40], and has been continued by many authors [28,25,30,50,36].

We write $E[\ell]$ for the subgroup of ℓ-torsion points of $E(\overline{\mathbb{F}}_q)$. If ℓ is coprime to p, then $E[\ell]$ is isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^2$. Furthermore, if ℓ is prime then $E[\ell]$ contains exactly $\ell + 1$ cyclic subgroups of order ℓ; it follows that, over $\overline{\mathbb{F}}_q$, there are exactly $\ell + 1$ distinct (non-isomorphic) separable ℓ-isogenies from E to other

curves. Generically, a connected component of the ℓ-isogeny graph over $\overline{\mathbb{F}}_q$ will be an infinite $(\ell+1)$-regular graph (a notable exception is the finite connected component of *supersingular* curves, used in SIDH and related protocols).

In the sequel we restrict to isogenies defined over $\mathbb{F}_q$, which makes the picture more complex. If E and E' are elliptic curves over $\mathbb{F}_q$, then an isogeny $\phi : E \to E'$ is defined over $\mathbb{F}_q$ (up to a twist of E') if and only if the Frobenius endomorphism π on E stabilizes $\ker\phi$. We emphasize that the points in $\ker\phi$ need not be defined over $\mathbb{F}_q$ themselves.

For the vertices of the $\overline{\mathbb{F}}_q$-isogeny graph we use j-invariants, which classify elliptic curves up to $\overline{\mathbb{F}}_q$-isomorphism; but in the sequel we want to work up to $\mathbb{F}_q$-isomorphism, a stronger equivalence. If E and $\tilde{E}$ are not $\mathbb{F}_q$-isomorphic but $j(E) = j(\tilde{E})$, then $\tilde{E}$ is the *quadratic twist* of E (which is defined and unique up to $\mathbb{F}_q$-isomorphism).[4] When E is ordinary, its quadratic twist has a different cardinality (if $\#E(\mathbb{F}_q) = q + 1 - t$, then $\#\tilde{E}(\mathbb{F}_q) = q + 1 + t$), so E and $\tilde{E}$ are in different components of the isogeny graph. But every $\mathbb{F}_q$-isogeny $\phi : E \to E'$ corresponds to an $\mathbb{F}_q$-isogeny $\tilde{\phi} : \tilde{E} \to \tilde{E}'$ of the same degree between the quadratic twists. The component of the $\mathbb{F}_q$-isogeny graph containing an ordinary curve and the component containing its twist are thus isomorphic; we are therefore justified in identifying them, using j-invariants in $\mathbb{F}_q$ for vertices in the $\mathbb{F}_q$-graph.[5] This is not just a mathematical convenience: we will see in Section 3 below that switching between a curve and its twist often allows a useful optimization in isogeny computations.

If an isogeny ϕ is defined over $\mathbb{F}_q$ *and cyclic*, then π acts like a scalar on the points of $\ker\phi$. Thus, for any prime $\ell \neq p$, the number of outgoing ℓ-isogenies from E defined over $\mathbb{F}_q$ can be completely understood by looking at how π acts on $E[\ell]$. Since $E[\ell]$ is a $\mathbb{Z}/\ell\mathbb{Z}$-module of rank 2, the action of π is represented by a 2×2 matrix with entries in $\mathbb{Z}/\ell\mathbb{Z}$ and characteristic polynomial $X^2 - tX + q \bmod \ell$. We then have four possibilities:

(0) π has no eigenvalues in $\mathbb{Z}/\ell\mathbb{Z}$, i.e. $X^2 - tX + q$ is irreducible modulo ℓ; then E has no ℓ-isogenies.

(1.1) π has one eigenvalue of (geometric) multiplicity one, i.e. it is conjugate to a non-diagonal matrix $\left(\begin{smallmatrix}\lambda & *\\ 0 & \lambda\end{smallmatrix}\right)$; then there is one ℓ-isogeny from E.

(1.2) π has one eigenvalue of multiplicity two, i.e. it acts like a scalar matrix $\left(\begin{smallmatrix}\lambda & 0\\ 0 & \lambda\end{smallmatrix}\right)$; then there are $\ell + 1$ isogenies of degree ℓ from E.

(2) π has two distinct eigenvalues, i.e. it is conjugate to a diagonal matrix $\left(\begin{smallmatrix}\lambda & 0\\ 0 & \mu\end{smallmatrix}\right)$ with $\lambda \neq \mu$; then there are two ℓ-isogenies from E.

The primes ℓ in Case (2) are called *Elkies primes* for E; these are the primes of most interest to us. Cases (1.x) are only possible if ℓ divides $\Delta_\pi = t^2 - 4q$,

[4] There is a slight technicality here for j-invariants 0 and 1728, where non-quadratic twists may exist. We can ignore these special cases because these curves will never appear in our cryptosystem: the class groups of their endomorphism rings are trivial, and keyspaces of size 1 are of limited utility in cryptography.

[5] The situation is much more complicated for supersingular graphs, because the curve and its twist are in the same component of the graph; see [22, §2] for details.

the discriminant of the characteristic equation of π; for ordinary curves $\Delta_\pi \neq 0$, so only a finite number of ℓ will fall in these cases, and they will be mostly irrelevant to our cryptosystem. We do not use any ℓ in Case (0).

Since all curves in the same isogeny class over $\mathbb{F}_q$ have the same number of points, they also have the same trace t and discriminant Δ_π. It follows that if ℓ is Elkies for some E in $\mathrm{Ell}_q(\mathcal{O})$, then it is Elkies for every curve in $\mathrm{Ell}_q(\mathcal{O})$.

Hence, if ℓ is an Elkies prime for a curve E, then the connected component of E in the ℓ-isogeny graph is a finite 2-regular graph—that is, a cycle. In the next subsection we describe a group action on this cycle, and determine its size.

2.3 Complex multiplication

In this subsection we focus exclusively on ordinary elliptic curves. If E is an ordinary curve with Frobenius π, then $\mathrm{End}(E)$ is isomorphic to an *order*[6] in the quadratic imaginary field $\mathbb{Q}(\sqrt{\Delta_\pi})$ (see [59, III.9]). A curve whose endomorphism ring contains an order $\mathcal{O}$ is said to have *complex multiplication* by $\mathcal{O}$. For a detailed treatment of the theory of complex multiplication, see [44,60].

The ring of integers $\mathcal{O}_K$ of $K = \mathbb{Q}(\sqrt{\Delta_\pi})$ is its *maximal order*: it contains any other order of K. Hence $\mathbb{Z}[\pi] \subset \mathrm{End}(E) \subset \mathcal{O}_K$, and there is only a finite number of possible choices for $\mathrm{End}(E)$. If we write $\Delta_\pi = d^2\Delta_K$, where Δ_K is the discriminant[7] of $\mathcal{O}_K$, then the index $[\mathcal{O}_K : \mathrm{End}(E)]$ must divide $d = [\mathcal{O}_K : \mathbb{Z}[\pi]]$.

It turns out that isogenies allow us to navigate the various orders. If $\phi : E \to E'$ is an ℓ-isogeny, then one of the following holds [40, Prop. 21]:

- $\mathrm{End}(E) = \mathrm{End}(E')$, and then ϕ is said to be *horizontal*;
- $[\mathrm{End}(E) : \mathrm{End}(E')] = \ell$, and then ϕ is said to be *descending*;
- $[\mathrm{End}(E') : \mathrm{End}(E)] = \ell$, and then ϕ is said to be *ascending*.

Notice that the last two cases can only happen if ℓ divides $d^2 = \Delta_\pi/\Delta_K$, and thus correspond to cases (1.x) in the previous subsection. If ℓ does not divide Δ_π, then ϕ is necessarily horizontal.

We now present a group action on the set of all curves up to isomorphism having complex multiplication by a fixed order $\mathcal{O}$; the key exchange protocol of Section 3 will be built on this action. Let $\mathfrak{a}$ be an invertible ideal in $\mathrm{End}(E) \simeq \mathcal{O}$, and define the $\mathfrak{a}$-*torsion* subgroup of E as

$$E[\mathfrak{a}] = \left\{ P \in E(\overline{\mathbb{F}}_q) \,\middle|\, \sigma(P) = 0 \ \text{ for all } \sigma \in \mathfrak{a} \right\}.$$

This subgroup is the kernel of an isogeny $\phi_\mathfrak{a}$; the codomain $E/E[\mathfrak{a}]$ of $\phi_\mathfrak{a}$ is well-defined up to isomorphism and will be denoted $\mathfrak{a} \cdot E$. The isogeny $\phi_\mathfrak{a}$ is always horizontal—that is, $\mathrm{End}(\mathfrak{a} \cdot E) = \mathrm{End}(E)$—and its degree is the *norm* of $\mathfrak{a}$ as an ideal of $\mathrm{End}(E)$.

[6] An *order* is a subring which is a $\mathbb{Z}$-module of rank 2.

[7] Δ_K is also called a *fundamental discriminant*; it is in $\{0, 1\} \bmod 4$, and either Δ_K or $\Delta_K/4$ is squarefree.

Let $\mathrm{Ell}_q(\mathcal{O})$ be the set of isomorphism classes over $\overline{\mathbb{F}}_q$ of curves with complex multiplication by $\mathcal{O}$, and assume it is non-empty. It turns out that the construction above extends to an action of the group of fractional ideals of $\mathcal{O}$ on $\mathrm{Ell}_q(\mathcal{O})$. Furthermore, the principal ideals act trivially (the corresponding isogenies are endomorphisms), so this action induces an action of the *ideal class group* $\mathcal{C}(\mathcal{O})$ on $\mathrm{Ell}_q(\mathcal{O})$.

The main theorem of complex multiplication states that this action is *simply transitive*. In other terms, $\mathrm{Ell}_q(\mathcal{O})$ is a PHS under the group $\mathcal{C}(\mathcal{O})$: if we fix a curve E as base point, then we have a bijection

$$\begin{aligned} \mathcal{C}(\mathcal{O}) &\longrightarrow \mathrm{Ell}_q(\mathcal{O}) \\ \text{Ideal class of } \mathfrak{a} &\longmapsto \text{Isomorphism class of } \mathfrak{a} \cdot E. \end{aligned}$$

The order of $\mathcal{C}(\mathcal{O})$ is called the *class number* of $\mathcal{O}$, and denoted by $h(\mathcal{O})$. An immediate consequence of the theorem is that $\#\,\mathrm{Ell}_q(\mathcal{O}) = h(\mathcal{O})$.

As before, in practice we work with $\mathbb{F}_q$-isomorphism classes. Then $\mathrm{Ell}_q(\mathcal{O})$ decomposes into two isomorphic PHSes under $\mathcal{C}(\mathcal{O})$, each containing the quadratic twists of the curves in the other. We choose one of these two components, that we will also denote $\mathrm{Ell}_q(\mathcal{O})$ in the sequel. (The choice is equivalent to the choice of isomorphism of $\mathrm{End}(E)$ with $\mathcal{O}$, which is determined by a choice of sign on the image of Frobenius in $\mathcal{O}$.)

Now let ℓ be an Elkies prime for $E \in \mathrm{Ell}_q(\mathcal{O})$. So far, we have seen that the connected component of E in the ℓ-isogeny graph is a cycle of horizontal isogenies. Complex multiplication tells us more. The restriction of the Frobenius to $E[\ell]$ has two eigenvalues $\lambda \neq \mu$, to which we associate the prime ideals $\mathfrak{a} = (\pi - \lambda, \ell)$ and $\hat{\mathfrak{a}} = (\pi - \mu, \ell)$, both of norm ℓ. We see then that $E[\mathfrak{a}]$ is the eigenspace of λ, defining an isogeny $\phi_{\mathfrak{a}}$ of degree ℓ. Furthermore $\mathfrak{a}\hat{\mathfrak{a}} = \hat{\mathfrak{a}}\mathfrak{a} = (\ell)$, implying that $\mathfrak{a}$ and $\hat{\mathfrak{a}}$ are the inverse of one another in $\mathcal{C}(\mathcal{O})$, thus the isogeny $\phi_{\hat{\mathfrak{a}}} : \mathfrak{a} \cdot E \to E$ of kernel $(\mathfrak{a} \cdot E)[\hat{\mathfrak{a}}]$ is the dual of $\phi_{\mathfrak{a}}$ (up to isomorphism).

Hence, the eigenvalues λ and μ define two opposite directions on the ℓ-isogeny cycle, independent of the starting curve, as shown in Figure 1. The size of the cycle is the order of $(\pi - \lambda, \ell)$ in $\mathcal{C}(\mathcal{O})$, thus partitioning the set $\mathrm{Ell}_q(\mathcal{O})$ into cycles of equal size.

3 Key exchange from isogeny graphs

We would like to instantiate the key exchange protocol of Algorithm 2 with the PHS $X = \mathrm{Ell}_q(\mathcal{O})$ for the group $G = \mathcal{C}(\mathcal{O})$, for some well chosen order $\mathcal{O}$ in a quadratic imaginary field. However, given a generic element of $\mathcal{C}(\mathcal{O})$, the best algorithm [37] to evaluate its action on $\mathrm{Ell}_q(\mathcal{O})$ has subexponential complexity in q, making the protocol infeasible. The solution, following Couveignes [18], is to fix a set S of small prime ideals in $\mathcal{O}$, whose action on X can be computed efficiently, and such that compositions of elements of S cover the whole of G. The action of an arbitrary element of G is then the composition of a series of actions by small elements in S. As Rostovtsev and Stolbunov [57] observed, it is useful to visualise this decomposed action as a walk in an isogeny graph.

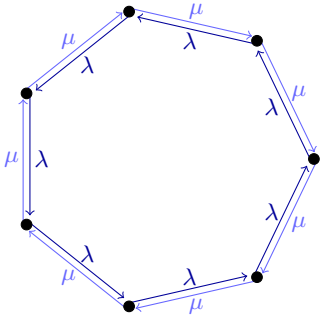

Fig. 1. An isogeny cycle for an Elkies prime ℓ, with edge directions associated with the Frobenius eigenvalues λ and μ.

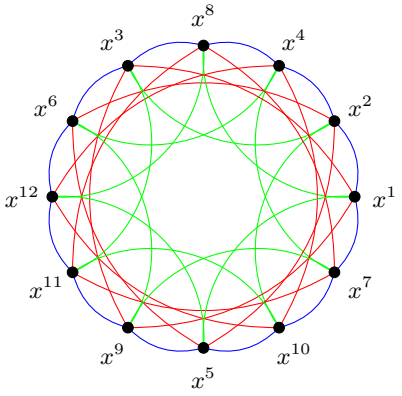

Fig. 2. Undirected Schreier graph on $\langle x \rangle \setminus \{1\}$ where $x^{13} = 1$, acted upon by $(\mathbb{Z}/13\mathbb{Z})^*$, generated by $S = \{2, 3, 5\}$ (resp. blue, red and green edges).

3.1 Walks in isogeny graphs

Let G be a group, X a PHS for G, and S a subset of G. The Schreier graph $\mathcal{G}(G, S, X)$ is the labelled directed graph whose vertex set is X, and where an edge labelled by $s \in S$ links x_1 to x_2 if and only if $s \cdot x_1 = x_2$. It is isomorphic to a Cayley graph for G. If S is symmetric (that is, $S^{-1} = S$), then we associate the same label to s and s^{-1}, making the graph undirected.

A *walk* in $\mathcal{G}(G, S, X)$ is a finite sequence $(s_1, \ldots, s_n)$ of *steps* in S. We define the action of this walk on X as

$$(s_1, \ldots, s_n) \cdot x = \Big(\prod_{i=1}^{n} s_i\Big) \cdot x.$$

In our application G is abelian, so the order of the steps s_i does not matter. We can use this action directly in the key exchange protocol of Algorithm 2, by simply taking private keys to be walks instead of elements in G.

Example 1. Figure 2 shows $\mathcal{G}(G, S, X)$ where $G = (\mathbb{Z}/13\mathbb{Z})^*$, $S = \{2, 3, 5\} \cup \{2^{-1}, 3^{-1}, 5^{-1}\}$, and $X = \langle x \rangle \setminus \{1\}$ is a cyclic group of order 13, minus its identity element. The action of G on X is exponentiation: $g \cdot x = x^g$. The action of 11, which takes x^k to x^{11k}, can be expressed using the walks $(2, 5, 5)$, or $(2^{-1}, 3^{-1})$, or $(3, 5)$, for example. Note that 5 has order 4 modulo 13, thus partitioning $\langle x \rangle \setminus \{1\}$ into 3 cycles of length 4.

Returning to the world of isogenies, we now take

- $X = \mathrm{Ell}_q(\mathcal{O})$ as the vertex set, for some well-chosen q and $\mathcal{O}$; in particular we require $\mathcal{O}$ to be the maximal order (see Section 5).

– $G = \mathcal{C}(\mathcal{O})$ as the group acting on X;
– S a set of ideals, whose norms are small Elkies primes in $\mathcal{O}$.

The graph $\mathcal{G}(G, S, X)$ is thus an isogeny graph, composed of many isogeny cycles (one for the norm of each prime in S) superimposed on the vertex set $\mathrm{Ell}_q(\mathcal{O})$. It is connected if S generates $\mathcal{C}(\mathcal{O})$. Walks in $\mathcal{G}(G, S, X)$ are called *isogeny walks*.

We compute the action of an ideal $\mathfrak{s}$ (a single *isogeny step*) on an $x \in \mathrm{Ell}_q(\mathcal{O})$ by choosing a representative curve E with $x = j(E)$, and computing an isogeny $\phi_{\mathfrak{s}} : E \to E'$ from E corresponding to $\mathfrak{s}$; the resulting vertex is $\mathfrak{s} \cdot x = j(E')$. The action of an isogeny walk $(\mathfrak{s}_i)_i$ is then evaluated as the sequence of isogeny steps $\phi_{\mathfrak{s}_i}$. Algorithms for these operations are given in the next subsection.

Using this "smooth" representation of elements in $\mathcal{C}(\mathcal{O})$ as isogeny walks lets us avoid computing $\mathcal{C}(\mathcal{O})$ and $\mathrm{Ell}_q(\mathcal{O})$, and avoid explicit ideal class arithmetic; only isogenies between elliptic curves are computed. In practice, we re-use the elliptic curve E' from one step as the E in the next; but we emphasize that when isogeny walks are used for Diffie–Hellman, the resulting public keys and shared secrets are not the final elliptic curves, but their j-invariants.

3.2 Computing isogeny walks

Since $\mathcal{C}(\mathcal{O})$ is commutative, we can break isogeny walks down into a succession of walks corresponding to powers of single primes $\mathfrak{s} = (\ell, \pi - \lambda)$; that is, repeated applications of the isogenies $\phi_{\mathfrak{s}}$. Depending on $\mathfrak{s}$, we will compute each sequence of $\phi_{\mathfrak{s}}$ using one of two different methods:

– Algorithm 5 (ElkiesWalk) uses Algorithm 3 (ElkiesFirstStep) followed by a series of calls to Algorithm 4 (ElkiesNextStep), both which use the *modular polynomial* $\Phi_\ell(X, Y)$. This approach works for any $\mathfrak{s}$.
– Algorithm 7 (VéluWalk) uses a series of calls to Algorithm 6 (VéluStep). This approach, which uses torsion points on E, can only be applied when λ satisfies certain properties.

Rostovtsev and Stolbunov only used analogues of Algorithms 3 and 4. The introduction of VéluStep, inspired by SIDH and related protocols (and now a key ingredient in the CSIDH protocol [11]), speeds up our protocol by a considerable factor; this is the main practical contribution of our work.

Elkies steps. Algorithms 3 and 4 compute single steps in the ℓ-isogeny graph. Their correctness follows from the definition of the modular polynomial Φ_ℓ: a cyclic ℓ-isogeny exists between two elliptic curves E and E' if and only if $\Phi_\ell(j(E), j(E')) = 0$ (see [58, §6] and [23, §3] for the relevant theory). One may use the classical modular polynomials here, or alternative, lower-degree modular polynomials (Atkin polynomials, for example) with minimal adaptation to the algorithms. In practice, Φ_ℓ is precomputed and stored: several publicly available databases exist (see [41] and [66,7,8], for example).

Given a j-invariant $j(E)$, we can compute its two neighbours in the ℓ-isogeny graph by evaluating $P(X) = \Phi_\ell(j(E), X)$ (a polynomial of degree $\ell + 1$), and

Algorithm 3: ElkiesFirstStep

Input: $E \in \mathrm{Ell}_q(\mathcal{O})$; (ℓ, λ) encoding $\mathfrak{s} = (\pi - \lambda, \ell)$
Output: $j(\mathfrak{s} \cdot E)$

$P \leftarrow \Phi_\ell(X, j(E))$
$\{j_1, j_2\} \leftarrow \textsc{Roots}(P, \mathbb{F}_q)$
$K \leftarrow \textsc{KernelPolynomial}(\textsc{Isogeny}(E, j_1, \ell))$ `// e.g. BMSS algorithm`
$Q \leftarrow$ a nonzero point in K `// e.g.` $(x, y) \in E(\mathbb{F}_q[x, y]/(y^2 - f_E(x), K(x)))$
if $\pi(Q) = [\lambda]Q$ **then**
  return j_1
else
  return j_2

Algorithm 4: ElkiesNextStep

Input: (ℓ, λ) encoding $\mathfrak{s} = (\pi - \lambda, \ell)$; $(j_0, j_1) = (j(E), j(\mathfrak{s} \cdot E))$ for E in $\mathrm{Ell}_q(\mathcal{O})$
Output: $j(\mathfrak{s} \cdot \mathfrak{s} \cdot E)$

$P \leftarrow \Phi_\ell(X, j_1)/(X - j_0)$
$j_2 \leftarrow \textsc{Root}(P, \mathbb{F}_q)$ `// It is unique`
return j_2

Algorithm 5: ElkiesWalk

Input: $E \in \mathrm{Ell}_q(\mathcal{O})$; (ℓ, λ) encoding $\mathfrak{s} = (\pi - \lambda, \ell)$; $k \geq 1$
Output: $\mathfrak{s}^k \cdot E$

$j_0 \leftarrow j(E)$
$j_1 \leftarrow \textsc{ElkiesFirstStep}(E, (\ell, \lambda))$
for $2 \leq i \leq k$ **do**
  $(j_0, j_1) \leftarrow (j_1, \textsc{ElkiesNextStep}((\ell, \lambda), (j_0, j_1)))$
$E_R \leftarrow \textsc{EllipticCurveFromJInvariant}(j_1)$
if ***not*** $\textsc{CheckTrace}(E_R, t)$ **then**
  $E_R \leftarrow \textsc{QuadraticTwist}(E_R)$
return E_R

Algorithm 6: VéluStep

Input: $E \in \mathrm{Ell}_q(\mathcal{O})$; (ℓ, λ) encoding $\mathfrak{s} = (\pi - \lambda, \ell)$; $r > 0$; $C_r = \#E(\mathbb{F}_{q^r})$
Output: $\mathfrak{s} \cdot E$

repeat
  $P \leftarrow \textsc{Random}(E(\mathbb{F}_{q^r}))$
  $Q \leftarrow [C_r/\ell]P$
until $Q \neq 0_E$
$K \leftarrow \prod_{i=0}^{(\ell-1)/2}(X - x([i]Q))$ `// Kernel polynomial of isogeny`
$E_R \leftarrow \textsc{IsogenyFromKernel}(E, K)$ `// Apply Vélu's formulæ`
return E_R

Algorithm 7: VéluWalk

Input: $E \in \mathrm{Ell}_q(\mathcal{O})$; (ℓ, λ) encoding $\mathfrak{s} = (\ell, \pi - \lambda)$; $k \geq 1$
Output: $\mathfrak{s}^k \cdot E$

1 $r \leftarrow \textsc{Order}(\lambda, \ell)$ `// Precompute and store for each` (ℓ, λ)
2 $C_r \leftarrow \#E(\mathbb{F}_{q^r})$ `// Precompute and store for each` r
3 **for** $1 \leq i \leq k$ **do**
4 $\quad E \leftarrow \textsc{VéluStep}(E, (\ell, \lambda), r, C_r)$
5 **return** E

then computing its two roots in $\mathbb{F}_q$. Using a Cantor–Zassenhaus-type algorithm, this costs $\tilde{O}(\ell \log q)$ $\mathbb{F}_q$-operations.

We need to make sure we step towards the neighbour in the correct direction. If we have already made one such step, then this is easy: it suffices to avoid backtracking. Algorithm 4 (ElkiesNextStep) does this by removing the factor corresponding to the previous j-invariant in Line 4; this algorithm can be used for all but the first of the steps corresponding to $\mathfrak{s}$.

It remains to choose the right direction in the first step for $\mathfrak{s} = (\ell, \pi - \lambda)$. In Algorithm 3 we choose one of the two candidates for $\phi_{\mathfrak{s}}$ arbitrarily, and compute its kernel polynomial. This costs $\tilde{O}(\ell)$ $\mathbb{F}_q$-operations using the Bostan–Morain–Salvy–Schost algorithm [6] with asymptotically fast polynomial arithmetic. We then compute an element Q of $\ker \phi_{\mathfrak{s}}$ over an extension of $\mathbb{F}_q$ of degree at most $\frac{\ell-1}{2}$, then evaluate $\pi(Q)$ and $[\lambda]Q$. If they match, then we have chosen the right direction; otherwise we take the other root of $P(X)$.

Algorithm 5 (ElkiesWalk) combines these algorithms to compute the iterated action of $\mathfrak{s}$. Line 5 ensures that the curve returned is the the correct component of the ℓ-isogeny graph. Both ElkiesFirstStep and ElkiesNextStep cost $\tilde{O}(\ell \log q)$ $\mathbb{F}_q$-operations, dominated by the calculation of the roots of $P(X)$.

Vélu steps. For some ideals $\mathfrak{s} = (\ell, \pi - \lambda)$, we can completely avoid modular polynomials, and the costly computation of their roots, by constructing $\ker \phi_{\mathfrak{s}}$ directly from ℓ-torsion points. Let r be the order of λ modulo ℓ; then $\ker \phi_{\mathfrak{s}} \subseteq E(\mathbb{F}_{q^r})$. If r is not a multiple of the order of the other eigenvalue μ of π on $E[\ell]$, then $E[\ell](\mathbb{F}_{q^r}) = \ker \phi_{\mathfrak{s}}$. Algorithm 6 (VéluStep) exploits this fact to construct a generator Q of $\ker \phi_{\mathfrak{s}}$ by computing a point of order ℓ in $E(\mathbb{F}_{q^r})$. The roots of the kernel polynomial of $\phi_{\mathfrak{s}}$ are then $x(Q), \ldots, x([(\ell-1)/2]Q)$.

Constructing a point Q of order ℓ in $E(\mathbb{F}_{q^r})$ is straightforward: we take random points and multiply by the cofactor C_r/ℓ, where $C_r := \#E(\mathbb{F}_{q^r})$. Each trial succeeds with probability $1 - 1/\ell$. Note that C_r can be easily (pre)computed from the Frobenius trace t: if we write $C_r = q - t_r + 1$ for $r > 0$ (so $t_1 = t$) and $t_0 = 2$, then the t_r satisfy the recurrence $t_r = t \cdot t_{r-1} - q \cdot t_{r-2}$.

We compute the quotient curve at Line 6 with Vélu's formulæ [69] in $O(\ell)$ $\mathbb{F}_q$-operations. Since $\log C_r \simeq r \log q$, provided $\ell = O(\log q)$, the costly step in Algorithm 6 is the scalar multiplication at Line 3, which costs $\tilde{O}(r^2 \log q)$ $\mathbb{F}_q$-operations.

Comparing the costs. To summarize:

- Elkies steps cost $\tilde{O}(\ell \log q)$ $\mathbb{F}_q$-operations;
- Vélu steps cost $\tilde{O}(r^2 \log q)$ $\mathbb{F}_q$-operations, where r is the order of λ in $\mathbb{Z}/\ell\mathbb{Z}$.

In general $r = O(\ell)$, so Elkies steps should be preferred. However, when r is particularly small (and not a multiple of the order of the other eigenvalue), a factor of ℓ can be saved using Vélu steps. The value of r directly depends on λ, which is in turn determined by $\#E(\mathbb{F}_p) \bmod \ell$. Thus, we see that better Step performances depend on the ability to find elliptic curves whose order satisfies congruence conditions modulo small primes. Unfortunately, we can only achieve this partially (see Section 4), so the most efficient solution is to use Vélu steps when we can, and Elkies steps for some other primes.

In practice, Algorithm 6 can be improved by using elliptic curve models with more efficient arithmetic. In our implementation (see Section 6), we used x-only arithmetic on Montgomery models [51,17], which also have convenient Vélu formulæ [16,56]. Note that we can also avoid computing y-coordinates in Algorithm 3 at Line 5 if $\lambda \neq \pm\mu$: this is the typical case for Elkies steps, and we used this optimization for all Elkies primes in our implementation.

Remark 1. Note that, in principle, Algorithm 6, can only be used to walk in one direction $\mathfrak{s}_\lambda = (\ell, \pi - \lambda)$, and not in the opposite one $\mathfrak{s}_\mu = (\ell, \pi - \mu)$. Indeed we have assumed that $E[\mathfrak{s}_\lambda]$ is in $E(\mathbb{F}_{q^r})$, while $E[\mathfrak{s}_\mu]$ is not. However, switching to a quadratic twist $\tilde{E}$ of E over $\mathbb{F}_{q^r}$ changes the sign of the Frobenius eigenvalues, thus it may happen that $\tilde{E}[\mathfrak{s}_{-\mu}]$ is in $\tilde{E}(\mathbb{F}_{q^r})$, while $\tilde{E}[\mathfrak{s}_{-\lambda}]$ is not. It is easy to force this behavior by asking that $p \equiv -1 \pmod{\ell}$, indeed then $\lambda = -1/\mu$.

For these eigenvalue pairs we can thus walk in both directions using Vélu steps at no additional cost, following either the direction λ on E, or the direction $-\mu$ on a twist. In Algorithm 6, only the curve order and the random point sampling need to be modified when using quadratic twists.

3.3 Sampling isogeny walks for key exchange

We now describe how keys are generated and exchanged in our protocol. Since the cost of the various isogeny walks depends on the ideals chosen, we will use adapted, or *skewed*, smooth representations when sampling elements in $\mathcal{C}(\mathcal{O})$ in order to minimize the total computational cost of a key exchange.

We take a (conjectural) generating set for $\mathcal{C}(\mathcal{O})$ consisting of ideals over a set S of small Elkies primes, which we partition into three sets according to the step algorithms to be used. We maintain three lists of tuples encoding these primes:

S_{VV} is a list of tuples (ℓ, λ, μ) such that the ideal $(\ell, \pi - \lambda)$ and its inverse $(\ell, \pi - \mu)$ are *both* amenable to VéluStep.

S_{VE} is a list of tuples (ℓ, λ) such that $(\ell, \pi - \lambda)$ is amenable to VéluStep but its inverse $(\ell, \pi - \mu)$ is *not*.

S_{EE} is a list of tuples (ℓ, λ, μ) such that *neither* $(\ell, \pi - \lambda)$ nor $(\ell, \pi - \mu)$ are amenable to VéluStep.

In S_{VV} and S_{EE}, the labelling of eigenvalues as λ and μ is fixed once and for all (that is, the tuples (ℓ, λ, μ) and (ℓ, μ, λ) do not both appear). This fixes directions in each of the ℓ-isogeny cycles. Looking back at Figure 1, for ℓ associated with S_{EE} and S_{VV}, both directions in the ℓ-isogeny graph will be available for use in walks; for S_{VE}, only the Vélu direction will be used.

Each secret key in the cryptosystem is a walk in the isogeny graph. Since the class group $\mathcal{C}(\mathcal{O})$ is commutative, such a walk is determined by the multiplicities of the primes $\mathfrak{s}$ that appear in it. Algorithm 8 (KeyGen) therefore encodes private-key walks as *exponent vectors*, with one integer exponent for each tuple in S_{VV}, S_{VE}, and S_{EE}. For a tuple (ℓ, λ, μ),

- a positive exponent k_ℓ indicates a walk of k_ℓ ℓ-isogeny steps in direction λ;
- a negative exponent $-k_\ell$ indicates k_ℓ ℓ-isogeny steps in direction μ.

For the tuples (ℓ, λ) in S_{VE}, where we do not use the slower μ-direction, we only only allow non-negative exponents. We choose bounds M_ℓ on the absolute value of the exponents k_ℓ so as to minimize the total cost of computing isogeny walks, while maintaining a large keyspace. As a rule, the bounds will be much bigger for the primes in S_{VV} and S_{VE}, where Vélu steps can be applied.

The public keys are j-invariants in $\mathbb{F}_q$, so they can be stored in $\log_2 q$ bits; the private keys are also quite compact, but their precise size depends on the number of primes ℓ and the choice of exponent bounds M_ℓ, which is a problem we will return to in Section 6.

Algorithm 8: KeyGen for cryptosystems in the isogeny graph on $\mathrm{Ell}_q(\mathcal{O})$ with walks based on S, and initial curve E_0. The ideal lists S_{EE}, S_{VV}, and S_{VE}, and the walk bounds M_ℓ, are system parameters.

Input: ()
Output: A secret key $(k_\ell)_{\ell \in S}$ and the corresponding public key $j(E)$

$E \leftarrow E_0$
for $(\ell, \lambda, \mu) \in S_{EE}$ **do**
  $k_\ell \leftarrow \textsc{Random}([-M_\ell, M_\ell])$
  if $k_\ell \geq 0$ **then** $\nu \leftarrow \lambda$
  else $\nu \leftarrow \mu$
  $E \leftarrow \textsc{ElkiesWalk}(E, (\ell, \nu), |k_\ell|)$
for $(\ell, \lambda, \nu) \in S_{VV}$ **do**
  $k_\ell \leftarrow \textsc{Random}([-M_\ell, M_\ell])$
  if $k_\ell \geq 0$ **then** $\nu \leftarrow \lambda$
  else $\nu \leftarrow \mu$
  $E \leftarrow \textsc{VéluWalk}(E, (\ell, \nu), |k_\ell|)$
for $(\ell, \lambda) \in S_{VE}$ **do**
  $k_\ell \leftarrow \textsc{Random}([0, M_\ell])$
  $E \leftarrow \textsc{VéluWalk}(E, (\ell, \lambda), k_\ell)$
return $((k_\ell)_{\ell \in S}, j(E))$

Algorithm 9 completes a Diffie–Hellman key exchange by applying a combination of Elkies and Vélu walks (Algorithms 5 and 7, respectively).

Algorithm 9: DH for the isogeny graph on $\mathrm{Ell}_q(\mathcal{O})$ with primes in S. The ideal lists S_{EE}, S_{VV}, and S_{VE}, and the walk bounds M_ℓ, are system parameters. Public key validation is not included here, but (if desired) should be carried out as detailed in Section 5.4.

Input: A private key $k_A = (k_{A,\ell})_{\ell \in S}$ corresponding to a walk $(\mathfrak{s}_1, \ldots, \mathfrak{s}_n)$, and a public key $j_B = j(E_B)$ for $E_B \in \mathrm{Ell}_q(O)$
Output: A shared secret $j(\prod_{i=1}^{n} \mathfrak{s}_i \cdot E_B)$

$E \leftarrow \textsc{EllipticCurveFromJInvariant}(j_B)$
if ***not*** $\textsc{CheckTrace}(E, t)$ **then**
  $E \leftarrow \textsc{QuadraticTwist}(E)$
for $(\ell, \lambda, \mu) \in S_{EE}$ **do**
  if $k_{A,\ell} \geq 0$ **then** $\nu \leftarrow \lambda$
  else $\nu \leftarrow \mu$
  $E \leftarrow \textsc{ElkiesWalk}(E, (\ell, \nu), |k_{A,\ell}|)$
for $(\ell, \lambda, \mu) \in S_{VV}$ **do**
  if $k_{A,\ell} \geq 0$ **then** $\nu \leftarrow \lambda$
  else $\nu \leftarrow \mu$
  $E \leftarrow \textsc{VéluWalk}(E, (\ell, \nu), |k_{A,\ell}|)$
for $(\ell, \lambda) \in S_{VE}$ **do**
  $E \leftarrow \textsc{VéluWalk}(E, (\ell, \lambda), k_{A,\ell})$
return $j(E)$

4 Public parameter selection

It is evident that the choice of public parameters has a heavy impact on the execution time: smaller Elkies primes, and smaller multiplicative orders of the Frobenius eigenvalues, will lead to better performance. Since all of this information is contained in the value of $\#E(\mathbb{F}_q)$, we now face the problem of constructing ordinary elliptic curves of prescribed order modulo small primes. Unfortunately, and in contrast with the supersingular case, no polynomial-time method to achieve this is known in general: the CM method [2,64], which solves this problem when the corresponding class groups are small, is useless in our setting (see Section 5).

In this section we describe how to use the Schoof–Elkies–Atkin (SEA) point counting algorithm with early abort, combined with the use of certain modular curves, to construct curves whose order satisfies some constraints modulo small primes. This is faster than choosing curves at random and computing their orders completely until a convenient one is found, but it still does not allow us to use the full power of Algorithm VéluStep.

Early-abort SEA. The SEA algorithm [58,52] is the state-of-the-art point-counting algorithm for elliptic curves over large-characteristic finite fields. In order to compute $N = \#E(\mathbb{F}_p)$, it computes the value of N modulo a series of small Elkies primes ℓ, before combining the results via the CRT to get the true value of N.

Cryptographers are usually interested in generating elliptic curves of prime or nearly prime order, and thus without small prime factors. While running SEA on random candidate curves, one immediately detects if $N \equiv 0 \pmod{\ell}$ for the small primes ℓ; if this happens then the SEA execution is aborted, and restarted with a new curve.

Here, the situation is the opposite: we *want* elliptic curves whose cardinality has many small prime divisors. To fix ideas, we choose the 512-bit prime

$$p := 7 \left(\prod_{2 \leq \ell \leq 380,\ \ell \text{ prime}} \ell \right) - 1 \,.$$

Then, according to Remark 1, Algorithm VéluStep can be used for ℓ-isogenies in both directions for any prime $\ell \leq 380$, as soon as the order of its eigenvalues is small enough. We now proceed as follows:

- Choose a smoothness bound B (we used $B = 13$).
- Pick elliptic curves E at random in $\mathbb{F}_p$, and use the SEA algorithm, aborting when any $\ell \leq B$ with $\#E(\mathbb{F}_p) \not\equiv 0 \pmod{\ell}$ is found.
- For each E which passed the tests above, complete the SEA algorithm to compute $\#E(\mathbb{F}_p)$, and estimate the key exchange running time using this curve as a public parameter (see Section 6).
- The "fastest" curves now give promising candidates for $\#E(\mathbb{F}_p)$.

In considering the efficiency of this procedure, it is important to remark that very few curves will pass the early-abort tests. The bound B should be chosen to balance the overall cost of the first few tests with that of the complete SEA algorithm for the curves which pass them. Therefore, its value is somewhat implementation-dependent.

Finding the maximal order. Once a "good" curve E has been computed, we want to find a curve E_0 having the same number of points, but whose endomorphism ring is maximal, and to ensure that its discriminant is a large integer. Therefore, we attempt to factor the discriminant Δ_π of $\mathbb{Z}[\pi]$: if it is squarefree, then E already has maximal endomorphism ring, and in general the square factors of Δ_π indicate which ascending isogenies have to be computed in order to find E_0.

Remark 2. Factoring random 512-bit integers is not hard in general, and discriminants of quadratic fields even tend to be slightly smoother than random integers. If a discriminant fails to be completely factored, a conservative strategy would be to discard it, but ultimately undetected large prime-square factors do not present a security issue because computing the possible corresponding large-degree isogenies is intractable (see Section 5).

Using the modular curve $X_1(N)$. Since we are looking for curves with smooth cardinalities, another improvement to this procedure is available: instead of choosing elliptic curves uniformly at random, we pick random candidates using an equation for the modular curve $X_1(N)$ [65], which guarantees the existence of a rational N-torsion point on the sampled elliptic curve. This idea is used in the procedure of selecting elliptic curves in the Elliptic Curve Method for factoring [70,71]. In our implementation we used $N = 17$, and also incorporated the existence test in [54] for Montgomery models for the resulting elliptic curves.

Results. We implemented this search using the Sage computer algebra system. Our experiments were conducted on several machines running Intel Xeon E5520 processors at 2.27GHz. After 17,000 hours of CPU time, we found the Montgomery elliptic curve $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with p as above, and

$$\begin{aligned} A = {} & 10861338504649280383859950140772947007703646408372831934324660566888732797778932142488253565145603672591 \\ & 94460221057142376768924003282944443946924252186417 1 . \end{aligned}$$

The trace of Frobenius t of E is

$$-14718955017252810490042213191226689859938755551292423176210772843254195297929 0 .$$

There is a rational ℓ-torsion point on E, or its quadratic twist, for each ℓ in

$$\{3, 5, 7, 11, 13, 17, 103, 523, 821, 947, 1723\} ;$$

each of these primes is Elkies. Furthermore, $\mathrm{End}(E)$ is the maximal order, and its discriminant is a 511-bit integer that has the following prime factorization:

$$\begin{aligned} -2^3 \cdot & 20507 \cdot 67429 \cdot 1171823817029067 7 \cdot 12248034502305872059 \\ & \cdot 60884358188204745129468762751254728712569 \\ & \cdot 68495197685926430905162211241300486171895491480444062860794276603493 . \end{aligned}$$

In Section 6, we discuss the practical performance of our key-exchange protocol using these system parameters. Other proposals for parameters are given in [38].

5 Security

We now address the security of the CRS primitive, and derived protocols. Intuitively, these systems rely on two assumptions:

1. given two curves E and E' in $\mathrm{Ell}_q(\mathcal{O})$, it is hard to find a (smooth degree) isogeny $\phi : E \to E'$; and
2. the distribution on $\mathrm{Ell}_q(\mathcal{O})$ induced by the random walks sampled in Algorithm 8 is computationally undistinguishable from the uniform distribution.

We start by reviewing the known attacks for the first problem, both in the classical and the quantum setting. Then, we formalize security assumptions and give security proofs against passive adversaries. Finally, we discuss key validation and protection against active adversaries.

5.1 Classical attacks

We start by addressing the following, more general, problem:

Problem 1. Given two ordinary elliptic curves E, E' defined over a finite field $\mathbb{F}_q$, such that $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$, find an isogeny walk $(\phi_i)_{1 \leq i \leq n}$ such that $\phi_n \circ \cdots \circ \phi_1(E) = E'$.

This problem was studied for some time before the emergence of isogeny-based cryptography [28,30,27], because of its applications to conventional elliptic curve cryptography [30,67,36]. The algorithm with the best asymptotic complexity is due to Galbraith, Hess and Smart [30]. It consists of three stages:

Stage 0. Use walks of *ascending* isogenies to reduce to the case where $\mathrm{End}(E) \cong \mathrm{End}(E')$ is the maximal order.

Stage 1. Start two random walks of horizontal isogenies from E and E'; detect the moment when they collide using a Pollard-rho type of algorithm.

Stage 2. Reduce the size of the obtained walk using index-calculus techniques.

To understand Stage 0, recall that all isogenous elliptic curves have the same order, and thus the same trace t of the Frobenius endomorphism π. We know that $\mathrm{End}(E)$ is contained in the ring of integers $\mathcal{O}_K$ of $K = \mathbb{Q}(\sqrt{\Delta_\pi})$, where $\Delta_\pi = t^2 - 4q$ is the Frobenius discriminant. As before we write $\Delta_\pi = d^2 \Delta_K$, where Δ_K is the discriminant of $\mathcal{O}_K$; then for any $\ell \mid d$, the ℓ-isogeny graph of E contains *ascending* and *descending* ℓ-isogenies; these graphs are referred to as *volcanoes* [25] (see Figure 3). Ascending isogenies go from curves with smaller endomorphism rings to curves with larger ones, and take us to a curve with $\mathrm{End}(E) \simeq \mathcal{O}_K$ in $O(\log d)$ steps; they can be computed efficiently using the algorithms of [40,25,34,21]. Assuming[8] all prime factors of d are in $O(\log q)$, we can therefore compute Stage 0 in time polynomial in $\log q$.

The set $\mathrm{Ell}_q(\mathcal{O}_K)$ has the smallest size among all sets $\mathrm{Ell}_q(\mathcal{O})$ for $\mathcal{O} \subset \mathcal{O}_K$, so it is always interesting to reduce to it. This justifies using curves with maximal endomorphism ring in the definition of the protocol in Section 3. We remark that when Δ_π is square-free, $\mathbb{Z}[\pi]$ is the maximal order, and this condition is automatically satisfied.

The collision search in Stage 1 relies on the birthday paradox, and has a complexity of $O(\sqrt{h(\mathcal{O}_K)})$. It is known that, on average, $h(\mathcal{O}_K) \approx 0.461 \cdots \sqrt{|\Delta_K|}$ (see [14, 5.10]), and, assuming the extended Riemann hypothesis, we even have a lower bound (see [46])

$$h(\mathcal{O}_K) \geq 0.147 \cdots \frac{(1 + o(1))\sqrt{|\Delta_K|}}{\log\log|\Delta_K|}.$$

Since $\Delta_K \sim q$, we expect Stage 1 to take time $O(q^{1/4})$, which justifies a choice of q four times as large as the security parameter. Unfortunately, class numbers

[8] This is typical for isogeny-based protocols. No counter-example has ever been constructed.

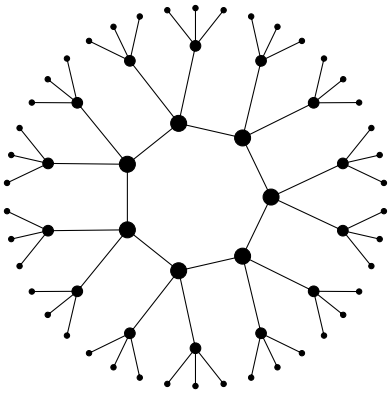

Fig. 3. 3-isogeny graph (*volcano*) containing the curve with $j(E) = 607$ over $\mathbb{F}_{6007}$. A larger vertex denotes a larger endomorphism ring.

are notoriously difficult to compute, the current record being for a discriminant of 300 bits [4]. Computing class numbers for ~ 500-bit discriminants seems to be expensive, albeit feasible; thus, we can only rely on these heuristic arguments to justify the security of our proposed parameters.

Finally, Stage 2, which outputs an isogeny walk of size polynomial in $\log q$, has a complexity bounded by that of Stage 1. It therefore has no impact on our security estimates.

Remark 3. The Cohen–Lenstra heuristic [15] predicts that the odd part of $\mathcal{C}(\mathcal{O}_K)$ is cyclic with overwhelming probability, and other heuristics [32] indicate that $h(\mathcal{O}_K)$ is likely to have a large prime factor. However, since there is no known way in which the group structure of $\mathcal{C}(\mathcal{O}_K)$ can affect the security of our protocol, we can disregard this matter. No link between the group structure of $E(\mathbb{F}_q)$ itself and the security is known, either.

5.2 Quantum attacks

On a quantum computer, an attack with better asymptotic complexity is given by Childs, Jao and Soukharev in [12]. It consists of two algorithms:

1. A (classical) algorithm that takes as input an elliptic curve $E \in \mathrm{Ell}_q(\mathcal{O})$ and an ideal $\mathfrak{a} \in \mathcal{C}(\mathcal{O})$, and outputs the curve $\mathfrak{a} \cdot E$;
2. A generic quantum algorithm for the dihedral hidden subgroup problem (dHSP), based upon previous work of Kuperberg [42,43] and Regev [55].

The ideal evaluation algorithm has sub-exponential complexity $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$. However, after a subexponential-time classical precomputation, any adversary can know the exact class group structure; in that case, this ideal evaluation step could possibly be performed in polynomial time (and non-negligible success probability) using LLL-based methods, as discussed in [63] and [18, §5].

The dHSP algorithm uses the ideal evaluation algorithm as a (quantum) black box, the number of queries depending on the variant. Childs–Jao–Soukharev gave two versions of this algorithm, Kuperberg's [42] and Regev's[55]. However, both are superseded by Kuperberg's recent work [43]: his new algorithm solves the dHSP in any abelian group of order N using $2^{O(\sqrt{\log N})}$ quantum queries and classical space, but only $O(\log N)$ quantum space. Given this estimate, we expect the bit size of q to grow at worst like the square of the security parameter.

Unfortunately, the analysis of Kuperberg's new algorithm is only asymptotic, and limited to N of a special form; it cannot be directly used to draw conclusions on concrete cryptographic parameters at this stage, especially since the value of the constant hidden by the $O()$ in the exponent is unclear. Thus, it is hard to estimate the impact of this attack at concrete security levels such as those required by NIST [53].

Nevertheless, we remark that the first version of Kuperberg's algorithm, as described in [55, Algorithm 5.1 and Remark 5.2] requires $O(2^{3\sqrt{\log N}} \log N)$ black-box queries and $\sim 2^{3\sqrt{\log N}}$ qubits of memory. Although the quantum memory requirements of this algorithm are rather high, we will take its query complexity as a crude lower bound for the complexity of Kuperberg's newer algorithm in the general case. Of course, this assumption is only heuristic, and should be validated by further study of quantum dHSP solvers; at present time, unfortunately, no precise statement can be made.

Table 1 thus proposes various parameter sizes, with associated numbers of quantum queries based on the observations above; we also indicate the estimated time to (classically) precompute the class group structure according to [4][9]. Whenever the quantum query complexity alone is enough to put a parameter in one of NIST's security categories [53], we indicate it in the table. We believe that using query complexity alone is a very conservative choice, and should give more than enough confidence in the post-quantum security properties of our scheme.

The system parameters we proposed in Section 4 correspond to the first line of Table 1, thus offering at least 56-bit of quantum security (and 128-bit classical security).

5.3 Security proofs

We now formalize the assumptions needed to prove the security of the key exchange protocol, and other derived protocols such as PKEs and KEMs, in various models. Given the similarity with the classical Diffie–Hellman protocol on a cyclic group, our assumptions are mostly modeled on those used in that context. Here we are essentially following the lead of Couveignes [18] and Stolbunov [62,63]. However, we take their analyses a step further by explicitly modeling the hardness of distinguishing random walks on Cayley graphs from the uniform distribution: this yields stronger proofs and a better separation of security concerns.

[9] Note however that computing the class group structure can be seen as an instance of the hidden subgroup problem, and thus can be solved in quantum polynomial time by Shor's algorithm.

$\log \Delta_K$	$\log h(\mathcal{O}_K)$	classical security	$L_{\lvert\Delta_K\rvert}(1/2, 1)$	quantum queries	NIST category
512	256	2^{128}	$2^{56.6}$	$> 2^{56}$	
688	344	2^{172}	$2^{67.0}$	$> 2^{64}$	1
768	384	2^{192}	$2^{71.4}$	$> 2^{67}$	1
1024	512	2^{256}	$2^{84.2}$	$> 2^{76}$	1
1656	828	2^{414}	$2^{110.8}$	$> 2^{96}$	3
3068	1534	2^{767}	$2^{156.9}$	$> 2^{128}$	5

Table 1. Suggested parameter sizes and associated classical security, class group computation time, and query complexity, using the heuristic estimations of Section 5.2.

For the rest of this section q is a prime power, $\mathcal{O}$ is an order in a quadratic imaginary field with discriminant $\Delta \sim q$, $\mathcal{C}(\mathcal{O})$ is the class group of $\mathcal{O}$, $\mathrm{Ell}_q(\mathcal{O})$ is the (non-empty) set of elliptic curves with complex multiplication by $\mathcal{O}$, and E_0 is a fixed curve in $\mathrm{Ell}_q(\mathcal{O})$. Finally, S is a set of ideals of $\mathcal{O}$ with norm polynomial in $\log q$, and σ is a probability distribution on the set S^* of isogeny walks (i.e. finite sequences of elements in S) used to sample secrets in the key exchange protocol. We write $x \xleftarrow{\sigma} X$ for an element taken from a set X according to σ, and $x \xleftarrow{R} X$ for an element taken according to the uniform distribution.

Our security proofs use four distributions on $\mathrm{Ell}_q(\mathcal{O})^3$:

$$\mathcal{G}_{q,\Delta} := \left\{ (\mathfrak{a} \cdot E_0, \mathfrak{b} \cdot E_0, \mathfrak{ab} \cdot E_0) \,\middle|\, \mathfrak{a}, \mathfrak{b} \xleftarrow{R} \mathcal{C}(\mathcal{O}) \right\},$$

$$\mathcal{W}_{q,\Delta,\sigma} := \left\{ \left((\mathfrak{a}_i)_i \cdot E_0, (\mathfrak{b}_j)_j \cdot E_0, (\mathfrak{a}_i)_i \cdot (\mathfrak{b}_j)_j \cdot E_0\right) \,\middle|\, (\mathfrak{a}_i)_i, (\mathfrak{b}_j)_j \xleftarrow{\sigma} S^* \right\},$$

$$\mathcal{R}_{q,\Delta,\sigma} := \left\{ \left((\mathfrak{a}_i)_i \cdot E_0, (\mathfrak{b}_i)_i \cdot E_0, E'\right) \,\middle|\, (\mathfrak{a}_i)_i, (\mathfrak{b}_i)_i \xleftarrow{\sigma} S^*,\ E' \xleftarrow{R} \mathrm{Ell}_q(\mathcal{O}) \right\},$$

$$\mathcal{U}_{q,\Delta} := \left\{ (E_a, E_b, E_{ab}) \,\middle|\, E_a, E_b, E_{ab} \xleftarrow{R} \mathrm{Ell}_q(\mathcal{O}) \right\}.$$

The assumption needed to prove security of the protocols is the hardness of a problem analogous to the classic Decisional Diffie–Hellman (DDH) problem.

Definition 1 (Isogeny Walk DDH (IW-DDH)). *Given a triplet of curves (E_a, E_b, E_{ab}) sampled with probability $\frac{1}{2}$ from $\mathcal{R}_{q,\Delta,\sigma}$ and $\frac{1}{2}$ from $\mathcal{W}_{q,\Delta,\sigma}$, decide from which it was sampled.*

We split this problem into two finer-grained problems. The first is that of distinguishing between commutative squares sampled uniformly at random and commutative squares sampled from the distribution σ.

Definition 2 (Isogeny Walk Distinguishing (IWD)). *Given a triplet of curves (E_a, E_b, E_{ab}) sampled with probability $\frac{1}{2}$ from $\mathcal{W}_{q,\Delta,\sigma}$ and $\frac{1}{2}$ from $\mathcal{G}_{q,\Delta}$, decide from which it was sampled.*

The second problem is a group-action analogue of DDH. It also appears in [18] under the name *vectorization*, and in [62,63] under the name DDHAP.

Definition 3 (Class Group Action DDH (CGA-DDH)). *Given a triplet of curves (E_a, E_b, E_{ab}) sampled with probability $\frac{1}{2}$ from $\mathcal{G}_{q,\Delta}$ and $\frac{1}{2}$ from $\mathcal{U}_{q,\Delta}$, decide from which it was sampled.*

We want to prove the security of protocols based on the primitive of Section 3 under the CGA-DDH and IWD assumptions combined. To do this we give a lemma showing that CGA-DDH and IWD together imply IW-DDH. The technique is straightforward: we use an IW-DDH oracle to solve both the CGA-DDH and IWD problems, showing that at least one of the two must be solvable with non-negligible advantage. The only technical difficulty lies in the fact that we need an efficient way to simulate the uniform distribution on $\mathrm{Ell}_q(\mathcal{O})$; for this, we use another Cayley graph on $\mathrm{Ell}_q(\mathcal{O})$, with a potentially larger edge set, that is proven in [36] to be an expander graph under the generalized Riemann hypothesis (GRH).

We let $\mathsf{Adv}^A_{\text{IW-DDH}}$ be the *advantage* of an adversary A against IW-DDH, defined as the probability that A answers correctly, minus 1/2:

$$2\mathsf{Adv}^A_{\text{IW-DDH}} = \Pr\left[A(\mathcal{R}_{q,\Delta,\sigma}) = 1\right] - \Pr\left[A(\mathcal{W}_{q,\Delta,\sigma}) = 1\right].$$

We define $\mathsf{Adv}^A_{\text{CGA-DDH}}$ and $\mathsf{Adv}^A_{\text{IWD}}$ similarly. Switching answers if needed, we can assume all advantages are positive. We let $\mathsf{Adv}_{\text{X}}(t)$ denote the maximum of $\mathsf{Adv}^A_{\text{X}}$ over all adversaries using at most t resources (running time, queries, etc.).

Lemma 1. *Assuming GRH, for q large enough and for any bound t on running time, and for any $\epsilon > 0$,*

$$\mathsf{Adv}_{\text{IW-DDH}}(t) \leq 2\mathsf{Adv}_{\text{IWD}}(t + \mathrm{poly}(\log q, \log \epsilon)) + \mathsf{Adv}_{\text{CGA-DDH}}(t) + \epsilon\,.$$

Proof (Sketch). We start with an adversary A for IW-DDH, and we construct two simulators S and T for CGA-DDH and IWD respectively.

- The simulator S simply passes its inputs to A, and returns A's response.
- The simulator T receives a triplet (E_a, E_b, E_{ab}) taken from $\mathcal{G}_{q,\Delta}$ or $\mathcal{W}_{q,\Delta,\sigma}$, and flips a coin to decide which of the two following actions it will do:
  - forward (E_a, E_b, E_{ab}) to A, and return the bit given by A; or
  - generate a random curve $E_c \in \mathrm{Ell}_q(\mathcal{O})$, forward (E_a, E_b, E_c) to A, and return the opposite bit to the one given by A.

The curve E_c must be sampled from a distribution close to uniform for the simulator T to work. The only way at our disposal to sample E_c uniformly would be to sample a uniform $\mathfrak{c} \in \mathcal{O}$ and take $E_c = \mathfrak{c} \cdot E_0$, but this would be too costly. Instead we use [36, Theorem 1.5], combined with standard results about random walks in expander graphs (for instance, an easy adaptation of the proof of [36, Lemma 2.1]), to sample E_c so that any curve in $\mathrm{Ell}_q(\mathcal{O})$ is taken with probability between $(1-\epsilon)/h(\mathcal{O})$ and $(1+\epsilon)/h(\mathcal{O})$, using only $\mathrm{poly}(\log q, \log \epsilon)$ operations.

We can consider this sampling as follows: with probability $1-\epsilon$, sample E_c uniformly, and with probability ϵ sample it from an unknown distribution.

Now, if T forwarded (E_a, E_b, E_{ab}) untouched, then we immediately get

$$2\mathsf{Adv}^{T}_{\text{IWD}} = \mathsf{Adv}^{A}_{\text{IW-DDH}} - \mathsf{Adv}^{S}_{\text{CGA-DDH}}\,;$$

if T forwarded (E_a, E_b, E_c), then we get

$$2\mathsf{Adv}^{T}_{\text{IWD}} \geq \mathsf{Adv}^{A}_{\text{IW-DDH}} - (1-\epsilon)\mathsf{Adv}^{S}_{\text{CGA-DDH}} - \epsilon\,.$$

Averaging over the two outcomes concludes the proof. □

Finally, we define an isogeny-walk analogue of the classic Computational Diffie–Hellman (CDH) problem for groups. Using the same techniques as above, we can prove the security of the relevant protocols based only on CGA-CDH and IWD, without the generalized Riemann hypothesis.

Definition 4 (Class Group Action CDH (CGA-CDH)). *Given $E_a = \mathfrak{a}\cdot E_0$ and $E_b = \mathfrak{b}\cdot E_0$ with $\mathfrak{a}, \mathfrak{b} \xleftarrow{R} \mathcal{C}(\mathcal{O})$, compute the curve $E_{ab} = \mathfrak{ab}\cdot E_0$.*

Stolbunov already proved the security of HHS Diffie–Hellman under the equivalent of CGA-DDH [62]. By repeating the same steps, we can prove the following theorem.

Theorem 1. *If the CGA-DDH and IWD assumptions hold, assuming GRH, the key-agreement protocol defined by Algorithms 8 and 9 is session-key secure in the authenticated-links adversarial model of Canetti and Krawczyk [10].*

Similarly, we can prove the IND-CPA security of the hashed El Gamal protocol derived from Algorithm 8 by replicating the classical techniques of, for instance, [29, 20.4.11].

Theorem 2. *Assuming CGA-CDH and IWD, the hashed El Gamal protocol derived from Algorithms 8 and 9 is IND-CPA secure in the random oracle model.*

A heuristic discussion of the IWD assumption. From its very definition, the IWD problem depends on the probability distribution σ we use to sample random walks in the isogeny graph. In this paragraph, we provide heuristic arguments suggesting that the IWD instances generated by Algorithm 9 are hard, provided

1. the keyspace size is at least $\sqrt{|\Delta_K|}$, and
2. S is *not too small*, i.e. the number of isogeny degrees used is in $\Omega(\log q)$.

Proving rapid mixing of isogeny walks with such parameters seems out of reach at present, even under number-theoretic hypotheses such as GRH. The best results available, like [36, Theorem 1.5] (used in the proof of Lemma 1), typically require isogeny degrees in $\Omega((\log q)^B)$ for some $B > 2$, and fully random walks that are not, for example, skewed towards smaller-degree isogenies.

However, numerical evidence suggests that these theoretical results are too weak. In [36, 7.2], it is asked whether an analogue of the previous theorem would be true with the sole constraint $B > 1$. In [30, Section 3], it is mentioned that many fewer split primes are needed to walk in the isogeny graph than theoretically expected. Practical evidence also suggests that the rapid mixing properties are not lost with skewed random walks: such walks are used in [27] to accelerate an algorithm solving Problem 1. We believe that these experiments can bring some evidence in favor of relying on the IWD assumptions with more aggressive parameters than those provided by GRH.

5.4 Key validation and active security

Modern practice in cryptography mandates the use of stronger security notions than IND-CPA. From the DLP assumption, it is easy to construct protocols with strong security against active adversaries. For example, it is well-known that the hashed El Gamal KEM achieves IND-CCA security in the random oracle model under various assumptions [1,49,19].

All of these constructions crucially rely on *key validation*: that is, Alice must verify that the public data sent by Bob defines valid protocol data (e.g., valid elements of a cyclic group), or abort if this is not the case. Failure to perform key validation may result in catastrophic attacks, such as small subgroup [45], invalid point [5], and invalid curve attacks [13].

In our context, key validation amounts to verifying that the curve sent by Bob really is an element of $\mathrm{Ell}_q(\mathcal{O}_K)$. Failure to do so exposes Alice to an *invalid graph attack*, where Bob forces Alice onto an isogeny class with much smaller discriminant, or different Elkies primes, and learns something on Alice's secret.

Fortunately, key validation is relatively easy for protocols based on the CRS primitive. All we need to check is that the received j-invariant corresponds to a curve with the right order, and with maximal endomorphism ring.

Verifying the curve order. Since we already know the trace t of the Frobenius endomorphism of all curves in $\mathrm{Ell}_q(\mathcal{O})$, we only need to check that the given E has order $q + 1 - t$. Assuming that E is cyclic, or contains a cyclic group of order larger than $4\sqrt{q}$, a very efficient randomized algorithm consists in taking a random point P and verifying that it has the expected order. This task is easy if the factorization of $q + 1 - t$ is known.

Concretely, the curve given in Section 4 has order

$$N = 2^2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13^2 \cdot 17 \cdot 103 \cdot 523 \cdot 821 \cdot 1174286389 \cdot (\text{432-bit prime}),$$

and its group structure is $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/\frac{N}{2}\mathbb{Z}$. To check that a curve is in the same isogeny class, we repeatedly take random points until we find one of order $N/2$.

Verifying the endomorphism ring level. The curve order verification proves that $\mathrm{End}(E)$ is contained between $\mathbb{Z}[\pi]$ and $\mathcal{O}_K$. We have already seen that there is only a finite number of possible rings: their indices in $\mathcal{O}_K$ must divide d where

$d^2 = \Delta_\pi/\Delta_K$. Ascending and descending isogenies connect curves with different endomorphism rings, thus we are left with the problem of verifying that E is on the crater of any ℓ-volcano for $\ell \mid d$. Assuming no large prime divides d, this check can be accomplished efficiently by performing random walks in the volcanoes, as described in [40, §4.2] or [25]. Note that if we choose Δ_π square-free, then the only possible endomorphism ring is $\mathcal{O}_K$, and there is nothing to be done.

Concretely, for the curve of Section 4 we have $\Delta_\pi/\Delta_K = 2^2$, so there are exactly two possible endomorphism rings. Looking at the action of the Frobenius endomorphism, we see that $\mathrm{End}(E) = \mathcal{O}_K$ if and only if $E[2] \simeq (\mathbb{Z}/2\mathbb{Z})^2$.

Example 2. Let p and $\mathcal{O}$ be as in Section 4. Suppose we are given the value

$$\alpha = \frac{6774653762400376370473362072511594555277819004969905295950079381173567249377 5}{1873774891388281639871569508662389079106938177131139788464911133375566528902 5}$$

in $\mathbb{F}_p$. It is claimed that α is in $\mathrm{Ell}_p(\mathcal{O})$; that is, it is a valid public key for the system with parameters defined in Section 4. Following the discussion above, to validate α as a public key, it suffices to exhibit a curve with j-invariant α, full rational 2-torsion, and a point of order $N/2$. Using standard formulæ, we find that the two $\mathbb{F}_p$-isomorphism classes of elliptic curves with j-invariant α are represented by the Montgomery curve $E_\alpha/\mathbb{F}_p : y^2 = x(x^2 + Ax + 1)$ with

$$A = \frac{4193809979435365668528368375333535083388979993941154941880421834369488741588 4}{6612599927969489869548583644605423817546131207840311667164101730172820139490 7}$$

and its quadratic twist E'_α. Checking the 2-torsion first, we have $E_\alpha[2](\mathbb{F}_p) \cong E'_\alpha[2](\mathbb{F}_p) \cong (\mathbb{Z}/2\mathbb{Z})^2$, because $A^2 - 4$ is a square in $\mathbb{F}_p$. Trying points on E_α, we find that $(23, \sqrt{23(23^2 + 23A + 1)})$ in $E_\alpha(\mathbb{F}_p)$ has exact order $N/2$. We conclude that $\mathrm{End}(E_\alpha) = \mathcal{O}$, so α is a valid public key. (In fact, E_α is connected to the initial curve by a single 3-isogeny step.)

Consequences for cryptographic constructions. Since both of the checks above can be done much more efficiently than evaluating a single isogeny walk, we conclude that key validation is not only possible, but highly efficient for protocols based on the CRS construction. This stands in stark contrast to the case of SIDH, where key validation is known to be problematic [31], and even conjectured to be has hard as breaking the system [68].

Thanks to this efficient key validation, we can obtain CCA-secure encryption from the CRS action without resorting to generic transforms such as Fujisaki–Okamoto [26], unlike the case of SIKE [3,33]. This in turn enables applications such as non-interactive key exchange, for which no practical post-quantum scheme was known prior to [11].

6 Experimental results

In order to demonstrate that our key-exchange protocol is usable at standard security levels, we implemented it in the Julia programming language. This proof of concept also allows us to estimate the cost of isogeny steps, which is needed in

the process of generating the initial curve in Section 4. We developed several Julia packages[10], built upon the computer algebra package Nemo [24]. Experiments were conducted on a Linux OS, with an Intel Core i7-5600U cpu at 2.60GHz, using Julia 0.6 and Nemo 0.7.3.

First, consider the time to compute one step for an ideal $\mathfrak{s} = (\ell, \pi - \lambda)$. Using Elkies steps, this is approximately the cost of finding the roots of the modular polynomial, which is roughly $0.017 \cdot \ell$ seconds in our implementation. Using Vélu steps, the cost of a step is approximately that of one scalar multiplication in $E(\mathbb{F}_{q^r})$. Table 2 lists per-step times in our implementation, for the extension degrees relevant to our parameters.

r	1	3	4	5	7	8	9
time (s)	0.02	0.10	0.15	0.24	0.8	1.15	1.3

Table 2. Timings for computing scalar multiplications in $E(\mathbb{F}_{p^r})$, the dominant operation in VéluStep (Algorithm 6), as a function of the extension degree r.

Using this data, finding efficient walk length bounds M_ℓ offering a sufficient keyspace size is easily seen to be an integer optimization problem. In order to find a satisfactory solution, we used the following heuristic procedure. Given a time bound T, let KeySpaceSize(T) be the keyspace size obtained when each M_ℓ is the greatest such that the *total time spent on ℓ-isogenies* is less than T. Then, if n denotes the (classical) security parameter, we look for the *least* T such that KeySpaceSize(T) $\geq 2^{2n}$ (according to Section 5), using dichotomic search. While the M_ℓ we obtain are most likely not the best possible, intuitively the outcome does not fall very far from the optimal.

In this way, we obtain a proposal for the walk length bounds M_ℓ to be used in Algorithm 8 along with the curve found in Section 4, to achieve 128-bit classical security. Table 3 lists the isogeny degrees amenable to Algorithm 6, each with the corresponding extension degree r (a star denotes that the twisted curve allows us to use both directions in the isogeny graph, as in Remark 1). Table 4 lists other primes for which we apply Algorithm 5.

r	M_ℓ	ℓ	r	M_ℓ	ℓ	r	M_ℓ	ℓ
1*	409	3, 5, 7, 11, 13, 17, 103	4	54	1013, 1181	8	7	881
1	409	523, 821, 947, 1723	5	34	31*, 61*, 1321	9	6	37*, 1693
3	81	19*, 661	7	10	29*, 71*, 547			

Table 3. Primes ℓ amenable to Algorithm 6 (VéluStep) for our candidate isogeny graph, with corresponding extension degrees r and proposed walk length bounds M_ℓ.

[10] The main code is available at `https://github.com/defeo/hhs-keyex/`, and the additional dependencies at `https://github.com/defeo/EllipticCurves.jl/` and `https://github.com/defeo/ClassPolynomials.jl/`.

M_ℓ	ℓ	M_ℓ	ℓ	M_ℓ	ℓ
20	23	6	73	2	157, 163, 167, 191, 193, 197, 223, 229
11	41	5	89	1	241, 251, 257, 277, 283, 293, 307
10	43	4	107, 109, 113	1	317, 349, 359
9	47	3	131, 151		

Table 4. Primes ℓ amenable to Algorithm 5 (ElkiesWalk) for our candidate isogeny graph, with proposed walk length bounds M_ℓ.

Using these parameters, we perform one isogeny walk in approximately 520 seconds. These timings are *worst-case* in the sense that the number of isogeny steps is taken to be exactly M_ℓ for each ℓ. This is about as fast as Stolbunov's largest parameter [62], which is for a prime of 428 bits and a keyspace of only 216 bits.

We stress the fact that this implementation is *not* optimised. Besides general gains on field arithmetic, optimised code could easily beat our proof-of-concept implementation on critical points of our algorithms, for instance the root finding steps in Algorithms 3 and 4.

For comparison, without using Algorithm 6, the total isogeny walk time would exceed 2000 seconds: our ideas thus bring an improvement by a factor of over 4 over the original protocol. A longer search for efficient public parameters would bring further improvement.

7 Conclusion

We have shown that the Couveignes–Rostovtsev–Stolbunov framework can be improved to become practical at standard pre- and post-quantum security levels; even more so if an optimized C implementation is made. The main obstacle to better performance is the difficulty of generating optimal system parameters: even with a lot of computational power, we cannot expect to produce ordinary curve parameters that allow us to use *only* Vélu steps. In this regard, the CSIDH protocol [11], which overcomes this problem using supersingular curves instead of ordinary ones, is promising.

One particularly nice feature of our protocol is its highly efficient key validation, which opens a lot of cryptographic doors. However, side-channel-resistant implementations remain an interesting problem for future work.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie–Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) Topics in Cryptology — CT-RSA 2001. pp. 143–158. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
2. Atkin, A.O.L., Morain, F.: Elliptic curves and primality proving. Math. Comp. **61**(203), 29–68 (1993). https://doi.org/10.2307/2152935

3. Azarderakhsh, R., Koziel, B., Campagna, M., LaMacchia, B., Costello, C., Longa, P., De Feo, L., Naehrig, M., Hess, B., Renes, J., Jalali, A., Soukharev, V., Jao, D., Urbanik, D.: Supersingular isogeny key encapsulation (2017), http://sike.org
4. Biasse, J.F., Jacobson, M.J., Silvester, A.K.: Security estimates for quadratic field based cryptosystems. In: Steinfeld, R., Hawkes, P. (eds.) Information Security and Privacy. pp. 233–247. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
5. Biehl, I., Meyer, B., Müller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. pp. 131–146. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
6. Bostan, A., Morain, F., Salvy, B., Schost, É.: Fast algorithms for computing isogenies between elliptic curves. Math. Comput. **77**(263), 1755–1778 (2008). https://doi.org/10.1090/S0025-5718-08-02066-8
7. Bröker, R., Lauter, K.E., Sutherland, A.V.: Modular polynomials via isogeny volcanoes. Math. Comput. **81**(278), 1201–1231 (2012). https://doi.org/10.1090/S0025-5718-2011-02508-1
8. Bruinier, J.H., Ono, K., Sutherland, A.V.: Class polynomials for nonholomorphic modular functions. Journal of Number Theory **161**, 204 – 229 (2016). https://doi.org/https://doi.org/10.1016/j.jnt.2015.07.002
9. Buchmann, J., Williams, H.C.: A key-exchange system based on imaginary quadratic fields. Journal of Cryptology **1**(2), 107–118 (Jun 1988). https://doi.org/10.1007/BF02351719
10. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045. Springer (2001)
11. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. Cryptology ePrint Archive, Report 2018/383 (2018), https://eprint.iacr.org/2018/383
12. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology **8**(1), 1–29 (2014)
13. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Designs, Codes and Cryptography **36**(1), 33–43 (Jul 2005). https://doi.org/10.1007/s10623-003-1160-8
14. Cohen, H.: A Course in Computational Algebraic Number Theory. Springer-Verlag New York, Inc., New York, NY, USA (1993)
15. Cohen, H., Lenstra, H.W.: Heuristics on class groups of number fields. In: Jager, H. (ed.) Number Theory Noordwijkerhout 1983. pp. 33–62. Springer Berlin Heidelberg, Berlin, Heidelberg (1984)
16. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology — ASIACRYPT 2017. pp. 303–329 (2017). https://doi.org/10.1007/978-3-319-70697-9_11
17. Costello, C., Smith, B.: Montgomery curves and their arithmetic. Journal of Cryptographic Engineering (2017). https://doi.org/10.1007/s13389-017-0157-6, https://hal.inria.fr/hal-01483768
18. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
19. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003). https://doi.org/10.1137/S0097539702403773
20. De Feo, L.: Mathematics of isogeny based cryptography. CoRR **abs/1711.04062** (2017), http://arxiv.org/abs/1711.04062

21. De Feo, L., Hugounenq, C., Plût, J., Schost, É.: Explicit isogenies in quadratic time in any characteristic. LMS Journal of Computation and Mathematics **19**(A), 267–282 (2016)
22. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. Des. Codes Cryptography **78**(2), 425–440 (2016). https://doi.org/10.1007/s10623-014-0010-1
23. Elkies, N.D.: Elliptic and modular curves over finite fields and related computational issues. In: Computational perspectives on number theory (Chicago, IL, 1995), AMS/IP Stud. Adv. Math., vol. 7, pp. 21–76. Amer. Math. Soc., Providence, RI (1998)
24. Fieker, C., Hart, W., Hofmann, T., Johansson, F.: Nemo/Hecke: Computer algebra and number theory packages for the Julia programming language. In: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation. pp. 157–164. ISSAC '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3087604.3087611
25. Fouquet, M., Morain, F.: Isogeny volcanoes and the SEA algorithm. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory Symposium. Lecture Notes in Computer Science, vol. 2369, pp. 47–62. Springer Berlin / Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45455-1_23
26. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 537–554. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
27. Galbraith, S., Stolbunov, A.: Improved algorithm for the isogeny problem for ordinary elliptic curves. Applicable Algebra in Engineering, Communication and Computing **24**(2), 107–131 (Jun 2013). https://doi.org/10.1007/s00200-013-0185-0
28. Galbraith, S.D.: Constructing isogenies between elliptic curves over finite fields. LMS Journal of Computation and Mathematics **2**, 118–138 (1999). https://doi.org/10.1112/S1461157000000097
29. Galbraith, S.D.: Mathematics of public key cryptography. Cambridge University Press (2012), `https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html`
30. Galbraith, S.D., Hess, F., Smart, N.P.: Extending the GHS Weil descent attack. In: Advances in cryptology — EUROCRYPT 2002 (Amsterdam), Lecture Notes in Computer Science, vol. 2332, pp. 29–44. Springer, Berlin (2002)
31. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Advances in Cryptology – ASIACRYPT 2016, Proceedings, Part I 22. pp. 63–91. Springer (2016)
32. Hamdy, S., Möller, B.: Security of cryptosystems based on class groups of imaginary quadratic orders. In: Okamoto, T. (ed.) Advances in Cryptology — ASIACRYPT 2000. pp. 234–247. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
33. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 341–371. Springer International Publishing, Cham (2017)
34. Ionica, S., Joux, A.: Pairing the volcano. Mathematics of Computation **82**(281), 581–603 (2013)
35. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) Post-Quantum Cryptography. Lecture Notes in Computer Science, vol. 7071, pp. 19–34. Springer Berlin / Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_2

36. Jao, D., Miller, S.D., Venkatesan, R.: Expander graphs based on GRH with an application to elliptic curve cryptography. Journal of Number Theory **129**(6), 1491–1504 (Jun 2009). https://doi.org/10.1016/j.jnt.2008.11.006
37. Jao, D., Soukharev, V.: A subexponential algorithm for evaluating large degree isogenies. In: ANTS IX: Proceedings of the Algorithmic Number Theory 9th International Symposium. Lecture Notes in Computer Science, vol. 6197, pp. 219–233. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14518-6_19
38. Kieffer, J.: Étude et accélération du protocole d'échange de clés de Couveignes–Rostovtsev–Stolbunov. Master's thesis, Inria Saclay & Université Paris VI (2017)
39. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.s., Park, C.: New public-key cryptosystem using braid groups. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. pp. 166–183. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
40. Kohel, D.R.: Endomorphism rings of elliptic curves over finite fields. Ph.D. thesis, University of California at Berkley (1996)
41. Kohel, D.R.: Echidna databases (2018), `http://iml.univ-mrs.fr/~kohel/dbs/`
42. Kuperberg, G.: A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. SIAM Journal of Computing **35**(1), 170–188 (2005)
43. Kuperberg, G.: Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In: Severini, S., Brandao, F. (eds.) 8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013). Leibniz International Proceedings in Informatics (LIPIcs), vol. 22, pp. 20–34. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013). https://doi.org/10.4230/LIPIcs.TQC.2013.20, `http://drops.dagstuhl.de/opus/volltexte/2013/4321`
44. Lang, S.: Elliptic Functions, Graduate texts in mathematics, vol. 112. Springer (1987)
45. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO '97. pp. 249–263. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
46. Littlewood, J.E.: On the class-number of the corpus $p(\sqrt{k})$. Proceedings of the London Mathematical Society **2**(1), 358–372 (1928)
47. Maze, G., Monico, C., Rosenthal, J.: Public key cryptography based on semigroup actions. Advances in Mathematics of Communications **1**(4), 489–507 (2007). https://doi.org/10.3934/amc.2007.1.489
48. Mestre, J.: La méthode des graphes. exemples et applications. In: Proceedings of the international conference on class numbers and fundamental units of algebraic number fields (Katata). pp. 217–242 (1986)
49. Michel Abdalla, M.B., Rogaway, P.: DHAES: An encryption scheme based on the Diffie–Hellman problem. Cryptology ePrint Archive, Report 1999/007 (1999), `https://eprint.iacr.org/1999/007`
50. Miret, J.M., Moreno, R., Sadornil, D., Tena, J., Valls, M.: An algorithm to compute volcanoes of 2-isogenies of elliptic curves over finite fields. Applied Mathematics and Computation **176**(2), 739–750 (2006)
51. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of computation **48**(177), 243–264 (1987)
52. Morain, F.: Calcul du nombre de points sur une courbe elliptique dans un corps fini: aspects algorithmiques. J. Théor. Nombres Bordeaux **7**(1), 255–282 (1995), `http://jtnb.cedram.org/item?id=JTNB_1995__7_1_255_0`, les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993)

53. National Institute of Standards and Technology: Announcing request for nominations for public-key post-quantum cryptographic algorithms (2016), https://www.federalregister.gov/d/2016-30615
54. Okeya, K., Kurumatani, H., Sakurai, K.: Elliptic curves with the Montgomery-form and their cryptographic applications. In: Imai, H., Zheng, Y. (eds.) Public Key Cryptography — PKC 2000. Lecture Notes in Computer Science, vol. 1751, pp. 238–257. Springer (2000). https://doi.org/10.1007/978-3-540-46588-1_17
55. Regev, O.: A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151 (Jun 2004), http://arxiv.org/abs/quant-ph/0406151
56. Renes, J.: Computing isogenies between Montgomery curves using the action of (0, 0). In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 229–247. Springer International Publishing (2018)
57. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145 (Apr 2006), http://eprint.iacr.org/2006/145/
58. Schoof, R.: Counting points on elliptic curves over finite fields. Journal de Théorie des Nombres de Bordeaux **7**(1), 219–254 (1995)
59. Silverman, J.H.: The arithmetic of elliptic curves, Graduate Texts in Mathematics, vol. 106. Springer-Verlag, New York (1992)
60. Silverman, J.H.: Advanced Topics in the Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, vol. 151. Springer (Jan 1994)
61. Stolbunov, A.: Reductionist security arguments for public-key cryptographic schemes based on group action. In: Mjølsnes, S.F. (ed.) Norsk informasjonssikkerhetskonferanse (NISK) (2009)
62. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Adv. Math. Commun. **4**(2) (2010)
63. Stolbunov, A.: Cryptographic schemes based on isogenies (2012)
64. Sutherland, A.V.: Accelerating the CM method. LMS J. Comput. Math. **15**, 172–204 (2012). https://doi.org/10.1112/S1461157012001015
65. Sutherland, A.V.: Constructing elliptic curves over finite fields with prescribed torsion. Mathematics of Computation **81**, 1131–1147 (2012)
66. Sutherland, A.V.: Modular polynomials (2018), https://math.mit.edu/~drew/ClassicalModPolys.html
67. Teske, E.: An elliptic curve trapdoor system. Journal of Cryptology **19**(1), 115–133 (Jan 2006). https://doi.org/10.1007/s00145-004-0328-3
68. Urbanik, D., Jao, D.: SoK: The problem landscape of SIDH. Cryptology ePrint Archive, Report 2018/336 (2018). https://doi.org/10.1145/3197507.3197516, https://eprint.iacr.org/2018/336
69. Vélu, J.: Isogénies entre courbes elliptiques. C. R. Acad. Sci. Paris Sér. A-B **273**, A238–A241 (1971)
70. Zimmermann, P., Dodson, B.: 20 years of ECM. In: Hess, F., Pauli, S., Pohst, M.E. (eds.) Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4076, pp. 525–542. Springer (2006). https://doi.org/10.1007/11792086_37
71. Zimmermann, P., et al.: GMP-ECM software (2018), http://ecm.gforge.inria.fr/