Faster algorithms in isogeny based cryptography

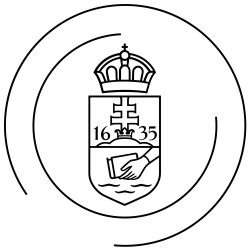
BSc Thesis

Made by:

Tot Bagi Márton
Mathematics BSc

Advisor:

Kutas Péter
ELTE Faculty of Informatics,
Department of Computer Algebra



Eötvös Loránd University
Faculty of Science

2025

Acknowledgements

I would like to thank Kutas Péter, who has been my supervisor for the past three years. He provided me with numerous opportunities and helped me overcome any administrative obstacles that came up. He always explained complex and deep concepts in elegant ways, enabling me to grab the essence of those ideas.

I would also like to thank Réka, who was always ready to provide support when needed. Without her I wouldn't have started writing this thesis until the week before the submission deadline.

Contents

1 Introduction

Cryptography is the practice of secure communication via insecure channels, integral to everyday Internet use, from online banking to social media. The security of cryptographic schemes relies on the hardness of specific algorithmic problems. Currently deployed schemes are based on problems that are believed to be hard for classical algorithms, such as factoring or the discrete logarithm problem. However, these may become vulnerable with the advent of large-scale quantum computers. This danger is not imminent, there is at least 10-15 years until this will pose a threat, but nonetheless it is important to build schemes, which are based on problems that cannot be solved even with a quantum computer.

Post-quantum cryptography aims to solve exactly this. One of the promising avenues to post-quantum schemes is isogeny based cryptography, which was first introduced in [8]

The first half of this thesis aims to introduce the necessary theoretical knowledge of elliptic curves, isogenies (Section 2), and cryptographic group actions (Section 3). The second half contains the optimizations I implemented in the PEARL-SCALLOP schemes precomputation (Section 4), and other algorithmic improvements which I have worked on since then (Section 5).

2 Elliptic curves

Definition 2.1. *An elliptic curve E defined over a field K (denoted by E/K) is the locus in $\mathbb{P}^2(\overline{K})$ of an equation*

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

with $a_i \in K$, given the locus is nonsingular as a curve and where $\overline{K}$ is the closure of K and $\mathbb{P}^2$ is the projective plane.

The above equation has only one solution with $Z = 0$, this is $\mathcal{O} = (0 : 1 : 0)$ the point at infinity. Because of this, we can substitute $x = X/Z$ and $y = Y/Z$ to get an affine equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

with the extra point $\mathcal{O}$. This equation is called a long Weierstrass equation.

If E/K is an elliptic curve and $K \subseteq N$ is another field, we denote the set of points defined over N (called the N-rational points) with $E(N) = \{(x, y) \in E \mid x, y \in N\} \cup \{\mathcal{O}\}$.

If $\text{char}(K) \neq 2, 3$, then every elliptic curve admits a short Weierstrass equation

$$y^2 = x^3 + ax + b$$

Since in isogeny based cryptography we are working in finite fields with large characteristic, from now on we will use these short Weierstrass equations.

Definition 2.2. *The discriminant of the Weierstrass equation is* $\Delta = -16(4a^3 + 27b^2)$. *The* j*-invariant of the elliptic curve is* $j(E) = -1728\frac{(4a)^3}{\Delta}$

The j-invariant is called as such, because isomorphisms preserve it. Not only that, but two elliptic curves are isomorphic if and only if their j-invariant is the same.

The discriminant is useful for checking nonsingularity. A curve is singular $\iff \Delta = 0$.

The definition at the beginning of this section is not completely precise. In reality an elliptic curve is an (E, O) pair, where E a smooth projective curve of genus 1, and O a distinguished point on E. However, one can show that any such curve is isomorphic to the locus of a Weierstrass equation, which also sends O to $(0 : 1 : 0)$ [15, Chapter III.3].

2.1 Group law

Elliptic curves have additional structure: one can define a group action on it's points, requiring that every 3 points on a line sum to 0. Because of Bézout's theorem, every line will intersect the curve in exactly 3 points. The 0 of our group will be the point at infinity.

If we have a point $P = (x, y)$ on the curve, it's easy to see that the point $Q = (x, -y)$ will also be on the curve. Moreover, the third point on the curve collinear with P and Q is $\mathcal{O}$, which implies that $Q = -P$. Now we will define the $+$ operation properly:

Definition 2.3. *Let* $E : y^2 = x^3 + ax + b$ *be an elliptic curve and let* $P_1 = (x_1, y_1)$ *and* $P_2 = (x_2, y_2)$ *be two points on E,* $(P_1 \neq \mathcal{O} \neq P_2)$. *Then:*

- $P + \mathcal{O} = P = \mathcal{O} + P$ *for every* $P \in E$
- *If* $x_1 = x_2$ *and* $y_1 = -y_2$, *then* $P_1 + P_2 = \mathcal{O}$
- *Otherwise let*

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P_1 = P_2 \end{cases}$$

then $P_1 + P_2 = (x_3, y_3)$ *is defined by*

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = -\lambda x_3 - y_1 + \lambda x_1$$

[13]

Theorem 2.4. *An elliptic curve* E *with the* $+$ *operation defines an Abelian group*

Proof. One can verify that this addition rule geometrically means that we get $P+Q$ by taking the third intersection of the line of P and Q with the curve, then taking the intersection of the curve and the vertical line through this third intersection.

From this it's obvious that if P, Q, R are collinear points on the curve, then $(P+Q)+R = \mathcal{O}$.

From the geometric interpretation, it is also clear, that this operation is commutative, a line through P and Q is the same as the line through Q and P.

The existence of a 0 element, and the existence of $-P$ is evident from our definition. What is left to show is the associativity, whose proof is too long for this thesis. For the long proof see [20, Section 2.4], for an involved one see [15, Chapter III, Proposition 3.4e]. □

We will denote the m-th multiple of a point P with $[m]P$. The m-torsion subgroup of E is the set of points whose order divides m, which will be denoted by $E[m]$. For any field K over which E is defined the set of K-rational points, $E(K)$ is also a subgroup.

Proposition 2.5. *Let* E *be an elliptic curve, and let* $m \neq 0$ *be an integer.* $E[m]$ *has the following structure*

- $E[m] \simeq (\mathbb{Z}/m\mathbb{Z})^2$ *if the characteristic of* k *does not divide* m
- *If* $p > 0$ *is the characteristic of* k*, then*

$$E[p^i] \simeq \begin{cases} \mathbb{Z}/p^i\mathbb{Z} \text{ for any } i \geq 0, \text{ or} \\ \{\mathcal{O}\} \text{ for any } i \geq 0 \end{cases}$$

Proof. See [15, Coro. 6.4] □

2.2 Isogenies

This section will mostly follow Chapter III, Section 4 and 6 of [15], with some additions from previous chapters as some concepts defined in them will be needed here.

We want to look at maps between elliptic curves, which preserve both part of the curves: the projective variety, and the group.

Definition 2.6. *Let* E, E' *be two elliptic curves. An isogeny* $\phi : E \to E'$ *is a non-constant algebraic map of projective varieties sending the point at infinity of* E *onto the point at infinity of* E'*.*

Theorem 2.7. *Let* E, E' *be elliptic curves defined over a field* K *and let* $\phi : E \to E'$ *be an isogeny between them. Then:*

- ϕ *is a group morphism*
- ϕ *has finite kernel*
- *If* K *is algebraically closed,* ϕ *is surjective*

If we have two isogenies ϕ and ψ, we can define their sum:

$$\begin{aligned} (\phi + \psi) : E &\to E' \\ P &\mapsto \phi(P) + \psi(P) \end{aligned}$$

Therefore, if we add the constant map, which sends every point to $\mathcal{O}$, to the set of isogenies between E and E', we get a group, which we will denote by $\mathrm{Hom}(E, E')$. Moreover, because

the composition of isogenies is again an isogeny, the set of endomorphisms of the curve E (isogenies from the curve to itself) forms a ring, which we will denote by $\mathrm{End}(E)$.

From multiplication with integers, we can create isogenies (endomorphisms):

$$\begin{aligned} [m] : E &\to E \\ P &\mapsto [m]P \end{aligned}$$

We will now define the degree of an isogeny, but before that it is neccessary to introduce function fields.

Definition 2.8. *Let E/K be an elliptic curve. It's ideal, denoted by $I(E/K)$ is the set of polynomials in $K[x, y]$ which vanish on every point of the curve.*
The coordinate ring of E/K is $K[E] = \frac{K[x,y]}{I(E/K)}$.
The function field of E/K is the quotient field of $K[E]$, denoted by $K(E)$.

Remark 2.9. *Because the curve is a variety, $K[E]$ will always be an integral domain, hence it makes sense to talk about the quotient field of $K[E]$.*

Now if we have an isogeny $\phi : E_1 \to E_2$, it induces an injection in the other way between the function fields: $\phi^* : K(E_2) \to K(E_1)$, $\phi^*(f) = f \circ \phi$. Using this, we can now define the degree of an isogeny.

Definition 2.10. *Let $\phi : E \to E'$ be an isogeny. It's degree, $\deg \phi$, is equal to $[K(E) : \phi^*(K(E'))]$, the extension degree of the function fields. An isogeny is said to be separable, inseparable or purely inseparable if the extension of function fields is.*

Proposition 2.11. *For isogenies, this extension of function fields is always finite*

We have a nice formula for the degree of composed isogenies:

$$\deg(\phi \circ \psi) = \deg(\phi) \cdot \deg(\psi)$$

Most of the time, we will work with separable isogenies, whose degree equals the size of their kernel.

If $m \neq 0$, then the isogeny $[m]$ has the kernel $E[m]$. The degree of $[m]$ equals m^2 (if $p \nmid m$).

Theorem 2.12. *Let*

$$\phi : E_1 \to E_2 \qquad \text{and} \qquad \psi : E_1 \to E_3$$

be nonconstant isogenies, and assume that ϕ is separable. If $\ker \phi \subset \ker \psi$, then there is a unique isogeny

$$\lambda : E_2 \to E_3$$

such that $\psi = \lambda \circ \phi$

Notice how this theorem did not require from ψ to be separable.

There is one very important isogeny which we haven't yet talked about, which we define now.

Definition 2.13. *Let $E/K : y^2 = x^3 + ax + b$ be an elliptic curve defined over a field of characteristic $p > 0$. Let $q = p^n$. Then we call the map*

$$\phi_q((x, y)) = (x^q, y^q)$$

the q-th power Frobenius map, which is an isogeny between E and $E^{(q)} : y^2 = x^3 + a^q x + b^q$.

In particular, if $K = \mathbb{F}_{p^s}$, and $q = p^{rs}$, then this map is an endomorphism, and we call it the q-th power Frobenius endomorphism.

Propably the most important fact about the Frobenius map is the following.

Proposition 2.14. [15, Chapter II, 2.11] *Let ϕ_q be the q-th power Frobenius map as previously. The ϕ_q is a purely inseparable isogeny, and $\deg \phi_q = q$.*

Remark 2.15. *It is easy to see, that if $q = p^n$, then $\phi_q = \phi_p^n = \phi_p \circ \cdots \circ \phi_p$*

Proposition 2.16. [15, Chapter II, 2.12] *Let E_1, E_2 be elliptic curves defined over a field with finite characteristic. Let $\psi : E_1 \to E_2$ be an isogeny. Then it can be factored as $\lambda \circ \phi_q$, where $q = \deg_i(\psi)$, ϕ_q is the q-th power Frobenius, and λ is a separable isogeny.*

Previously we said that we will mostly work with separable isogenies. We can do that, because of this proposition, we can factor every inseparable isogeny into the composition of a separable isogeny and the Frobenius map. From this it also follows, that every purely inseparable isogeny is a q-th power Frobenius for some q.

Theorem 2.17. *Let E be an elliptic curve and let G be a finite subgroup of E. Then there exists a unique (up to isomorphism) elliptic curve E' and a unique (up to composition with isomorphism) separable isogeny ϕ, such that $\phi : E \to E'$ and $\ker \phi = G$.*

Definition 2.18. *An isogeny ϕ is cyclic, if $\ker \phi$ is a cyclic group.*

Another really important property of isogenies is the existence of the dual isogeny.

Theorem 2.19. *Let $\phi : E_1 \to E_2$ be an isogeny. There exists a unique isogeny $\hat{\phi} : E_2 \to E_1$ which together with ϕ satisfies $\hat{\phi} \circ \phi = [m]$ on E_1 and $\phi \circ \hat{\phi} = [m]$ on E_2.*

Theorem 2.20. [15, Chapter III, Theorem 6.2.] *Let $\phi, \psi : E_1 \to E_2$ be an isogeny, and let $\hat{\phi}$ be it's dual. Let $\lambda : E_2 \to E_3$ be another isogeny. The dual isogenies have the following properties:*

1. $\widehat{\lambda \circ \phi} = \hat{\phi} \circ \hat{\lambda}$
2. $\widehat{\phi + \psi} = \hat{\phi} + \hat{\psi}$
3. $\forall m \in \mathbb{Z}\ \widehat{[m]} = [m]$
4. $\deg \hat{\phi} = \deg \phi$
5. $\hat{\hat{\phi}} = \phi$

Proof. 1. Using the fact, that the composition of isogenies is associative and that $\forall m \in \mathbb{Z} :\ [m]$ commutes with any isogeny we can see that

$$(\hat{\phi} \circ \hat{\lambda}) \circ (\lambda \circ \phi) = \hat{\phi} \circ [\deg \lambda] \circ \phi = [\deg \lambda] \circ \hat{\phi} \circ \phi = [\deg \lambda \cdot \deg \phi] = [\deg(\lambda \circ \phi)]$$

From the above equation and the fact that the dual isogeny is unique, it follows that $\widehat{\lambda \circ \phi} = \hat{\phi} \circ \hat{\lambda}$

2. The proof of this involves divisors, which we won't introduce here, see [15, Chapter III, Theorem 6.2]

3. From the definition, it's clear that $\widehat{[m]} \circ [m] = [\deg([m])] = [m^2]$. Now it's clear that $[m] \circ [m] = [m^2]$, and from the uniqueness of the dual isogeny, it follows that $[m] = \widehat{[m]}$.

4. $\deg \hat{\phi} \cdot \deg \phi = \deg(\hat{\phi} \circ \phi) = \deg([\deg \phi]) = (\deg \phi)^2 \implies \deg \hat{\phi} = \deg \phi$

5. Using part 1 and 4 of the theorem, we can see that

$$\hat{\phi} \circ \phi = [\deg \phi] = \widehat{[\deg \phi]} = \widehat{\hat{\phi} \circ \phi} = \hat{\phi} \circ \hat{\hat{\phi}}$$

And again from the uniqueness we see that $\phi = \hat{\hat{\phi}}$

□

2.3 The endomorphism ring

Definition 2.21. *Let $d < 0$ be an integer. $\mathbb{Q}(\sqrt{d})$ is called an imaginary quadratic field.*

Definition 2.22. *A quaternion algebra is an algebra of the form*

$$\mathcal{K} = \mathbb{Q} + \mathbb{Q}\alpha + \mathbb{Q}\beta + \mathbb{Q}\alpha\beta$$

where α and β satisfies $0 > \alpha^2, \beta^2 \in \mathbb{Q}$ and $\alpha\beta = -\beta\alpha$

There is one specific quaternion algebra in which we are interested in: the quaternion algebra which ramifies at p (a specific prime) and at ∞. For every prime p there is up to isomorphism 1 unique quaternion algebra satisfying this. We will denote it by $B_{p,\infty}$.

Definition 2.23. *Let $\mathcal{K}$ be a $\mathbb{Q}$-algebra that is finitely generated over $\mathbb{Q}$. An order R of $\mathcal{K}$ is a subring that is also a finitely generated $\mathbb{Z}$-module, which satisfies $R \otimes \mathbb{Q}$.*

Theorem 2.24. *Let E be an elliptic curve defined over a field K of characteristic p. The ring End(E) is isomorphic to one of the following:*

- $\mathbb{Z}$;
- *An order $\mathcal{O}$ in a quadratic imaginary field. In this case we say E has complex multiplication by $\mathcal{O}$;*
- *Only if $p > 0$, a maximal order $\mathcal{O}$ in $B_{p,\infty}$. In this case we say that E has quaternionic multiplication by $\mathcal{O}$.*

2.4 Elliptic curves over Finite fields

While so far we have talked about elliptic curves over a general field, now we will focus our attention to finite fields. From now on p will always denote the (positive) characteristic of our field, and we will denote the finite field with $q = p^n$ elements by either $\mathbb{F}_q$ or $\mathbb{F}_{p^n}$. We will denote the closure of $\mathbb{F}_q$ by $\overline{\mathbb{F}}_q$.

One question which arises while working with elliptic curve over finite fields, is that if a curve E is defined over $\mathbb{F}_q$, how many $\mathbb{F}_q$-rational points are there? We will denote this number by $\#E(\mathbb{F}_q)$.

First we show that the number of rational points is "roughly" q.

Theorem 2.25. *(Hasse) Let $E/\mathbb{F}_q$ be an elliptic curve. Then*

$$|q + 1 - \#E(\mathbb{F}_q)| \leq 2\sqrt{q}$$

The first question that might arise in someone is: where does this number, $q+1-\#E(\mathbb{F}_q)$, come from? To answer this, we have to back away a bit.

Because in finite fields $x \in \mathbb{F}_q \iff x^q = x$, a point $P \in E$ is $\mathbb{F}_q$ rational if and only if $\pi_q(P) = P$. This means that $\ker(\pi_q - 1)$ is exactly the group of $\mathbb{F}_q$ rational points.

As it turns out the endomorphism $\pi_q - 1$ is a separable endomorphism, so $\#\ker(\pi_q - 1) = \deg(\pi_q - 1)$. To prove this, one has to introduce invariant differentials, which is out of scope for this work, but one can read about them in [15, III.5].

Now what is $\deg(\pi_q - 1)$? As we have seen previously,

$$[\deg(\pi_q - 1)] = (\pi_q - 1) \circ \widehat{(\pi_q - 1)}$$

and using the properties of the dual isogeny

$$[\deg(\pi_q - 1)] = (\pi_q - 1) \circ (\hat{\pi}_q - 1) = [q] + 1 - (\pi_q + \hat{\pi}_q)$$

Now rearranging the terms, we get that

$$\pi_q + \hat{\pi}_q = [q] + 1 - [\deg(\pi_q - 1)]$$

What we have shown that $\pi_q + \hat{\pi}_q = [t]$ for some $t \in \mathbb{Z}$. From this it also follows, that the Frobenius endomorphism satisfies the quadratic equation

$$x^2 - tx + q = 0$$

While we have not mentioned this yet, the degree map on the endomorphism ring is a positive definite quadratic form (see [15, III.6.3]

Proposition 2.26. [15, V.1.2] *Let A be an abelian group and let*

$$d : A \to \mathbb{Z}$$

be a positive definite quadratic form. Then

$$|d(\psi - \phi) - d(\phi) - d(\psi)| \leq 2\sqrt{d(\phi)d(\psi)}$$

for all $\phi, \psi \in A$.

Proof. See [15, V.1.2] □

Now we can prove Hasse's theorem

Proof. As we have seen above, we need to prove, that $|\deg(\pi_q - 1) - q - 1| \leq 2\sqrt{q}$. But because $q = \deg(\pi_q)$ and $1 = \deg(1)$ this is exactly the Cauchy-Schwarz inequality above. □

Now we will define supersingular elliptic curves. In isogeny based cryptography most of the schemes are based on them. They are useful for many reasons, we will list some of them. The first and most important reason is that given an equation for a supersingular elliptic curve, it is hard to calculate it's endomorphism ring, and this is equivalent with finding an isogeny between two supersingular curves [21], which is the hard problem on which most of the schemes are based on. The second reason is that a lot of schemes need curves with "smooth" cardinality, that is a cardinality whose prime divisors are "small" for some meaning of small. Supersingular curves' cardinality and group structure are known, and we can generate supersingular curves with smooth cardinality. The third reason is that every supersingular curves' j-invariant is defined over $\mathbb{F}_{p^2}$, which means that it is computationally much easier to manage them.

Definition 2.27. *Let $E/\mathbb{F}_q$ be an elliptic curve. The curve is supersingular, if $E[p^i] = \{\mathcal{O}\}$, for any (every) i.*

Curves not supersingular are called ordinary.

Now we will give other equivalent characterisations of supersingular curves.

Proposition 2.28. *An elliptic curve E defined over $\mathbb{F}_q$ is supersingular if and only if one of the following holds:*

- *The trace of it's Frobenius endomorphism is divisible by p*
- *$End(E)$ is an order in a quaternion algebra*

Proposition 2.29. *Let E be a supersingular elliptic curve. Then*

- *$j(E) \in \mathbb{F}_{p^2}$,*
- *There exists an isomorphism from E to a curve $E'/\mathbb{F}_{p^2}$ with the trace of the Frobenius endomorphism equal to $-2p$*

[13, Proposition 87]

Proposition 2.30. *Let E be a supersingular elliptic curve over $\mathbb{F}_{p^2}$. Then if the trace of the p^2-Frobenius is $-2p$ on E, then $\pi_{p^2}(P) = -p \cdot P$ for every P point on E.*

Proof. If $t = -2p$, then the minimal polynomial of the Frobenius is $x^2 + 2px + p^2 = (x+p)^2$, which means that $(\pi_{p^2} + p)^2(P) = \mathcal{O} \implies (\pi_{p^2} + p)(P) = \mathcal{O}$, that is $\pi_{p^2} = -p$. □

These two propositions mean, that when working with supersingular curves one can always assume that the p^2-Frobenius is just multiplication by $-p$.

3 Isogeny-based cryptography

3.1 Cryptographic group actions

Before we talk about concrete isogeny based cryptographic constructions, it is useful to talk about cryptographic group actions. It's an abstraction which can be used to decompose the

process of creating and understanding many isogeny based schemes into two steps: The first step is to show, that a group action with certain properties can be used to build a primitive, and the second is to show that a certain group action in isogenies satisfies those properties, hence can be used to create that primitive.

Definition 3.1. *A group G acts on the set X, if there is a map $\star : G \times X \to X$ which satisfies to following:*

- *If e is the identity element of G, then $e \star x = x \ \forall x \in X$*
- *$\forall g, h \in G$ and $\forall x \in X \ g \star (h \star x) = gh \star x$*

We denote the group action by $(G, X, \star)$

Definition 3.2. *A group action $(G, X, \star)$ is a cryptographic group action, if G and X is finite, G is commutative and there exists efficient algorithms for the following:*

- *Membership testing: Given a string g, decide whether g represents an element from G*
- *Equality testing: Given $g_1, g_2 \in G$ decide whether $g_1 = g_2$*
- *Group operations: Given $g_1, g_2 \in G$ calculate g_1^{-1}, $g_1 g_2$*
- *Sampling: Find a random element in G with uniform probability*
- *Membership testing: Given a string x, decide whether it represents an element from X*
- *Equality testing: Given $x_1, x_2 \in X$, decide whether $x_1 = x_2$*
- *Action: Given $g \in G$ and $x \in X$ compute $g \star x$*

and the following problem is hard:

Given $x_1, x_2 \in X$ find $g \in G$, such that $g \star x_1 = x_2$. (In groups this problem is called the Discrete Logarithm Problem (DLP))

Remark 3.3. *While we have not required to be able to sample from X uniformly, that's because we can just fix an element $x_0 \in X$, then get a random element by applying a random $g \in G$ to it.*

The Diffie-Hellman key exchange was defined for groups, but it is easy to see, that it works exactly the same way for group actions.

Say Alice and Bob want to exchange keys. They can follow the protocol below:

1. Alice chooses a random element $x_0 \in X$ and a random element $g_1 \in G$. She computes $x_1 = g_1 \star x_0$, and sends (x_0, x_1) to Bob.

2. Bob chooses a random element $g_2 \in G$ and computes $x_2 = g_2 \star x_0$. He sends x_2 to Alice. The secret key is $K = g_2 \star x_1$.

3. Alice computes the secret key $K = g_2 \star x_1 = g_1 \star h_2$.

[8, Section 3]

Remark 3.4. *Like the Diffie-Hellman based on groups, the version based on group actions does not rely on the DLP or it's group action version, but on a subtly different problem. Given x_0, x_1, x_2, where $x_1 = g \star x_0$, compute $g \star x_2$. This is the Computational Diffie Hellman problems (CDH) group action equivalent.*

Remark 3.5. *It is obvious that the CDH is easier than the DLP in both groups and group actions (that is if one solves DLP one also solves CDH). However, from a cryptographic perspective the other direction holds much interest, because the assumption is that DLP is hard (for groups in the classical setting, for group actions even with quantum computers).*

In the classical setting it is not known generally, whether solving the CDH for groups also solves the DLP, however it is known for group actions that the CDH and DLP are quantum equivalent [14].

3.2 Evaluating isogenies

So far we only talked about isogenies abstractly, but how can we store and evaluate them in practice?

If we have a cyclic isogeny from the curve E, it corresponds to a cyclic subgroup G on E. What we store is a generator P of G. We will shortly see Velu's formula which we can use to evaluate the isogeny using only P.

If our isogeny is non-cyclic, then we can write it as a composition of cyclic isogenies (and possibly a composition with the Frobenius endomorphism).

Proposition 3.6. *Vélu formulas* [19] *Let $E : y^2 = x^3 + ax + b$ be an elliptic curve defined over a field K and let $G \subset E(\overline{K})$ be a finite subgroup. Denote with $x(P), y(P)$ the x and y coordinates of a point P. Let $\phi : E \to E'$ be the unique (up to isomorphism) separable isogeny with kernel G. Then*

$$\phi(P) = \left(x(P) + \sum_{Q \in G \setminus \mathcal{O}} x(P+Q) - x(P), y(P) + \sum_{Q \in G \setminus \mathcal{O}} y(P+Q) - y(P) \right)$$

for every $P \in E(\overline{K}) \setminus G$. Moreover E' admits an equation $y^2 = x^3 + a'x + b'$, where

$$\begin{aligned} a' &= a - 5 \sum_{Q \in G \setminus \mathcal{O}} (3x(Q)^2 + a) \\ b' &= b - 7 \sum_{Q \in G \setminus \mathcal{O}} (5x(Q)^3 + 3ax(Q) + 2b) \end{aligned}$$

This gives us an algorithm to evaluate a separable isogeny in $O(d)$ field operations, where d is the degree of the isogeny. We can now see why we break down isogenies into the composition of cyclis ones. If the degree of a separable isogeny is $\prod_{i=1}^{s} p_i^{\alpha_i}$, then we can evaluate it in $O(\sum_{i=1}^{s} \alpha_i p_i)$ steps instead of $O(\prod_{i=1}^{s} p_i^{\alpha_i})$.

There exists a variant of the formula which is called $\sqrt{}$elu. It can evaluate an isogeny with prime degree l in $O(\sqrt{l})$ field operations [4]. This is not just a theoretical improvement which is useless in practice because of a large hidden constant in the O, it becomes more efficient than Vélu's formula for $l < 100$.

4 PEARL-SCALLOP

As discussed before cryptographic group actions are a very close analogue of the pre-quantum discrete logarithm problem and thus are an ideal setting to build certain advanced protocols. The first truly efficient instantiation is CSIDH [7] where the class group of $\mathbb{Z}[\sqrt{-p}]$ acts on the set of supersingular elliptic curves over $\mathbb{F}_p$. This action has several subtleties. First

elements of the class group are ideal classes but representatives of the ideal class matter from an algorithmic point of view. Namely, the norm of an ideal corresponds to the degree of the corresponding isogeny and since p is of cryptographic size, a random representative will not admit an efficient representation. The way this is handled in CSIDH is that one only works with smooth norm ideals and conjecturally these span the entire class group.

This approach is sufficient for a key exchange but not sufficient for signatures where one needs to sample random elements from an ideal class. The problem that arises is that p is quite big an thus computing the class group for appropriate sizes is infeasible. This issue was handled by CSi-FiSh [5] for the CSIDH-512 parameter set using a record class group computation but that approach does not even scale to CSIDH-1024. There are other methods to avoid this issue but knowing the order of the class group is actually highly important for advanced applications such as threshold schemes [10].

A completely new approach to resolve this issue was proposed in SCALLOP [9] where the acting group is still a class group but of a non-maximal order of large prime conductor of a quadratic field with class number 1. Such class numbers can be computed using the celebrated class number formula. However, computing the structure of the class group is a bit more involved and that poses some scaling problems as well. Furthermore, these specific issues pose both security and efficiency problems.

The aim of PEARL-SCALLOP is to design a variant of SCALLOP with the following properties:

- Maximal order that has a medium size class number
- A conductor which is the product of a few large primes
- Efficient group action

One difficulty of SCALLOP variants versus CSIDH is that one has to transfer orientations through the group action (as opposed to the canonical Frobenius) and PEARL-SCALLOP achieves the most practical instantiation in that regard as the orientation is represented by a 2^k-degree isogeny. One practical problem in PEARL-SCALLOP is the generation of an

oriented curve which is part of parameter generation. This is known to be polynomial time but naive methods can be too costly in practice. This where I contributed to the entire project.

4.1 My contributions

As a part of the precomputation we needed the basis for a lot of torsion subgroups on the curve $E : y^2 = x^3 + x$ (from now on, E will always denote this curve). More precisely:

Problem 1. *Given a prime p, and a lower bound T, select $m_1, m_2, \dots m_s$ prime powers, such that $p \mid m_i$, m_i is not "too large" and $lcm(m_1, m_2, \dots, m_s) \geq T$, and for each m_i find a basis (P_i, Q_i) of the torsion subgroup $E[m_i]$, where $E : y^2 = x^3 + x$ over $\overline{\mathbb{F}}_p$.*

These basis where later used to evaluate isogenies of order m_i, hence why they cannot be too large (in practice each $m_i < 100000$).
The main reason that this part of the precomputation was the bottleneck, that to find these torsion basis, it was necessary to do calculations over relatively large extensions of $\mathbb{F}_{p^2}$.

Now let us see how one can find a torsion bases for a given m_i.
First, we have to find the smallest extension $\mathbb{F}_{p^{2k}}$, for which $E[m_i] \subseteq E(\mathbb{F}_{p^{2k}})$.

It is known [12, Theorem 2], that the group structure of

$$E(\mathbb{F}_{p^{2k}}) \simeq (\mathbb{Z}/(p^k - (-1)^k)\mathbb{Z})^2$$

This means that we only have to find the smallest k, for which $m_i \mid p^k - (-1)^k$, and simply incrementing k until this happens is fast enough for us.

I will now show the high level algorithm to find the torsion basis.

Algorithm 1 Finding the m-torsion basis in $E(\mathbb{F}_{p^{2k}})$

function TorsionBasis(m, p, k)

while true **do**

$P \leftarrow$ TorsionPoint(m, p, k)

$Q \leftarrow$ TorsionPoint(m, p, k)

if P, Q generate the torsion **then**

return (P, Q)

Algorithm 2 Finding a point of order m in $E(\mathbb{F}_{p^{2k}})$

function TorsionPoint(m, p, k)

while true **do**

$R \leftarrow$ RandomPoint(p, k)

$c \leftarrow \frac{p^k - (-1)^k}{m}$

$P \leftarrow c \cdot R$

if order of P is m **then**

return P

Algorithm 3 Generating a random point in $E(\mathbb{F}_{p^{2k}})$

function RandomPoint(p, k)

while true **do**

$x \leftarrow$ Random$(\mathbb{F}_{p^{2k}})$

$z \leftarrow x^3 + x$

if $\sqrt{z} \in \mathbb{F}_{p^{2k}}$ **then**

return $(x, \sqrt{z})$

I will now describe those optimizations I implemented, which are also interesting from a mathematical perspective.

After some testing, it turned out that by far the largest bottleneck in the torsion basis finding is the single line $P \leftarrow c \cdot R$ in the torsion point finding. This single operation took the majority of the time.

After looking closer, this makes sense: as m is really small (< 100000) and p is large $\log(p) \sim 2000$, hence $c \sim p^k$. Then, multiplication by an integer c takes $\log(c) \sim k \cdot \log(p)$ additions and doublings on the curve over $E(\mathbb{F}_{p^{2k}})$. Even if we assume that a multiplication/division in $\mathbb{F}_{p^{2k}}$ takes $2k\log(k)$ operations in $\mathbb{F}_p$, overall we need at least $2k^2\log(k)\log(p)$ operations in $\mathbb{F}_p$.

There are two main ways to reduce the time taken up by these multiplications. The first is to make the multiplications faster and the second is to make fewer multiplications. I'll start by explaining the latter.

4.1.1 Fewer multiplications

A naive idea would be to instead of finding torsion bases for $m_1, m_2, \ldots m_s$, what if we tried to find a torsion bases for $\prod_{i=1}^{s} m_i$? Then from that it would be straightforward to calculate the m_i-torsion bases.

The main problem with this idea is that while one by one each of the m_i-torsion may be defined over relatively small extensions of $\mathbb{F}_{p^2}$, the $\prod_{i=1}^{s} m_i$-torsion is only defined over a huge extension, where it would be impossible to make computations.

However what we can do is to group together those m_i-torsions, who are defined over the same extension. As there is not many small prime powers whose torsion is defined over a given extension, calculating them together takes roughly the same time as it would have a single one.

By this optimalization it was possible to achieve a 2-5x speedup in each extension.

Another optimalization independent from the previous is the following:

We know the endomorphism ring of E, concretely we know the following two endomorphisms:

$$\pi_p : (x, y) \mapsto (x^p, y^p)$$

and

$$i : (x, y) \mapsto (-x, \sqrt{-1}y)$$

Find a point P of order m. Then instead of finding another point the usual way check whether $\pi_p(P)$ or $i(P)$ forms a basis together with P. While I was not able to determine the precise probability of this method working, in practice this did indeed work in most of the cases.

This provided another roughly 2x speedup.

4.1.2 More efficient multiplication

The main fact that will be used to speed up the multiplication is that the p^2-Frobenius on E is equivalent to multiplication by $-p$

$$\forall P \in E : \pi_{p^2}(P) = -p \cdot P$$

This follows from Proposition 2.28, and the fact that $Tr(\pi_{p^2}) = p^2 + 1 - \#E(\mathbb{F}_{p^2}) = p^2 + 1 - (p+1)^2 = -2p$.

Thus we can compute $p \cdot P$ much faster that the simple double and add algorithm, by calculating $-\pi_{p^2}(P) = (x^{p^2}, -y^{p^2})$, where $P = (x, y)$. However we can evaluate the Frobenius endomorphism even faster, but for that we need to talk about how $\mathbb{F}_{p^{2k}}$ is represented in software. This is primarily done by taking an irreducible, degree $2k$ polynomial $f \in \mathbb{F}_p[x]$, then representing elements of $\mathbb{F}_{p^{2k}}$ by polynomials of degree less than $2k$. The computations are done mod (p), mod (f).

Now, instead of raising $z \in \mathbb{F}_{p^{2k}}$ to the pth power we can calculate the polynomial $\mathbb{F}_p[x] \ni g \equiv x^p \mod (f)$ with just as many operations. Then whenever we need to calculate z^p for a $z \in \mathbb{F}_{p^{2k}}$, which is actually just the polynomial, we can calculate $g(z) \equiv z^p \mod (f)$. This can be done in $O((2k)^{(1+\omega)/2})$, where ω is the coefficient for the complexity of matrix

multiplication, meaning that we can multiply two $n \times n$ matrices in $O(n^\omega)$ operations [6]. While there are known algorithms with $\omega \leq 2.5$, we can just work with the naive $\omega = 3$, which makes the modular polynomial composition work in $O(k^2)$.

Moreover if we want to raise multiple field elements $z_1, z_2, \dots z_t$ to the power of p^r, we can first compute $g_r = g(g(g(\dots)))$, that is g composed r times with itself. Computing g_r can be done in $O(\log(r))$ compositions. Then after calculating g_r, we can compose it with our elements $g_r(z_1), g_r(z_2), \dots g_r(z_t)$.

Assume that we would calculate z^{p^2} for t different field elements during our precomputation. Then using the naive approach would take $O(\log(p^2) \cdot t)$ modular polynomial multiplications which is $O(\log(p) \cdot M(k) \cdot t)$ operations in $\mathbb{F}_p$, where $M(k)$ is the number of operations needed to calculate $f \cdot g \mod (h)$ for $f, g, h \in \mathbb{F}_p[x]$, with their degrees at most k. For the range of k in our case this goes from the naive k^2 to the FFT multiplication's $k \log(k)$.

Using modular composition instead, we need $O(\log(p))$ polynomial multiplications and $t+1$ modular compositions, resulting in an overall $O(\log(p) \cdot M(k) + t \cdot k^2)$ operations in $\mathbb{F}_p$, which is much better than the naive approach.

Now that the amortized cost of evaluating $-\pi_{p^2}(P)$ is near zero compared to computing $p \cdot P$ naively, how can we utilize this?
We could try to write $c \cdot P$ as

$$\left(\sum_{i=0}^{k-1} c_i p^i \right) \cdot P = \sum_{i=0}^{k-1} c_i (p^i \cdot P)$$

compute $p^i \cdot P$ efficiently, then calculating $c_i \cdot p^i P$ using double and add (here $\sum_{i=0}^{k-1} c_i p^i$ is c written into base p). The problem with this approach is that in general c_i is a "random" number between 0 and p, meaning that almost always $\log(c_i) \sim \log(p)$, hence our overall complexity is still $k \cdot \log(p)$.
However there are two ways to make this idea work.

The first idea is to notice that $c = \frac{p^k - (-1)^s}{m}$ which is not a random number. Let us denote

by Φ_d the dth cyclotomic polynomial. If $k = 2l$ is even, then

$$p^k - (-1)^k = p^k - 1 = \prod_{d|k} \Phi_d(p)$$

and if $k = 2l + 1$ is odd, then

$$p^k - (-1)^k = p^k + 1 = \prod_{d|k} \Phi_{2d}(p)$$

as $(p^k + 1) \cdot (p^k - 1) = p^{2k} - 1$.

Let us assume for a moment that m is prime. Then it must divide $\Phi_d(p)$ if k is even and Φ_{2d} if k is odd for some $d \mid k$. But we also know which d this must be, because we know that k is the smallest positive integer for which $m \mid p^k - (-1)^k$.

If k is odd, then $\Phi_{2d}(p) \mid p^d + 1 = p^d - (-1)^d$, hence the only factor which does not occur for smaller extension degrees is $\Phi_{2k}(p)$.

If $k = 2l$ with l odd, then $\forall d \mid l$, $\Phi_{2d}(p) \mid p^l - (-1)^l$ and $\Phi_d(p) \mid p^{2d} - 1 = p^{2d} - (-1)^{2d}$. Hence the only factor, whose first occurrence is for k is $\Phi_l(p)$.

If $4 \mid k$, then with a similar logic the only new factor will be $\Phi_k(p)$.

This means that for every k we can select an l_k, such that $m \mid \Phi_{l_k}(p) \mid p^k - (-1)^k$. This means that we can write $c \cdot P$ as

$$\frac{p^k - (-1)^k}{m} \cdot P = \frac{\Phi_{l_k}(p)}{m} \cdot \frac{p^k - (-1)^k}{\Phi_{l_k}(p)} \cdot P$$

with both terms being integers. This is great for us, because the second term $\frac{p^k-(-1)^k}{\Phi_{l_k}(p)}$ has small coefficients in base p (because it is the product of not too many cyclotomic polynomials evaluated at p), which means that we can use the above described algorithm using the Frobenius endomorphism to quickly compute $\frac{p^k-(-1)^k}{\Phi_{l_k}(p)} \cdot P$. Then instead of needing $k \cdot \log(p)$ doublings and additions, we only need $\deg(\Phi_{l_k}) \cdot \log(p)$ doublings and additions.

At the start of this technique we restricted ourselves to m being a prime. What we do in practice instead is to search from the start not divisors of $p^k - (-1)^k$ but divisors of $\Phi_{l_k}(p)$. This does mean that we sometimes don't pick prime powers which we otherwise could have, but overall this is a great tradeoff.

The other technique we can use is multi-exponentiation (which in our case becomes "multi-multiplication" as our group is denoted additively). Let

$$\frac{\Phi_{l_k}(p)}{m} = \sum_{i=0}^{r} e_i p^i$$

and as before, we can compute $p^i \cdot P$ efficiently. Then we would need to compute $\sum_{i=0}^{r} e_i(p^i \cdot P)$. This can be done efficiently using the 2^ω-ary method for $\omega = 1$ also called Shamir's trick [3]:

Algorithm 4 Shamir's trick for multiexponentiation

function MULTIEXP($P_1, \ldots, P_r, e_1, \ldots, e_r$)

  for $(x_1, \ldots x_r) \in \{0,1\}^r$ **do**

    $Q[x_1, \ldots, x_r] \leftarrow \sum_{i=1}^{r} x_i \cdot P_i$

  $l \leftarrow \max\{\lceil \log(e_i) \rceil \mid i = 1, 2, \ldots, r\}$

  $R \leftarrow \mathcal{O}$

  for $i = l, l-1, \ldots 0$ **do**

    $R \leftarrow 2 \cdot R$

    $R \leftarrow R + Q[e_1^i, e_2^i, \ldots, e_r^i]$ (where e_j^i is the ith bit of e_j)

  return R

This algorithm needs 2^r additions on the curve in the precomputation phase and l doublings and additions in the main phase. In our case $l \sim \log(p)$ and $r = \deg(\Phi_{l_k})$. However this algorithm also needs to store 2^r points, which did actually put an upper bound on r in our case, because if $k \geq 100$, then storing a single point means storing 2 $2k$ degree polynomials in $\mathbb{F}_p[x]$, which means storing $2k$ $\log(p)$ sized numbers requiring more than $4 \cdot 100 \cdot 2000 = 800000$ bits = 100 Kilobytes of storage.

Because of this, we limited the multiexponentiation to only do 10 points at once. That is if $\deg(\Phi_{l_k}) > 10$, we split up $P, p \cdot P, \ldots p^{\deg(\Phi_{l_k})} \cdot P$ into chunks of 10, calculated $\sum_{i=10t}^{10t+9} e_i(p^i \cdot P)$ for each chunk, then summed them.

Overall (for large k) instead performing $\deg(\Phi_{l_k}) \cdot \log(p) \sim \deg(\Phi_{l_k}) \cdot 2000$ doublings and additions, we performed $\frac{\deg(\Phi_{l_k})}{10} \cdot (1024 + \log(p)) \sim \deg(\Phi_{l_k}) \cdot 300$ doublings and additions

which is 6-7x factor.

4.1.3 Faster square root

After all these improvements to multiplication by an integer, it got fast enough for other bottlenecks to appear: namely taking square roots in finite fields.

We needed square roots in two places:

- Embedding $\mathbb{F}_{p^2} = \mathbb{F}_p[x]/(x^2+1)$ into $\mathbb{F}_{p^{2k}}$. For this we needed to find $\sqrt{p-1}$ in $\mathbb{F}_{p^{2k}}$.
- Finding random points on the elliptic curve

For square root finding in finite fields, PARI/GP used the Tonelli-Shanks algorithm, which works for any group. However it's complexity depends on $s = val_2(p^{2k}-1)$, the 2-adic valuation of the cardinality of $\mathbb{F}_{p^{2k}}^{\times}$, which in our case was quite large. There are algorithms specifically for square root finding in finite fields like [1] and [11]. The latter one is actually superior over both the former and the Tonelli-Shanks algorithm. I implemented this in C using the PARI/GP library. Later I also contributed this and it's extension for tth root finding to PARI.

The overall time reduction in the precomputation meant that the $\log(p) \sim 2000$ case had roughly the same runtime as the $\log(p) \sim 1000$ had before my optimizations.

The code I contributed to the project can be found at `https://github.com/mtotbagi/SCALLOP-params/blob/main/starting-curve/` in the precompute_parallel file and the c_torsion folder.

As a result of these contributions I became a co-author of the paper Faster SCALLOP from Non-Prime Conductor Suborders in Medium Sized Quadratic Fields which got published in the IACR International Conference on Public-Key Cryptography [2].

5 Other algorithmic improvements

5.1 Finding the order of a group element

Let G be a finite group of order n. Assume that we know the prime factorization of $n == \prod_{i=1}^{s} p_i^{\alpha_i}$ with p_i being distinct primes, e.g. because n is powersmooth.

It is important to be able to find the order of an element of the group efficiently. We first define two problems:

Problem 2. *Let g be an element in the group G. Find the $o(g)$, the order of g in polynomial of $\log(n)$ number of operations (assuming we know the factorization of n).*

In the problem above, we don't have anything information about the element g, which we sometimes do have, hence the following problem:

Problem 3. *Let g be an element in the group G, and let $k \in \mathbb{Z}^{>0}$ such that $g^k = 1$. Assume that we know the factorization of k. Find the $o(g)$, the order of g in polynomial of $\log(k)$ number of operations.*

One can see that this is a generalization of the first problem, we always know that $g^n = 1$, so it is enough to provide an algorithm for this.

Let's give an algorithm for a subset of this problem, when k is a prime power, $k = p^\alpha$. This will be used in the general algorithm.

Well, as k is a prime power, there aren't many options for k, namely $1, p, p^2, \ldots, p^\alpha$.

It's easy to see that the worst case is when $o(g) = k$, and that in this case the number of group operations is $O(\alpha \cdot \log(p)) = O(\log(k))$.

Now we will see two algorithms to solve the problem generally, both based on a similar principle.

Let $k = \prod_{i=1}^{s} p_i^{\alpha_i}$ and $g^k = 1$. Let $c_i = \frac{k}{p_i^{\alpha_i}}$. Using the previous algorithm we can calculate $o(g^{c_i}) = p_i^{\beta_i}$, with $0 \leq \beta_i \leq \alpha_i$. Then we know that $g^{c_i p_i^{\beta_i}} = (g^{c_i})^{p_i^{\beta_i}} = 1$, so

$$o(g) \mid c_i p_i^{\beta_i} \implies o(g) \mid \gcd\{c_i p_i^{\beta_i} \mid i = 1, 2, \ldots s\} = \prod_{i=1}^{s} p_i^{\beta_i}$$

Algorithm 5 Finding the order of an element in the easy case

function EASYORDER(g, p, α)

$c \leftarrow 0$

$h \leftarrow g$

while $c < \alpha$ **do**

if $h = 1$ **then**

break

$h \leftarrow h^p$

$c \leftarrow c + 1$

return p^c

It is also easy to see, that $g^{\prod_{i=1}^{s} p_i^{\beta_i}} = 1$, hence $o(g) = \prod_{i=1}^{s} p_i^{\beta_i}$.

From this we can construct the algorithm which was also used in the PARI/GP computer algebra software [16].

Algorithm 6 General algorithm for finding the order of an element

function GENORDER($g, k = \prod_{i=1}^{s} p_i^{\alpha_i}$)

$o \leftarrow k$

$h \leftarrow g$

for $i \leftarrow 1..s$ **do**

$c \leftarrow o/p_i^{\alpha_i}$

$h \leftarrow g^c$

$t \leftarrow$ EASYORDER(h, p_i, α_i)

$o \leftarrow c \cdot t$

return o

How many group operations does this algorithm use? Well, as in the easier case, the worst case is when $o(g) = k$. Then in the ith iteration, the algorithm performs $O(\alpha_i \log(p_i) + \log(k/p_i^{\alpha_i})) = O(\log(k))$. As we have s number of iterations (which is the number of distinct prime divisors of k), the overall complexity is $O(s \log(k))$.

We can also notice, that we only need to store $O(1)$ number of group elements at once.

However we can achieve better time performance than $O(s\log(k))$. I will now present my alternative algorithm for computing the order of an element. The main idea is, that instead of iterating through each possible prime divisor of the order of g, we can instead split them into two, then recursively calculate the order for the first and second halves.

Algorithm 7 Recursive algorithm for finding the order of an element

function RecOrder($g, k = \prod_{i=1}^{s} p_i^{\alpha_i}$)

  if $s = 1$ **then**

    return EasyOrder(g, p_1, α_1)

  $c \leftarrow \prod_{i=\lfloor s/2 \rfloor+1}^{s} p_i^{\alpha_i}$

  $h \leftarrow g^c$

  $o_1 \leftarrow$ RecOrder($h, k/c = \prod_{i=1}^{\lfloor s/2 \rfloor} p_i^{\alpha_i}$)

  $h \leftarrow g^{o_1}$

  $o_2 \leftarrow$ RecOrder($h, o_1 = \prod_{i=\lfloor s/2 \rfloor+1}^{s} p_i^{\beta_i}$)

  return $o_1 \cdot o_2$

We can see that the depth of the recursivity will be $\lceil \log(s) \rceil$ and the number of distinct prime divisors at the ith depth level will be $\lceil \frac{s}{2^i} \rceil$. Using this, we can calculate the number of group operations used in the algorithm (in the worst case).

At the last depth level, when $s = 1$, as we have already seen there will be $O(\log(p_i^{\alpha_i}))$ operations in one function call, overall $O(\log(k))$. At other depth levels we cannot give a nice formula for the number of operations in one function call. However, we can see that overall in a fixed depth level there will be overall $O(\log(k))$ + the recursive calls.

This means that overall there will be $O(\log(s)\log(k))$ group operations. This gives us an improvement from the previous algorithm, which needed $O(s\log(k))$ operations. In some use cases this is a really significant improvement e.g when $s = 50$, this results in a 12x faster algorithm.

There is one small downside of my algorithm, which is that it does need to store $O(\log(s))$ group elements. However in practice this should not mean a problem, even if $k \sim 2^{100000}$ $s < 10000$ meaning $\log(s) \leq 14$, which can be handled easily.

5.2 Division fields

Definition 5.1. *The m-division field of an elliptic curve $E : y^2 = x^3 + ax + b$ over a finite field $\mathbb{F}_q$ is the smallest extension of $\mathbb{F}_q$ over which the m-torsion is rational.*

Problem 4. *Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$. Find the extension degree of the m-division field.*

In Section 4.1, we solved this problem with a naive approach, which also utilized that we know the group structure of supersingular curves. In this section we will introduce a better one, which also works for ordinary curves.

One algorithm is to take the mth division polynomial φ_m, whose roots are the x-coordinates of the m-torsion points, and find the splitting field of it over $\mathbb{F}_q$, lets say that it is $\mathbb{F}_{q^k}$. Then every x-coordinate of m-torsion points is defined over $\mathbb{F}_{q^k}$, and every y-coordinate can be written as $\sqrt{x^3 + ax + b}$, which is either in $\mathbb{F}_{q^k}$ or in $\mathbb{F}_{q^{2k}}$. It can also be proven [18, Lemma 4.4], that it is enough to check whether a single y-coordinate (of a point of full order m) is in $\mathbb{F}_{q^k}$ and if it is, then the whole torsion is defined over $\mathbb{F}_{q^k}$.

This is the algorithm implemented in SageMath [17]. However creating the splitting field and checking whether an y-coordinate is defined over it is quite slow even for $n > 100$.

5.2.1 A faster algorithm

We will now explain algorithm 1 from [18], which works for primes. Then we will show our algorithm which works for any odd prime power, consequently also working for any odd number.

First, the case $m = 2$ is a special case. If $P = (x_0, y_0)$ and $2P = \mathcal{O}$, then $P = -P \implies y_0 = 0 \implies x_0$ is a root of the polynomial $x^3 + ax + b$. Thus the 2-division field is the splitting field of $x^3 + ax + b$.

From now on let m denote an odd prime ($m \neq p$). As before, we denote the Frobenius endomorphism by π_q and we will denote $\pi_q|_{E[m]} = \varphi_q$. Now consider that $\pi_q^n = \pi_{q^n}$ and that if $\pi_{q^n}(P) = P$ then $P \in E(\mathbb{F}_{q^n})$. These two facts imply that the order of $\varphi_q =$ the degree

of the m-division field. Here the order of φ_q (from now on denoted by $o(\varphi_q)$) is meant as a group homomorphism acting on $E[m]$. This means that it is enough to find $o(\varphi_q)$

By choosing a base of $E[m]$ P, Q, we can map φ_q to a matrix $M \in GL_2(\mathbb{F}_m)$. Their minimal polynomial of the Frobenius endomorphism modulo (m) (which is $f(x) = x^2 - tx + q$, where $t = q + 1 - \#E(\mathbb{F}_q)$) will be equal to the minimal polynomial of M, and $o(\varphi_q) = o(M)$.

Over $\overline{\mathbb{F}}_m$ there is an M' matrix in Jordan-normal form which is similar to M, $M \sim M'$. We also know that $o(M) = o(M')$, and their minimal polynomial is also equal. The diagonal elements of M' are the roots of the minimal polynomial. Using this, we can (in most cases) determine $o(M') = o(M) = o(\varphi_q)$.

We can create 3 cases based on the roots of the minimal polynomial:

- If the roots of f are $\alpha \neq \beta \in \mathbb{F}_m$, then M' is simply

$$\begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}$$

  and $o(M') = \mathrm{lcm}(o(\alpha), o(\beta))$.

- If the roots are not in $\mathbb{F}_m$, then they are α and $\alpha^m \in \mathbb{F}_{m^2}$, and $o(M') = \mathrm{lcm}(o(\alpha), o(\alpha^n)) = o(\alpha)$.

- The last case is when there is a single root $\alpha \in \mathbb{F}_m$ with multiplicity 2. This is the hardest case, as M' can now have two forms:

$$\begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix} \qquad \begin{pmatrix} \alpha & 1 \\ 0 & \alpha \end{pmatrix}$$

  In the first case $o(M') = o(\alpha)$ and in the second case $o(M') = m \cdot o(\alpha)$. This follows from the fact that m is the smallest positive integer for which M'^m is a diagonal matrix and $o(\alpha) = o(\alpha^m)$.

  How can we decide between the two cases?

  We can use the division polynomial. Let $d = o(\alpha)$. If the mth division polynomial splits into linear factors over $\mathbb{F}_{q^d}$, then we know that the m-division field is either $\mathbb{F}_{q^d}$

or $\mathbb{F}_{q^{2d}}$. But we also know that the m-division field must be either $\mathbb{F}_{q^d}$ or $\mathbb{F}_{q^{md}}$ and we also know that $m \neq 2$, hence the m-division field must be $\mathbb{F}_{q^d}$.

If the mth division polynomial does not split over $\mathbb{F}_{q^d}$ the the m-division field must be $\mathbb{F}_{q^{md}}$.

Notice that algorithm 1 in [18] starts with calculating $\#E(\mathbb{F}_q)$. But this is unnecessary, we only need to determine the trace of the Frobenius $\mod(m)$, which is one of the steps in Schoof's algorithm to determine $\#E(\mathbb{F}_q)$. So instead we can just use that substep from Schoof's algorithm which will be much faster then computing the whole cardinality.

5.2.2 Prime powers

Now let $1 < k \in \mathbb{Z}$ and we want to find the m^k-division fields extension degree, assuming that the m^{k-1} is rational over $\mathbb{F}_q$ (if it isn't, we can use this algorithm recursively). Our main tool again will be the Frobenius endomorphism restricted to $E[m^k]$.

We want to find $o(\pi_q|_{E[m^k]})$. Let P, Q be a basis for $E[m^k]$. Then $o(\pi_q|_{E[m^k]})$ is the smallest j for which $\pi_q^j(P) = \pi_q^j(Q) = \mathcal{O}$. Now using the fact that the m^{k-1} torsion is rational:

$$\begin{aligned} m \cdot \pi_q(P) = \pi_q(m \cdot P) &= m \cdot P \\ m \cdot (\pi_q(P) - P) &= \mathcal{O} \end{aligned}$$

Thus we can write $\pi_q(P) = P + P'$, where P' is an m-torsion point. From this

$$\pi_q^s(P) = \pi_q^{s-1}(P + P') = \cdots = P + s \cdot P'$$

What we got is that $o(\pi_q|_{E[m^k]})$ is either 1 or m. Notice that so far this is all theory, no calculations were needed for this. To actually decide whether the m-division field degree is 1 or m we can again use the division polynomial in the same way we used it when the minimal polynomial had a single root.

5.2.3 Composite numbers

Now to find the n-division field degree for an odd composite number n ($p \nmid n$), we first need to factor it. We can easily do this, as next to the following steps the cost of this is insignificant.

Let $n = \prod_{i=1}^{s} p_i^{\alpha_i}$. Calculate the $p_i^{\alpha_i}$ for each i, let the result of it be d_i. Then the n-division field degree is $\operatorname{lcm}\{d_i \mid i = 1, 2, \ldots s\}$.

References

[1] Gora Adj and Francisco Rodríguez-Henríquez, *Square root computation over even extension fields*, IEEE Transactions on Computers **63** (2013), no. 11, 2829–2841.

[2] Bill Allombert, Jean-François Biasse, Jonathan Komada Eriksen, Péter Kutas, Chris Leonardi, Aurel Page, Renate Scheidler, and Márton Tot Bagi, *Faster scallop from non-prime conductor suborders in medium sized quadratic fields*, Iacr international conference on public-key cryptography, 2025, pp. 333–363.

[3] Vidal Attias, Luigi Vigneri, and Vassil Dimitrov, *Rethinking modular multi-exponentiation in real-world applications*, Journal of Cryptographic Engineering **13** (2023), no. 1, 57–70.

[4] Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith, *Faster computation of isogenies of large prime degree*, Open Book Series **4** (2020), no. 1, 39–55.

[5] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren, *Csi-fish: efficient isogeny based signatures through class group computations*, International conference on the theory and application of cryptology and information security, 2019, pp. 227–247.

[6] Richard P Brent and Hsiang T Kung, *Fast algorithms for manipulating formal power series*, Journal of the ACM (JACM) **25** (1978), no. 4, 581–595.

[7] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes, *Csidh: an efficient post-quantum commutative group action*, Advances in cryptology–asiacrypt 2018: 24th international conference on the theory and application of cryptology and information security, brisbane, qld, australia, december 2–6, 2018, proceedings, part iii 24, 2018, pp. 395–427.

[8] Jean-Marc Couveignes, *Hard homogeneous spaces*, 2006. `https://eprint.iacr.org/2006/291`.

[9] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski, *SCALLOP: scaling the CSI-FiSh*, PKC (2023).

[10] Luca De Feo and Michael Meyer, *Threshold schemes from isogeny assumptions*, Iacr international conference on public-key cryptography, 2020, pp. 187–212.

[11] Javad Doliskani and Éric Schost, *Taking roots over high extensions of finite fields*, Mathematics of Computation **83** (2014), no. 285, 435–446.

[12] Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni, *Deuring for the people: Supersingular elliptic curves with prescribed endomorphism ring in general characteristic.*, IACR Cryptol. ePrint Arch. **2023** (2023), 106.

[13] Luca De Feo, *Mathematics of isogeny based cryptography*, ArXiv **abs/1711.04062** (2017).

[14] Steven Galbraith, Yi-Fu Lai, and Hart Montgomery, *A simpler and more efficient reduction of dlog to cdh for abelian group actions*, Iacr international conference on public-key cryptography, 2024, pp. 36–60.

[15] Joseph H Silverman, *The arithmetic of elliptic curves*, Vol. 106, Springer, 2009.

[16] *PARI/GP version 2.15.4*, The PARI Group, Univ. Bordeaux, 2023. available from `http://pari.math.u-bordeaux.fr/`.

[17] The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 10.3)*, 2024. `https://www.sagemath.org`.

[18] A van Tuyl, *The field of n-torsion points of an elliptic curve over a finite field*, Ph.D. Thesis, 1997.

[19] Jacques Vélu, *Isogénies entre courbes elliptiques*, CR Acad. Sci. Paris, Séries A **273** (1971), 305–347.

[20] Lawrence C Washington, *Elliptic curves: number theory and cryptography*, Chapman and Hall/CRC, 2008.

[21] Benjamin Wesolowski, *The supersingular isogeny path and endomorphism ring problems are equivalent*, 2021 ieee 62nd annual symposium on foundations of computer science (focs), 2022, pp. 1100–1111.

I, Tot Bagi Márton, declare I have not used AI based tools while working on my thesis.