Contents

Chapter 1

Introduction

The word "*cryptography*" derives from the Greek κρυπτὸς, *kryptos*, meaning hidden. Its origin is usually dated in 2000 B.C., with the Egyptian practice of hieroglyphics which full meaning was only known to an elite few.
Formally, cryptography is the science of protecting information between two or more parties. It has advanced exponentially by taking advantage of the most developed mathematical theories and it has became more secure by relying on hard mathematical problems. Its algorithms are designed around computational complexity assumptions: it is theoretically possible to break a cryptosystem but it is infeasible in practice due to excessively long running times. These schemes are therefore said computationally secure.

Introduced in the early 19-th century, Modern Cryptography is a wider field than pure cryptography since it embraces additional properties vital to modern communications that can be summed up in:

- Authenticity - ensures that the information comes from the source it claims to be from;
- Confidentiality - ensures that information is not made available or disclosed to unauthorised individuals, entities, or processes;
- Integrity - ensures the information sent is exactly the same information received;

- Non Repudiation - provides proof of the integrity and origin of transmitted information.

In modern cryptography, Symmetric Key and Public Key encryption schemes are the best known and world-wide used. The symmetric key algorithms make use of a single key to both encrypt and decrypt messages. This key is shared by the communication parties hence the name symmetric. An example is the Advanced Encryption Standard (AES), born in 1997 to supersede its predecessor DES in terms of both efficiency and security offered. More on this protocol can be found in [18].
Public key algorithms use a pair of keys: the public key serves to encrypt messages, the private one to decrypt it. These special algorithms are also known as Asymmetric since encryption and decryption keys are different and not shared by the parties. These schemes are commonly implemented in a 2-parties version in which the users Alice and Bob want to communicate without revealing their messages' content to others. Many protocols have been proposed to this purpose such that today we have RSA [19] as the most world spread algorithm and Elliptic Curve Cryptography (ECC) [20] as most recent and efficient scheme implemented.

In 1982 Richard Feynman observed that simulating quantum physics on classical computers seemed an intractable task [2]. He then followed up by conjecturing that physical devices relying on quantum phenomena would have been good candidates for simulating quantum mechanics. Soon after, David Deutsch in [21] formalised the notion of a quantum Turing machine showing that it was universal: a quantum Turing machine can simulate any quantum mechanical process with small overhead and independently of the substrate.
The basic idea behind a quantum computer is to exploit quantum bits, or qubits for short. As opposed to binary bits, qubits can exist in additional states in between the two binary states. This is defined as a superposition of the digital states. A full disclosure about quantum computers is beyond the scopes of this thesis but a greed reader can refer to [1,2,4].
With these premises it was undeniable wondering about whether quantum computers could solve natural computational problems faster than classical computers. To this question Shor answered in 1994 with a paper [5] in which he presented

polynomial-time quantum algorithms to solve the integer factorisation and discrete logarithm problems for which no efficient classical algorithms exist. The impact on modern public key cryptography is obvious: large enough quantum computers will break factorisation-based cryptosystems, such as RSA, as well as cryptosystems based on the discrete logarithm, such as elliptic curve cryptosystems.

1.1 Post-Quantum Cryptography

With the term "*Post-quantum cryptography*" we refer to the science of protecting information against both quantum and classical attacks, as well as to the collection of tools that accomplish this task. Unfortunately, the adoption of post-quantum cryptography is not cost-free. The post-quantum hard problems, except for hash inversion, have been studied less than integer factorisation and the discrete logarithm problem. Consequently a post-quantum hard problem inevitably conveys less security assurance compared to a classic alternative. The reason is due to the greater potential of future improvements on quantum attacks. Additionally, many of the hard problems that hold promise of resisting attacks on quantum computers require far greater memory and bandwidth impeding their adoption into cryptosystems for low-cost devices.

1.1.1 Shor's algorithm

Theorised in 1994 by Peter Williston Shor, his algorithm stays at the core of post-quantum cryptography. By making use of a quantum computer, Shor's algorithm can efficiently solve both the integer factorisation and the discrete logarithm problem.
The idea behind Shor's algorithm [4, 5] is to utilise quantum computing to compare the phases of prime numbers as sinus waves to factorise great integers. Using number theory, the problem of number factorisation can be converted into a search for the period of a really long sequence, or rather, the length at which a sequence repeats itself. Then this periodic pattern is run through a quantum computer which works as a computational interferometer creating an interference pattern. This will output the period, which can be processed using a classical computer, finally being able to factorise the initially given number.

1.1.2 NIST's Standardisation Challenge

Over the last decade there has been an intense research effort to find hard mathematical problems that would be at the same time quantum resistant and could be used to build new efficient cryptosystems.
Of great importance we can cite "*The Post-Quantum-Cryptography Standardization Challenge*" a competition started by the National Institute of Standards and Technology (NIST) in April 2016 accepting proposals for post-quantum protocols. It entered a first evaluation stage in November 2017 when NIST stopped accepting new algorithms for consideration. On 30 January 2019, the project went into the second evaluation stage, said *Round 2*, with NIST announcing 26 out of 69 original submissions as final competitors. This round may take up to 18 months before completion, after which there may be a third round and only then official standard algorithms will be chosen.
The final algorithms will be rated into a five quantum level list which can be represented as below:

Quantum Level	Security	Reference protocol
I	Comparable to or greater than a block cipher with a 128-bit key against an exhaustive key search	AES128
II	Comparable to or greater than a 256-bit hash function against a collision search	SHA256
III	Comparable to or greater than a block cipher with a 192-bit key against an exhaustive key search	AES192
IV	Comparable to or greater than a 384-bit hash function against a collision search	SHA384
V	Comparable to or greater than a block cipher with a 256-bit key against an exhaustive key search	AES256

As a final note, most published post-quantum public-key schemes are focused on the following approaches [1]:

- Hash-based cryptography (e.g. Merkle's hash-tree public-key signature system);
- Multivariate quadratic-equations cryptography (e.g. HFE signature scheme);
- Lattice-based cryptography (e.g. NTRU encryption scheme);
- Code-based cryptography (e.g. McEliece encryption scheme, Niederreiter encryption scheme).

Another possible scheme is the newer "*Supersingular elliptic curve isogeny*"-based cryptography, central point of this thesis.

1.2 This Thesis

In this thesis we want to present a C implementation of a post-quantum protocol introduced in [9] by Vitse. The proposed algorithm exploits the supersingular isogeny problem, which is quantum resistant, for the key generation. Thereafter it exploits a zero knowledge protocol to exchange some data between the parties without revealing unnecessary information.
By implementing this protocol first we show that classical computers are still effective in a post-quantum reality or, viceversa, quantum computers are not compulsory when implementing post-quantum algorithms. Most importantly we adapted an open source library and created something new, useful and effective against the upcoming developments in the cryptography field.

In chapter 2 we will introduce the elliptic curve cryptography providing a brief but exhaustive disclosure on elliptic curves and isogenies among other topics. Whenever possible we will discuss on state-of-the-art optimisations also presenting examples when assumed necessary. Chapter 3 is the *theoretical core* of this thesis since it is focused on two cryptography protocols (sections 3.1, 3.2) which can be used in symbiosis to create third (section 3.3). We also want to supply a thorough security analysis on the latter before introducing the chapter 4. This chapter is our *practical core*: there will be a full commentary on the implementation of SIDH-based OT protocol from section 3.3.

Chapter 2

Elliptic Curve Cryptography

The Elliptic Curve Cryptography (ECC) is an alternative to RSA public key scheme based on the mathematics of elliptic curves. Cryptographic algorithms build on ECC require elliptic curves over finite fields and provide the same security offered by other public key schemes while requiring much smaller keys. ECC main applications are in encryption via public key protocols but it can also be used in symmetric algorithms, digital signatures, pseudo-random number generators and many others.
ECC security lies on the Elliptic Curve Discrete Logarithm Problem (ECDLP) which will be discussed in the following section after a brief introduction to elliptic curves to better understand how they work.

2.1 Elliptic Curves

An elliptic curve [22] is a cubic curve defined over a field $\mathbb{K}$ and having a K-rational point. It is important to note that the K-rational point can be the point at infinity expressed projectively as $\mathcal{O} = [0 : 1 : 0]$. Whenever the field $\mathbb{K}$ has characteristic $char(\mathbb{K}) \neq \{2, 3\}$, an elliptic curve can be defined in an affine form along with a smoothness condition and the point at infinity $\mathcal{O}$ as expressed in [23]:

$$\textit{Weierstrass equation:} \begin{cases} y^2 = x^3 + ax + b \\ 4a^3 \neq 27b^2 \end{cases} \bigcup \{\mathcal{O}\} \tag{2.1}$$

Point of an elliptic curve E form a group $(E, +)$ with group law $+$ known as Point Addition [24, 25]. A rigorous definition can be found in [23]:

Definition 2.1.1. Group Law Let $P, Q \in E$, two points on the elliptic curve E, let L be the line through P and Q (if $P = Q$, let L be the tangent line to E at P), and let R be the third point of intersection of L with E. Let now L' be the line through R and $\mathcal{O}$. Then L' intersects the curve E at R, $\mathcal{O}$, and a third point. We denote that third point by $P + Q$.

We may now briefly list some of the most important properties for an elliptic curve from a cryptography point of view:

- The Point Addition allows to sum two different points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ of the curve obtaining a third point $R = (x_R, y_R) = P + Q$ computed as follows:

$$\begin{cases} m = \dfrac{y_P - y_Q}{x_P - x_Q} \\ x_R = m^2 - (x_P + x_Q) \\ y_R = m(x_P - x_R) - y_P \end{cases}$$

- Following the group law definition it is evident that when $P = Q$ we have to consider the tangent line to the curve E. In this particular case the group law is called Point Doubling and the line's slope coefficient m changes to $m = \dfrac{3x^2 + a}{2y}$;

- The curve points form an *abelian group* [25] meaning that the group law is commutative, hence $R = P + Q = Q + P$;

- By adding a point P to itself k times it is possible to compute a scalar multiplication. This operation can be written as $[k]P$;

- Given a cyclic generator G of an elliptic curve, its order is the minimum number k such that $[k]G = \mathcal{O}$. The curve generated by G is denoted as $E : \langle G \rangle$ and has cardinality equal to G's order;

- An elliptic curve, defined in a finite field $\mathbb{F}_p$ with p prime greater than 3, having cardinality $p + 1$, is said *Supersingular*, otherwise is said Ordinary.

- Given the elliptic curve $E : y^2 = x^3 + ax + b$ over a field with characteristic different than 2 and 3, its j-invariant is defined as follows:

  $$j(E) = 1728 \cdot \frac{4a^3}{4a^3 + 27b^2}$$

  The j-invariant uniquely defines an elliptic curve such that it could be thought as its hash. For this very reason, the j-invariant is commonly used in practical ECC implementations.

At this point we can introduce the concept of Elliptic Curve Discrete Logarithm Problem (ECDLP). This problem is considered hard for classical security and is the fundamental security consideration upon which ECC schemes are based on:

Definition 2.1.2. ECDLP Given an elliptic curve E, two of its points G, P, find an integer k such that $P = [k]G$.

In figure 2.1 we can see two common affine representations of an elliptic curve: on the left $a = -3$, $b = 1$, on the right $a = -2$, $b = 2$.

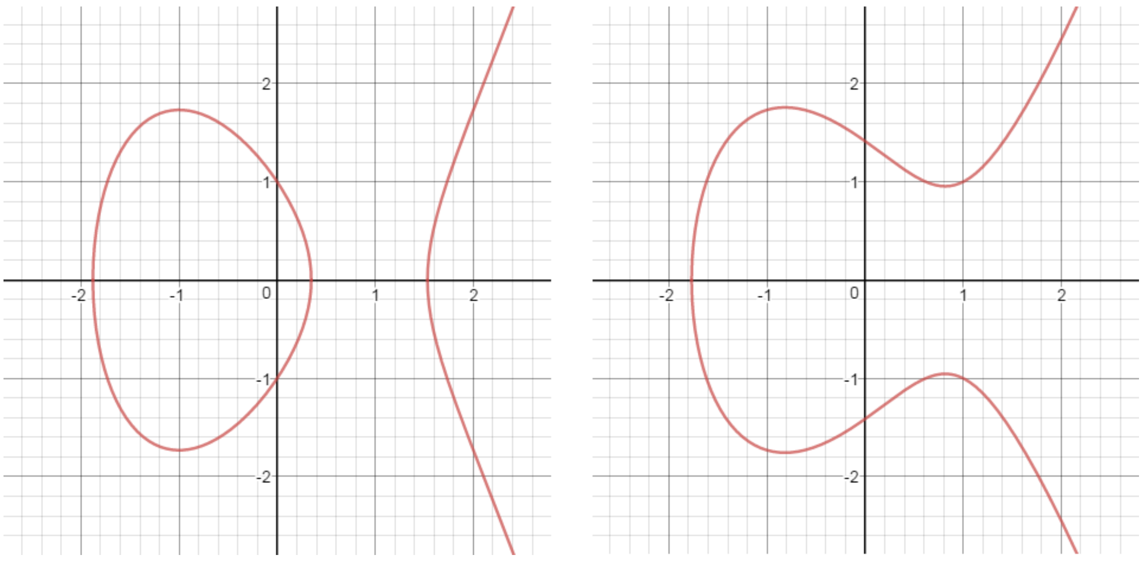

Figure 2.1: Two affine elliptic curves

2.2 Important concepts

In this section are listed many fundamental mathematical concepts that will be used in the next chapters.

2.2.1 m-torsion group

Given an elliptic curve E defined over a field K with characteristic p, an integer $m \neq 0$, the m-torsion group $E[m]$ is defined as follows:

$$E[m] \simeq (\mathbb{Z}/m\mathbb{Z})^2 \ \textit{if}\ p \nmid m$$

Otherwise in case we have $m = p^i$ then:

$$E[p^i] \simeq \begin{cases} \mathbb{Z}/p^i\mathbb{Z} & \forall i \geq 0 \text{ if } E \text{ is Ordinary} \\ \{\mathcal{O}\} & \forall i \geq 0 \text{ if } E \text{ is Supersingular} \end{cases}$$

As consequence of above, a supersingular curve has no points of order p.

2.2.2 Isomorphic curves

Given two groups $(G, \circ)$, $(H, *)$, the function $\phi : G \to H$ is said group homomorphism if holds:

$$\phi(g \circ h) = \phi(g) * \phi(h) \quad \forall g, h \in G$$

If the homomorphism is also bijective then it is called *isomorphism*, it is represented as $G \simeq H$ and the two groups have the same algebraic structure.
Since the points of an elliptic curve form a group [25], one may say that there could be a homomorphism and an isomorphism between elliptic curves.This is true and, in fact, two elliptic curves are said *isomorphic* over an algebraic closure $\overline{\mathbb{K}}$ if and only if they have the same j-invariant.

2.2.3 Isogenies

An isogeny is a surjective group homomorphism between two elliptic curves. Moreover, given two elliptic curves E, E' and the map $\phi : E \to E'$, the following are equivalent:

- ϕ is a surjective group homomorphism said *isogeny*
- ϕ is a group homomorphism with finite kernel
- ϕ is an algebraic, non constant map of projective varieties which maps the point at infinity $\mathcal{O}$ of E on the point at infinity $\mathcal{O}'$ of E'.

Two elliptic curves E, E' are thus said *isogenous* if there exists an isogeny between them. Moreover, the isogeny preserves the cardinality of each curve hence we can say that two elliptic curves are isogenous if and only if they have the same cardinality $\#E = \#E'$.

The isogeny equation can be computed via Velù's formulae: let $E : y^2 = x^3 + ax + b$ be an elliptic curve, P a generic point of E, $\langle K \rangle$ a cyclic group generated by K; let $E' \simeq E/\langle K \rangle$ is the quotient elliptic curve such that $\phi : E \rightarrow E'$ is the isogeny between the two curves, with kernel $\langle K \rangle$. The isogeny has equation:

$$\phi(P) = \left(x(P) + \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(x(P+Q) - x(Q) \Big), \, y(P) + \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(y(P+Q) - y(Q) \Big) \right) \tag{2.2}$$

Note that with $x(P)$ and $y(P)$ we define respectively the x and y coordinate of P. The image curve $E' \simeq E/\langle K \rangle : y^2 = x^3 + a'x + b'$ has parameters a', b' defined as follows:

$$a' = a - 5 \cdot \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(3x(Q)^2 + a \Big) \tag{2.3}$$

$$b' = b - 7 \cdot \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(5x(Q)^3 + 3ax(Q) + b \Big) \tag{2.4}$$

The operations in 2.2, 2.3 and 2.4 require one computation for each $Q \in \langle K \rangle \setminus \{\mathcal{O}\}$ hence the computational complexity of Velù's formulae is $\mathcal{O}(\# \langle K \rangle)$ where $\# \langle K \rangle$ denotes the cardinality of $\langle K \rangle$.
An example of isogeny [10, 26] is shown in figure 2.2 where the left curve $E/\mathbb{F}_{11} : y^2 = x^3 + x$ is mapped on the right one $E'/\mathbb{F}_{11} : y^2 = x^3 - 4x$. The kernel is the

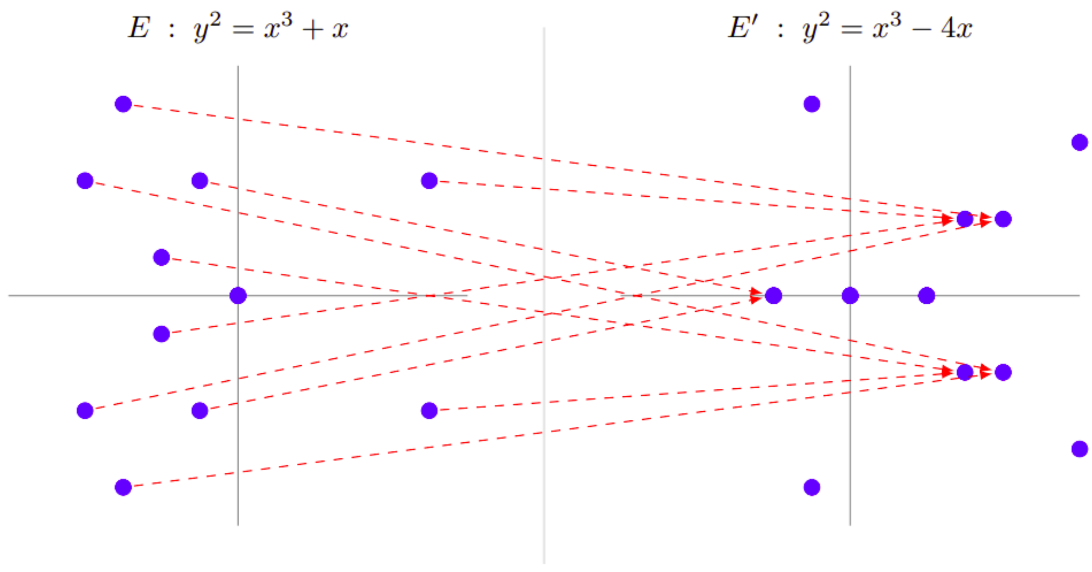

Figure 2.2: Isogeny mapping between E and E'

point $(0,0)$ on E and the isogeny has parameters:

$$\phi(x,y) = \left(\frac{x^2+1}{x}, y\frac{x^2-1}{x^2}\right).$$

Degree

Following [6, 26], let $\phi : E \rightarrow E'$ be an isogeny defined over a field $\mathbb{K}$, and let $\mathbb{K}(E)$, $\mathbb{K}(E')$ be the function fields of respectively E, E'. By composing ϕ with the functions of $\mathbb{K}(E')$ we obtain a subfield of $\mathbb{K}(E)$ that we denote by $\phi * \mathbb{K}(E')$. The degree of ϕ is thus defined as $deg(\phi) = [\mathbb{K}(E) : \phi * \mathbb{K}(E')]$ and it is always finite. It is common in literature to abbreviate an isogeny ϕ of degree $deg(\phi) = d$ as d-isogeny and so we will hereafter.
Let E be an elliptic curve and G a finite subgroup of E then there is a *unique* elliptic curve $E' : E/G$ and a unique separable isogeny ϕ such that $ker\ \phi = G$ and $\phi : E \rightarrow E'$.

Dual Isogenies

For any isogeny $\phi : E \to E'$ between elliptic curves there exists a dual isogeny [26] $\widehat{\phi} : E' \to E$ such that $\widehat{\phi} \circ \phi = \phi \circ \widehat{\phi} = [deg\,\phi]$. Furthermore it is important noting that $deg\,\widehat{\phi} = deg\,\phi$, $\widehat{\widehat{\phi}} = \phi$.

2.2.4 Endomorphism

Isogenies from a curve to itself are called *endomorphisms*. Given an elliptic curve E, a point P of E and an integer m, the scalar multiplication-by-m defined by $[m] : P \to [m]P$ is an endomorphism of E. Its kernel is exactly the m-th torsion subgroup $E[m]$.

Frobenius Endomorphism

Let E be an elliptic curve defined over a field $\mathbb{K}$ with q elements, its Frobenius endomorphism is the map $\pi : (X : Y : Z) \to (X^q : Y^q : Z^q)$.
It also holds:

- $ker\ \pi = \{\mathcal{O}\}$
- $ker(\pi - 1) = E(\mathbb{K})$

2.2.5 Bases and Weil Pairing

Given an elliptic curve E defined over a finite prime field $\mathbb{F}_p$, a basis for the curve E is a pair $(P, Q) \in E$ of linearly independent points such that the set of all their linear combinations forms a cyclic subgroup of E. To be more precise one can say that two points P, Q of an elliptic curve are linearly independent if and only if $P \neq uQ$, $\forall u \in \mathbb{Z}$, viceversa $Q \neq vP$, $\forall v \in \mathbb{Z}$; moreover we can write $\langle P, Q\rangle \in E$.
If both points have the same order k then the set of all their linear combinations form an elliptic curve of order k denoted as $\langle P, Q\rangle = E[k] \subseteq E$.
How to check two points' linear independence is possible thanks to the Weil Pairing [25].

Definition 2.2.1. Weil Pairing Let E be an elliptic curve over a field $\mathbb{K}$ and let n be an integer not divisible by the characteristic of $\mathbb{K}$. Then $E[n] \simeq (\mathbb{Z}/n\mathbb{Z})^2$. Let $\mu_n = \{x \in \bar{\mathbb{K}} | x^n = 1\}$ be the group of n-th roots of unity in $\bar{\mathbb{K}}$. It is a cyclic group of order n and any generator ζ of μ_n is called a *primitive n-th root of unity.* The Weil pairing is hence defined as $e_n : E[n] \times E[n] \to \mu_n$.

For a basis (P, Q) of $E[n]$ the Weil pairing $e_n(P, Q)$ is primitive n-th root of unity. In this case we say (P, Q) are linearly independent and the basis has full order.

2.3 Various optimisations

This section aims to sum up some of the many optimisations used in ECC practical implementations.

2.3.1 Montgomery Curves

A Montgomery curve over a field $\mathbb{K}$, with p prime, is a special form of an elliptic curve with affine equation:

$$M_{(A,B)} : By^2 = x(x^2 + Ax + 1) \tag{2.5}$$

where the parameters A, B are in $\mathbb{K}$ and satisfy $B \neq 0$, $A^2 \neq 4$. In projective coordinates $(X : Y : Z)$ with $x = X/Z$, $y = Y/Z$ the equation becomes:

$$M_{(A,B)} : BY^2Z = X(X^2 + AXZ + Z) \tag{2.6}$$

The latter has a unique point at infinity $\mathcal{O} = (0 : 1 : 0)$ and it is the **only** point where $Z = 0$.

There are many improvements on choosing a Montgomery curve over a simple Weierstrass elliptic curve. For a complete discussion about these special curves it is recommended to read [8]; for this thesis the most relevant speed up are:

- reduced number of operations for the group law;
- only x-coordinates operations are possible thus halving the overall operations when computing the scalar multiplication;

- the Point Addition can be further improved: when adding the points P, Q, the knowledge of a third point $Q - P$ reduces the overall computations. In this case we talk about Differential Addition.

Other important improvements rely on a different curve equation and precomputing some constants used in scalar multiplication and j-invariant calculation [10]. First of all it is important to remember that the notation $(X_p : Z_p)$ represents a projective tuple in $\mathbb{P}^1$ over $\mathbb{F}_{p^2}$, also the affine x-coordinate can be represented as $x_p = X_p/Z_p$. Now, starting from the affine curve $E_a/\mathbb{F}_{p^2} : y^2 = x^3 + ax^2 + x$, as done in [11, 27], we can write the equivalence $(A : C) \sim (a : 1)$ which transforms the curve into:

$$E : CBy^2 = Cx^3 + Ax^2 + Cx \tag{2.7}$$

This new equation allows to reduce the number of inversions, the most costly operation, with the little disadvantage of slightly increasing the total operations needed. The pair $(A_{24} : C_{24})$ denotes the constant $(a - 2)/4$ in $\mathbb{P}^1$ which can be expressed as:

$$(A_{24} : C_{24}) \sim (a - 2 : 4)$$

Lastly the constants used in practice are:

$$\begin{aligned}
(A_{24}^{+} : C_{24}) &\sim (A + 2C : 4C) \\
(A_{24}^{+} : A_{24}^{-}) &\sim (A + 2C : A - 2C) \\
(a_{24}^{+} : 1) \quad &\sim (A + 2C : 4C)
\end{aligned}$$

j-invariant

The j-invariant of a standard projective Montgomery curve depends only on the parameter A and is computed as follows:

$$j(M_{(A,B)}) = \frac{256(A^2 - 3)^3}{A^2 - 4}$$

In implementation scenarios we use the equation 2.7 which requires the computation of $(A : C)$ to compute the new j-invariant:

$$j(M_{(A,B)}) = \frac{256(A^2 - 3C^2)}{C^4(A^2 - 4C^2)}.$$

2.3.2 Weil Pairing

The base algorithm for the Weil Pairing requires, roughly, $\mathcal{O}(n)$ operations which is greatly inefficient for large order bases; as an example, typical orders are 2^{250} and 3^{159}. We now show a faster way to run the algorithm as described in [7].
Let P, Q be two points both of order mn so that they are in $E[mn]$; the n-th power of the pairing $e_{mn}(P,Q)$ can be computed as follows:

$$e_{mn}(P,Q)^n = e_m([n]P,[n]Q)$$

We now consider an example: let be $m = 4$, $n = 2^{370}$ such that $mn = 2^{372}$, it is possible [7, 25] to show that

$$e_{mn}(P,Q)^n = e_{4\cdot 2^{370}}(P,Q)^{2^{370}} = e_4([2^{370}]P,[2^{370}]Q)$$

If P and Q have order 2^{372} then it is also true that $P' = [2^{370}]P$ and $Q' = [2^{370}]Q$ both have order 4.
Another speed up: all the involved checks are possible considering the only x coordinates of input basis. For the following procedure we are going to denote with $x(P)$ the affine x-coordinate of the point P, and with $X(P)$, $Z(P)$ its projective X- and Z-coordinate respectively.
At this point it is mandatory to check that $x(P') \neq x(Q')$ but since ECC implementations work with projective points we can transform the latter expression into $\frac{X(P')}{Z(P')} \neq \frac{X(Q')}{Z(Q')}$ and finally $X(P')Z(Q') \neq X(Q')Z(P')$ which is called projective cross-multiplication.
Lastly, we need to check:

- Said $(X : Z) = x([2]P')$, then it must hold true that $Z \neq 0$ meaning that $[2]P'$ is not the point at infinity;
- Said $(\overline{X} : \overline{Z}) = x([4]P')$, then it must hold true that $\overline{Z} = 0$ meaning that $[4]P'$ is the point at infinity.

Same checks have to be performed on Q'.
At this point if both P' and Q' have passed all these tests then $[4]P' = [4]Q' = \mathcal{O}$ finally proving that P and Q both have order 2^{372}. The pair (P,Q) is then a basis of order 2^{372}.

2.3.3 Isogenies

We have seen in section 2.2.3 that, said $\langle K \rangle$ a cyclic group, the worst drawback of Velù's formulae is that they require $\mathcal{O}(\# \langle K \rangle)$ operations hence making their implementation intractable. A great improvement is possible when the order of K is smooth, meaning it is a power of a prime number, via composition of small degree isogenies [7, 9] obtaining a cost drop to $\mathcal{O}\Big(\log(\# \langle K \rangle) \cdot \log(\log(\# \langle K \rangle))\Big)$. Since typical applications use kernels of order 2^n or 3^m, this speed up is achievable.

Costello et al. [7] write about computing large degree isogenies via small degree isogenies. We now describe the idea of composing isogenies.
Given an elliptic curve E, a cyclic subgroup $\langle R \rangle \subseteq E[l^e]$ of order l^e, with l prime, there is a unique isogeny $\phi_R : E \to E/\langle R \rangle$ of degree l^e defined over $\mathbb{F}_{p^2}$ with kernel $\langle R \rangle$. Set $E_0 = E$, $R_0 = R$, such isogeny can be computed by composing e l-isogenies [12], in this way we iteratively compute $E_{i+1} \simeq E_i/\langle [l^{e-i-1}]R_i \rangle$ for $0 \leq i < e$. Note that the point R_i is an l^{e-i}-torsion point and the point $\langle [l^{e-i-1}]R_i \rangle$ has order l. The resulting isogeny is then $\phi = \phi_{e-1} \circ \ldots \circ \phi_0$ having degree l^e as required.
Even though we may compute the curve E_{i+1} and its isogeny phi_i via Velù's formulae, this strategy is still too onerous. A better strategy is an isogeny walk, as shown in figure 2.3 for a problem with $e = 6$. Bullets "$\bullet$" represent points: those at the same horizontal level have the same order, those lying on the same left-diagonal belong to the same curve. Dashed edges "$---$" are directed from top to bottom: leftward edges represent a multiplication-by-l, rightwards edges represent the evaluation of an l-isogeny.
At the beginning of the algorithm only the root R is known and our aim is to compute all the points on the bottom line.

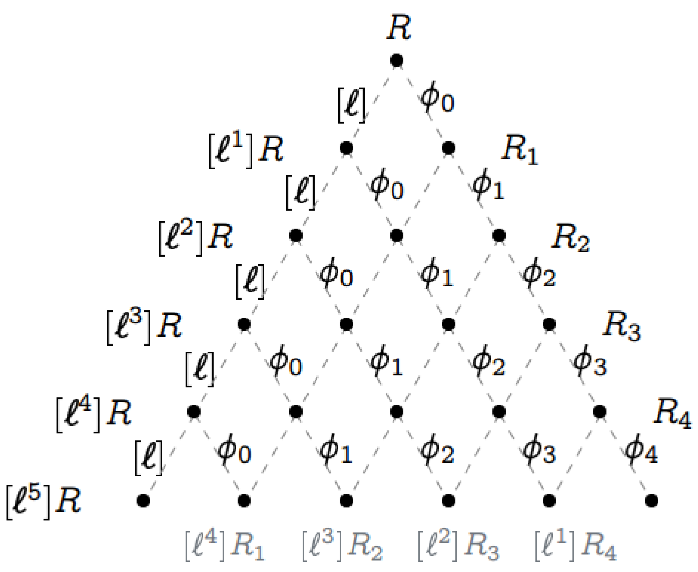

Figure 2.3: l-isogeny walks up to degree l^5

At this point we need a "*strategy*" [7,12] to minimise the operations in traversing the graph in figure 2.3. We may define a strategy as a binary tree topology that can be hence decomposed into two sub-strategies on strictly smaller leaf sets. The overall cost for the strategy is the sum of the sub-strategy costs plus the cost for moving down the tree along the edges to the roots of the sub-strategies. Given the costs of all optimal sub-strategies, one can select the optimal strategy by going through all possibilities of combining two of them. A strategy can be stored in a simple list L of integers of length equal to the total number of leaves. The i-th entry $L[i]$ in the list characterises the sub-strategy on a graph with i leaves.
For the given walk there are seven well formed strategies as shown in figure 2.4.

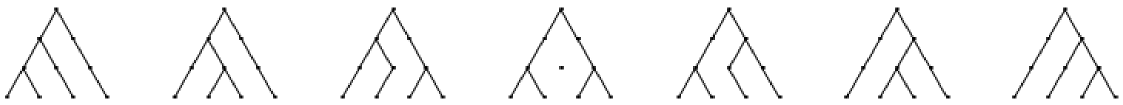

Figure 2.4: Well formed strategies for $e = 4$

Among all strategies, there are optimal ones which minimise the number of operations and these can be precomputed offline. As result of all these improvements, large smooth degree isogeny computations, say l^e-isogeny, are done with complexity $\mathcal{O}(e \log e)$.

Small degree isogenies

In this section we show the alternative formulae to Velù's used in real applications. Since we are talking about 2-parties protocols, only two different isogenies should be studied: 2- and 3-isogenies. In the former case, a further speed up is possible if we consider 4-isogenies thus halving the total operations needed [11].

2-isogenies

Let $E_{A,B}$, $E_{A',B'}$ be two elliptic curves, $(x_2, y_2) \in E_{A,B}$ be a point of order 2 having $x_2 \neq 0$. Let $\phi_2 : E_{A,B} \to E_{A',B'}$ be the unique 2-isogeny with kernel $\langle (x_2, y_2) \rangle$. The image curve can be computed as follows:

$$E_{A',B'} : \big(A', B'\big) = \Big(2 \cdot \big(1 - 2x_2^2\big)\,, \quad Bx_2\Big)$$

For any point $P = (x_P, y_P) \notin \langle (x_2, y_2) \rangle$ of $E_{A,B}$, its image $\phi_2 : (x_P, y_P) \mapsto (x_{\phi_2}, y_{\phi_2})$ can be computed as:

$$\begin{aligned} x_{\phi_2} &= \frac{x_P^2 x_2 - x_P}{x_P - x_2} \\ y_{\phi_2} &= y_P \cdot \frac{x_P^2 x_2 - 2x_P x_2^2 + x_2}{\big(x_P - x_2\big)^2} \end{aligned}$$

4-isogenies

Let $E_{A,B}$, $E_{A',B'}$ be two elliptic curves, $(x_4, y_4) \in E_{A,B}$ be a point of order 4 having $x_4 \neq \pm 1$. Let $\phi_4 : E_{A,B} \to E_{A',B'}$ be the unique 4-isogeny with kernel $\langle (x_4, y_4) \rangle$. The image curve can be computed as follows:

$$E_{A',B'} : \big(A', B'\big) = \Big(4x_4^4 - 2, \quad -x_4(x_4^2 + 1) \cdot B/2\Big)$$

For any point $P = (x_P, y_P) \notin \langle (x_4, y_4) \rangle$ of $E_{A,B}$, its image $\phi_4 : (x_P, y_P) \mapsto (x_{\phi_4}, y_{\phi_4})$ can be computed as:

$$
\begin{aligned}
x_{\phi_4} &= \frac{-x_P(x_P x_4^2 + x_P - 2x_4) \cdot (x_P x_4 - 1)^2}{(x_P - x_4)^2 \cdot (2x_P x_4 - x_4^2 - 1)} \\
y_{\phi_4} &= y_P \cdot \frac{-2x_4^2(x_P x_4 - 1)\Big(x_P^4(x_4^2+1) - 4x_P^3(x_4^3 + x_4) + 2x_P^2(x_4^4 + 5x_4^2) - 4x_P(x_4^3 + x_4) + x_4^2 + 1\Big)}{(x_P - x_4)^3(2x_P x_4 - x_4^2 - 1)^2}
\end{aligned}
$$

3-isogenies

Let $E_{A,B}$, $E_{A',B'}$ be two elliptic curves, $(x_3, y_3) \in E_{A,B}$ be a point of order 3. Let $\phi_3 : E_{A,B} \to E_{A',B'}$ be the unique 3-isogeny with kernel $\langle (x_3, y_3) \rangle$. The image curve can be computed as follows:

$$E_{A',B'} : \big(A', B'\big) = \Big(\big(Ax_3 - 6x_3^2 + 6\big)\, x_3, \quad Bx_3^2 \Big)$$

For any point $P = (x_P, y_P) \notin \langle (x_3, y_3) \rangle$ of $E_{A,B}$, its image $\phi_3 : (x_P, y_P) \mapsto (x_{\phi_3}, y_{\phi_3})$ can be computed as:

$$
\begin{aligned}
x_{\phi_3} &= \frac{x_P(x_P x_3 - 1)^2}{(x_P - x_3)^2} \\
y_{\phi_3} &= y_P \cdot \frac{(x_P x_3 - 1)(x_P^2 x_3 - 3x_P x_3^2 + x_P + x_3)}{(x_P - x_3)^3}
\end{aligned}
$$

2.4 Isogeny-based Cryptography

Traditional Elliptic Curve Cryptography (ECC) relies its security on the Discrete Logarithm Problem (DLP) which can be stated as: "*Let E be an elliptic curve defined over $\mathbb{F}_p$ with p prime greater than* 3. *Given two points P, Q of the curve, find an integer k such that $Q = [k]P$*".
That said, Shor's algorithm presents a threat to DLP based cryptography since it can compute discrete logarithms in polynomial time. A quantum resistant solution involves the use of the *computational supersingular isogeny problem* stated as follows: "*given two supersingular elliptic curves E and E', find an isogeny ϕ such that $\phi : E \to E'$*". While for ordinary elliptic curves there exists a sub-exponential quantum attack, for the given problem does not exist thus making it post-quantum resistant.

Chapter 3

SIDH-based OT

The protocol studied and implemented in this thesis relies on an adaptation of Supersingular Isogeny Diffie-Hellman (SIDH) as a base structure for an Oblivious Transfer (OT) protocol. By joining these two protocols together it is possible to construct a post-quantum multi-party scheme.
First things first, we explain how SIDH [7,11,12] and OT separately work, then how it is possible to merge them into a new protocol [9].

3.1 SIDH Key Exchange

The Supersingular Isogeny Diffie-Hellman is a protocol developed by Jao and De Feo in 2011 [12] which offers a great ratio efficiency-security having keys smaller than other post-quantum protocols (lattice-based and code-based), moreover they are smaller than traditional Diffie-Hellman public keys.
The scheme begins with a shared supersingular elliptic curve and two parties, Alice and Bob, who get assigned two different torsion groups. After few isogenies and data exchanges both parties have two curves with the same j-invariant. The latter is then used as shared key hence used to encrypt and decrypt data between the parties. Let's now describe the protocol deeper in details.

All the supersingular elliptic curves used in SIDH are defined over $\mathbb{F}_{p^2}$ where $p = l_A^{e_A} l_B^{e_B} \pm 1$ prime, l_A and l_B small primes, e_A and e_B integers such that $l_A^{e_A} \approx l_B^{e_B}$.

Typical choices for e_A fall within the range $[100, 500]$ according to the security level desired. All SIDH implementations consider $l_A = 2$, $l_B = 3$ and so we will henceforth. In order to simplify the notation we will be using $e_A = n$ and $e_B = m$ and assign $l_A^{e_A} = 2^n$ to Alice, $l_B^{e_B} = 3^m$ to Bob.
Initial public parameters:

- A supersingular elliptic curve $E \in \mathbb{F}_{p^2}$ with $p = 2^n 3^m \pm 1$ prime;
- A bases $(U, V) \in E[2^n]$ in $\mathbb{F}_{p^2}$;
- A bases $(P, Q) \in E[3^m]$ in $\mathbb{F}_{p^2}$.

The protocol proceeds as follows:

Alice	**Bob**
Chooses $x_A, y_A \in \mathbb{Z}/2^n\mathbb{Z}$ randomly, at least one of them coprime to 2	Chooses $x_B, y_B \in \mathbb{Z}/3^m\mathbb{Z}$ randomly, at least one of them coprime to 3
Computes $R_A = x_A \cdot U + y_A \cdot V$	Computes $R_B = x_B \cdot P + y_B \cdot Q$
Computes the curve $E_A \simeq E/\langle R_A \rangle$	Computes the curve $E_B \simeq E/\langle R_B \rangle$
Computes the isogeny $\phi_A : E \to E_A$	Computes the isogeny $\phi_B : E \to E_B$

$$\xleftrightarrow{\text{Exchange their new curves } E_A,\ E_B}$$

Computes $P' = \phi_A(P)$, $Q' = \phi_A(Q)$	Computes $U' = \phi_B(U)$, $V' = \phi_B(V)$

$$\xleftrightarrow{\text{Exchange their new points } U',\ V',\ P',\ Q'}$$

Computes $E_{BA} \simeq E_B/\langle x_A U' + y_A V' \rangle$	Computes $E_{AB} \simeq E_A/\langle x_B P' + y_B Q' \rangle$

At the end of the protocol both parties have $E_{AB} \simeq E_{BA}$ having the same j-invariant which can be used as shared secret.
Figure 3.1 shows the protocol graphically to ease the comprehension.

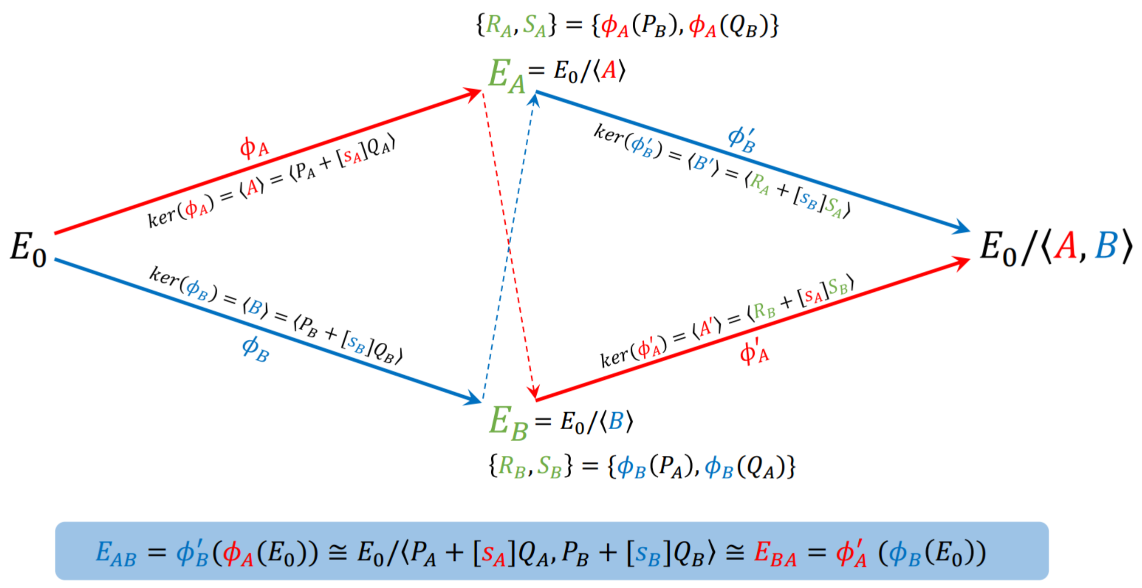
$\{R_A, S_A\} = \{\phi_A(P_B), \phi_A(Q_B)\}$
$E_A = E_0/\langle A\rangle$
ϕ_A
$ker(\phi_A) = \langle A\rangle = \langle P_A + [s_A]Q_A\rangle$
ϕ'_B
$ker(\phi'_B) = \langle B'\rangle = \langle R_A + [s_B]S_A\rangle$
E_0
$E_0/\langle A, B\rangle$
$ker(\phi_B) = \langle B\rangle = \langle P_B + [s_B]Q_B\rangle$
ϕ_B
$ker(\phi'_A) = \langle A'\rangle = \langle R_B + [s_A]S_B\rangle$
ϕ'_A
$E_B = E_0/\langle B\rangle$
$\{R_B, S_B\} = \{\phi_B(P_A), \phi_B(Q_A)\}$
$E_{AB} = \phi'_B(\phi_A(E_0)) \cong E_0/\langle P_A + [s_A]Q_A, P_B + [s_B]Q_B\rangle \cong E_{BA} = \phi'_A(\phi_B(E_0))$

Figure 3.1: SIDH schema. Alice private parameters are in red, Bob's are in blue. Public parameters are green.

3.1.1 Correctness

Alice computes:

$$\begin{aligned}
E_{BA} &\simeq E_B / \langle x_A U' + y_A V' \rangle \\
&\simeq E_B / \langle x_A \phi_B(U) + y_A \phi_B(V) \rangle \\
&\simeq E_B / \langle \phi_B(x_A U + y_A V) \rangle \\
&\simeq E_B / \langle x_A U + y_A V \rangle \\
&\simeq E_B / \langle R_A \rangle \\
&\simeq E / \langle R_B \rangle / \langle R_A \rangle \\
&\simeq E / E[3^m] / E[2^n]
\end{aligned}$$

Bob computes:

$$\begin{aligned}
E_{AB} &\simeq E_A / \langle x_B P' + y_B Q' \rangle \\
&\simeq E_A / \langle x_B \phi_A(P) + y_B \phi_A(Q) \rangle \\
&\simeq E_A / \langle \phi_A(x_B P + y_B Q) \rangle \\
&\simeq E_A / \langle x_B P + y_B Q \rangle \\
&\simeq E_A / \langle R_B \rangle \\
&\simeq E / \langle R_A \rangle / \langle R_B \rangle \\
&\simeq E / E[2^n] / E[3^m]
\end{aligned}$$

Therefore $E_{BA} \simeq E/E[3^m]/E[2^n] \simeq E/E[2^n]/E[3^m] \simeq E_{AB}$ proving that Alice and Bob now share a curve isomorphic to their party's curve.

3.1.2 Security

Before introducing SIDH's security problem it is important to state the *Standard Isogeny* problem: "Given a prime $p = 2^n 3^m \pm 1$, two supersingular elliptic curves E, E_A over $\mathbb{F}_{p^2}$, determine the 2^n-isogeny $\phi_A : E \to E_A$".
SIDH's security is similar to the *Standard Isogeny* problem but it adds some information to the attacker. One can express SIDH's problem [13] as follows: "Given a prime $p = 2^n 3^m \pm 1$, two supersingular elliptic curves E, E_A over $\mathbb{F}_{p^2}$, determine the 2^n-isogeny $\phi_A : E \to E_A$ **also** knowing the basis $(P, Q) \in E[3^m]$ and $P' = \phi_A(P)$, $Q' = \phi_A(Q)$".
This problem is assumed to be as hard as the *Standard Isogeny* problem. Moreover we remember that classical cryptography algorithms (e.g., RSA) are still used and considered secure even though the progresses made in the last years; therefore even the existence of a quantum subexponential attack against Ordinary Elliptic Curves should not preclude the implementation of Supersingular Elliptic Curves. The latter, currently the most preferred, is more computationally efficient and all the known classical and quantum attacks against it are exponential. In fact the fastest known attack is Tani's Claw Finding attack which requires $\mathcal{O}(\sqrt[4]{p})$ operations on a classical computer and $\mathcal{O}(\sqrt[6]{p})$ on a quantum computer.

SIDH's primes p had initially been selected on accounting only a simplified running time of the Claw Finding attack without considering its spacial cost amounting to $\mathcal{O}(\sqrt[6]{p})$ qubits. As an example [17], in NIST's Challenge's Round 1 introduced in 1.1.2, in order to achieve a b-bit security level against known classical and quantum attacks SIDH primes p were selected with bitlength of approximately $6b$. Hence the 751-bit prime $p = 2^{372}3^{239} - 1$ was proposed for the 128-bit classical security level, respectively Quantum Level I. Moreover the 964-bit prime $p = 2^{486}3^{301} - 1$ was proposed for the 160-bit classical security level.
By NIST's Challenge Round 2, few revisions were made thus reaching: p_{434} for 128-bit classical and Quantum Level II security, p_{503} for 160-bit classical and Quantum Level III security, p_{610} for 192-bit classical and Quantum Level IV security, finally p_{751} for 256-bit classical and Quantum Level V security. The 964-bit curve was hence discarded.
A full disclosure about attacks and examples can be found in [14–16].

3.2 Oblivious Transfer

The *oblivious transfer*, or **OT**, is a multi-party cryptography scheme in which two or more parties are involved. A typical OT application is the secure function evaluation where every party holds an input for a given function. In this scenario, the output should be computed in a way such that no party has to reveal unnecessary information about their input. Correctness of the protocol is usually proved with a zero knowledge proof.
The oblivious transfer has many different implementations each of them achieving different yet similar goals. The base idea is to send one of many pieces of information to a second party while the sender has no knowledge of which piece has been sent. A classical implementation is the *Rabin OT* in which Alice, with a probability of 1/2, sends a simple bit to Bob. This scheme leaves Alice "oblivious" of whether Bob has received it or not.

There exists another variation proposed by Shimon Even, Oded Goldreich and Abraham Lempel called "*1 out of 2 Oblivious Transfer*", often written as $\binom{2}{1}$-OT,

which can easily be generalised to "*1 out of n OT*". In this variation, a party say B, receives one out of two (alternatively n) piece of information but the sender party, say A, does not know which piece B has received.
As shown in figure 3.2, A sends the pieces b_0, b_1 to B which, in turn, chooses a random integer c. At the end of the transfer B will have knowledge of the c-th piece b_c of A.

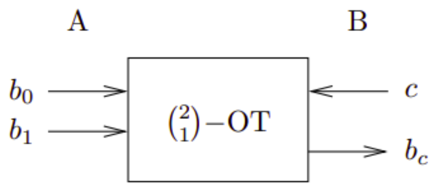

Figure 3.2: Simple $\binom{2}{1}$-OT

3.3 SIDH-based OT

Initial public parameters:

- A supersingular elliptic curve $E \in \mathbb{F}_{p^2}$ with $p = 2^n 3^m \pm 1$ prime;
- A bases $(P, Q) \in E[3^m]$ in $\mathbb{F}_{p^2}$;
- A secure symmetric encryption protocol Enc such that: $c = Enc(m, k)$, $m = Enc^{-1}(c, k)$;
- A key derivation function *KDF* such that k = *KDF(seed)*.

For simplicity we assume a $\binom{2}{1}$-OT in which Alice has *two* secrets $\{s_0, s_1\}$. The protocol then proceeds as follows:

Alice **Bob**

Computes *two* bases:

$(U_0, V_0) \in E[2^n]$,

$(U_1, V_1) \in E[2^n]$

Computes *two* curves and isogenies:

$E_{A,0} \simeq E/\langle R_0 \rangle$, $\phi_{A,0} : E \to E_{A,0}$

$E_{A,1} \simeq E/\langle R_1 \rangle$, $\phi_{A,1} : E \to E_{A,1}$

Computes *two* pairs of points:

$P_0 = \phi_{A,0}(P),\ Q_0 = \phi_{A,0}(Q)$,

$P_1 = \phi_{A,1}(P),\ Q_1 = \phi_{A,1}(Q)$

$$\xrightarrow{\text{Sends Bob } \textit{two} \text{ tuples } \{E_{A,0},\, P_0,\, Q_0\},\, \{E_{A,1},\, P_1,\, Q_1\}}$$

Chooses an integer $k \in [0, 1]$

Chooses an integer $b \in \mathbb{Z}/3^m\mathbb{Z}$

Computes the curve and j-invariant:

$E_B \simeq E/\langle P + bQ \rangle$, $j_B = j(E_B)$

Computes *two* bases:

$(U_0,\, V_0) \in E_{A,0}[2^n]$,

$(U_1,\, V_1) \in E_{A,1}[2^n]$

all having the same Weil Pairing:

$e(U_0,\, V_0) = e(U_1,\, V_1)$

Computes the curve and isogeny:

$E'_B \simeq E_{A,k}/\langle P_k + bQ_k \rangle$, $\phi'_B = E_{A,k} \to E'_B$

Computes the points:

$U'_k = \phi'_B(U_k),\ V'_k = \phi'_B(V_k)$

$$\xleftarrow{\text{Sends Alice } \{E'_B,\, U'_k,\, V'_k\} \text{ and the } \textit{two} \text{ basis } (U_0,\, V_0),\, (U_1,\, V_1)}$$

Computes *two* pairs $x_i,\, y_i \in \mathbb{Z}$:
$(x_0,\, y_0) \mid \phi_{A,0}(T_0) = x_0U_0 + y_0V_0$
$(x_1,\, y_1) \mid \phi_{A,1}(T_1) = x_1U_1 + y_1V_1$

Computes *two* curves and j-invariants:
$F_0 \simeq E'_B / \langle x_0U'_k + y_0V'_k \rangle\,,\, j_0 = j(F_0)$
$F_1 \simeq E'_B / \langle x_1U'_k + y_1V'_k \rangle\,,\, j_1 = j(F_1)$

Encrypts her *two* secrets:
$S_0 = Enc(s_0,\, KDF(j_0))$
$S_1 = Enc(s_1,\, KDF(j_1))$

$$\xrightarrow{\text{Sends Bob the } \textit{two} \text{ encrypted secrets } S_0,\, S_1}$$

Decrypts its chosen k-th secret:
$s_k = Enc^{-1}(S_k,\, KDF(j_B))$

What explained above can be generalised to a $\binom{n}{1}$-OT: every "*two*" in the sketch should be changed to n; Bob's k should be chosen in $[0, n-1]$ range.

3.3.1 Correctness

How can be able to decrypt the k-th Alice's secret? Since Alice encrypts her secrets with j_i, Bob must have that $j_k = j_B$. In order for this to happen, Alice's curve F_i must be isomorphic to E_B computed by Bob. We now show the correctness of the protocol. For the sake of simplicity we assume $i = k = 0$ without loss of generality.

$$\begin{aligned}
F_0 &\simeq E'_B / \langle x_0 U'_0 + y_0 V'_0 \rangle \\
&\simeq E'_B / \langle x_0 \phi'_B(U_0) + y_0 \phi'_B(V_0) \rangle \\
&\simeq E'_B / \langle \phi'_B(x_0 U_0 + y_0 V_0) \rangle \\
&\simeq E'_B / \langle x_0 U_0 + y_0 V_0 \rangle \\
&\simeq E'_B / \langle \phi_{A,0}(T_0) \rangle \\
&\simeq E_{A,0} / \langle P_0 + bQ_0 \rangle / \langle \phi_{A,0}(T_0) \rangle \\
&\simeq E_{A,0} / \langle P_0 + bQ_0 \rangle / \langle T_0 \rangle \\
&\simeq E / \langle R_0 \rangle / \langle P_0 + bQ_0 \rangle / \langle T_0 \rangle \\
&\simeq E / \langle R_0,\, T_0 \rangle / \langle P_0 + bQ_0 \rangle \\
&\simeq E / E[2^n] / \langle P_0 + bQ_0 \rangle \\
&\simeq E / \langle P_0 + bQ_0 \rangle \\
&\simeq E_B
\end{aligned}$$

3.3.2 Security

For this section it is important to first introduce few hard problems [9, 12] in order to ease the analysis.

XDSSI - ***Extended Decisional Supersingular Isogeny*** problem

Given two supersingular elliptic curves E and E' defined over $\mathbb{F}_{p^2}$, the pairs (U, V), (U', V') such that they both form a 2^n-order basis of E, and having Weil Pairing $e(U, V)^{3^m} = e(U', V')$; determine if there exists a 3^m-isogeny $\phi : E \to E'$ such that $\phi(U) = U'$ and $\phi(V) = V'$.

CSSI - ***Computational Supersingular Isogeny*** problem

Given a supersingular elliptic curve $E \in \mathbb{F}_{p^2}$, the bases $(U, V) \in E[2^n]$ and $(P, Q) \in E[3^m]$, an integer $a \in \mathbb{Z}/2^n\mathbb{Z}$, the isogeny $\phi_A : E \to E_A$ with kernel $ker\, \phi_A = \langle U + aV \rangle$, finally given E_A and the points $\phi_A(P)$, $\phi_A(Q)$; determine the isogeny kernel $ker\, \phi_A$.

SSCDH - *Supersingular Computational Diffie-Hellman Isogeny* problem

Let $\phi_A : E_0 \to E_A$ be an isogeny with kernel $\langle [m_A]P_A + [n_A]Q_A \rangle$ where m_A and n_A are chosen at random from $\mathbb{Z}/2^n\mathbb{Z}$ and not both divisible by 2. Let $\phi_B : E_0 \to E_B$ be an isogeny with kernel $\langle [m_B]P_B + [n_B]Q_B \rangle$ where m_B and n_B are chosen at random from $\mathbb{Z}/3^m\mathbb{Z}$ and not both divisible by 3.
Given the curves E_A, E_B and the points $\phi_A(P_B)$, $\phi_A(Q_B)$, $\phi_B(P_A)$, $\phi_B(Q_A)$, find the j-invariant of $E_{AB} \simeq E_0/\big\langle [m_A]P_A + [n_A]Q_A,\ [m_B]P_B + [n_B]Q_B \big\rangle$.

SSDDH - *Supersingular Decision Diffie-Hellman* problem

Given a tuple sampled with probability 1/2 from one of the following two distributions:

- $\Big(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_{AB}\Big)$, where all the parameters are in the *SSCDH* problem and $E_{AB} \simeq E_0/\big\langle [m_A]P_A + [n_A]Q_A,\ [m_B]P_B + [n_B]Q_B \big\rangle$;
- $\Big(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_C\Big)$, where all the parameters are in the *SSCDH* problem and $E_C \simeq E_0/\big\langle [m'_A]P_A + [n'_A]Q_A,\ [m'_B]P_B + [n'_B]Q_B \big\rangle$ and m'_A, n'_A (resp. m'_B, n'_B) are chosen at random from $\mathbb{Z}/2^n\mathbb{Z}$ (resp. $\mathbb{Z}/3^m\mathbb{Z}$) and not both divisible by 2 (resp. 3);

determine from which distribution the tuple is sampled.

2-inv-CSSI - *2-inverse Computational Supersingular Isogeny* problem

Let E, E_0, E_1 be three supersingular elliptic curves defined over $\mathbb{F}_{p^2}$ such that E_0 and E_1 are 2^n-isogenous to E with the corresponding isogenies $\phi_0 : E \to E_0$, $\phi_1 : E \to E_1$. Let (P, Q) be a basis of $E[3^m]$ and for each $i = \{0, 1\}$, let (U_i, V_i) be a basis of $E_i[2^n]$ and (x_i, y_i) be the coordinates in this basis of a generator of the dual isogeny $\widehat{\phi_i}$. Given the points $P, Q, U_0, V_0, U_1, V_1, \phi_0(P), \phi_0(Q), \phi_1(P), \phi_1(Q)$, find three supersingular elliptic curves E', F_0, F_1 and a basis $(U', V') \in E'[2^n]$ such that $F_0 \simeq E'/\langle x_0U' + y_0V' \rangle$ and $F_1 \simeq E'/\langle x_1U' + y_1V' \rangle$.

2-inv-DSSI - *2-inverse Decisional Supersingular Isogeny* problem

This problem uses the same notations of *2-inv-CSSI* problem for the following parameters: curves E, E_0, E_1, points $P, Q, U_0, V_0, U_1, V_1, \phi_0(P), \phi_0(Q), \phi_1(P), \phi_1(Q)$. Given the following settings:

- Bob sends the challenge oracle a supersingular elliptic curve E' and a basis $(U', V') \in E'[2^n]$;
- the oracle computes the supersingular curves $F_0 \simeq E'/\langle x_0U' + y_0V' \rangle$, $F_1 \simeq E'/\langle x_1U' + y_1V' \rangle$, $F_0 \simeq E'/\langle W_0 \rangle$, $F_0 \simeq E'/\langle W_1 \rangle$, where the points W_0 and W_1 are random points of $E'[2^n]$;
- the oracle chooses randomly, uniformly and independently two bits b_0, b_1. Then it outputs two pairs (C_0, C_0') and (C_1, C_1') of supersingular curves such that:

$$(C_0, C_0') = \begin{cases} (F_0, F_0') \text{ if } b_0 = 0 \\ (F_0', F_0) \text{ if } b_0 = 1 \end{cases} \quad \text{and } (C_1, C_1') = \begin{cases} (F_1, F_1') \text{ if } b_1 = 0 \\ (F_1', F_1) \text{ if } b_1 = 1 \end{cases}$$

- Bob must answer whether $b_0 = b_1$ or $b_0 \neq b_1$.

Bob's advantage in this game is defined as "*$\mathcal{P}$(correct answer) - 1/2*", where *$\mathcal{P}$(correct answer)* is the probability of answering correctly. Then the *2-inv-DSSI* problem is hard if no algorithm can achieve a non-negligible advantage for Bob in probabilistic polynomial time.

We have now completed the disclosure on hard problems and can finally approach a malicious Alice and malicious Bob analysis with all the preliminary knowledge required.

Malicious Alice

Malicious Alice's analysis focuses on Bob's message:

$$\xleftarrow{\text{Sends Alice } \{E_B', U_k', V_k'\} \text{ and the } \textit{two} \text{ basis } (U_0, V_0), (U_1, V_1)}$$

With this message, Alice knows $\{E'_B,\, U'_k,\, V'_k\}$, even more she knows that E'_B is 3^m-isogenous to one of her $E_{A,i}$: she might be able to recover Bob's secret k by finding which of her curves is isogenous to E'_B. This problem is addressed as *Decisional Supersingular Isogeny* (*DSSI*) problem and is expected to be computationally intractable. Nevertheless Alice has more information at her disposal: $\phi'_B(U_k)$, $\phi'_B(V_0)$ and the n pairs (U_i, V_i). It is possible now to consider a Weil Pairing's property such that:

$$e(\phi'_B(U_k), \phi'_B(V_k)) = e(U_k, V_k)^{deg\,\phi'_B} = e(U_k, V_k)^{3^m}$$

What would happen if Bob had calculated his n bases with different pairings? Alice would be able to check all bases until she finds the correct one which holds the condition above, as a consequence she would know Bob's k. In order to prevent this scenario, [9] introduces the *XDSSI* problem and, according to the protocol settings, this is completely analogous to the *CSSI* problem. Since the latter is considered a hard problem then consequently the former *XDSSI* is hard too and the presented protocol is secure with respect to Bob's secret k.

Malicious Bob

The security of the random oracle model relies on the hardness of *2-inv-CSSI* problem since if the generators of $ker\,\phi_0$ and $ker\,\phi_1$ can be efficiently computed, then it would be easy to obtain x_0, y_0, x_1, y_1 and solve the problem. However there is no reduction to the *SSCDH* problem since the curves E_0 and E_1 and their associated points are not 3^m-isogenous to E. The difficulty in solving the *2-inv-CSSI* problem actually depends upon the Oblivious Transfer hardness. In fact it would require to find a curve E' and points U', V' such that U' (resp. V') is related to both U_0, U_1 (resp. V_0, V_1) although this is expected to be computationally infeasible even on a quantum computer.
There could be another way for Bob to break the scheme and it requires him to send Alice a pair (U', V') that is not a $E'[2^n]$ basis. In this way Bob is limiting the possible values of $x_iU' + y_iV'$. In order to avoid this situation, Alice should always perform a safety check on the received basis before proceeding further.

Ultimately it is assumed that the combination of the symmetric encryption scheme *Enc* and the *KDF* is IND-CPA (Indistinguishable under Chosen Plaintext Attack), hence the *2-inv-DSSI* problem must be computationally hard. Even though this problem is easier than its computational version *2-inv-CSSI* there is no known reduction to the *SSDDH* problem.

Chapter 4

Implementation details

The SIDH-based OT implementation relies strongly on the functions used in SIDH implementations. For the purposes of this these we have chosen the *Supersingular Isogeny Key Encapsulation* (SIKE) library [11] as a starting point in our implementation. Along with said library, additional functions have been coded in order to reproduce the exact behaviour of SIDH-based OT protocol.

4.1 SIKE Library

This section presents a brief disclosure on the SIKE library [11], a collection of functions and definitions to make the SIDH Key Exchange possible. The library is presented as a collection of different base implementations: one for portable C; one for x64 platforms; two for ARM64 and FPGA optimising different aspects of the platform; finally a simple textbook implementation. Many of these are protected against timing and cache attacks at the software level. Moreover the optimised x64 implementation is further subdivided into *standard* and *compressed* versions, both of them support four different parameters sets: p_{434}, p_{503}, p_{610}, p_{751}.
The latter implementation offers compressed parameters sets and optimised, state-of-the-art, functions and strategies to achieve the highest speed ups possible. Ultimately, our choice is the x64 optimised, compressed, *SIKEp503_ compressed* which also grants us to easily and readily adapt our code to the other *compressed* parameters sets.

4.1.1 x64 Optimised, Compressed implementations

This software version is written in portable C and, in addition to the standard optimised version, provides public key compression and key encapsulation.
The compression is performed for both the static and ephemeral public keys.
The uncompressed public key (resp. ciphertext) sizes corresponding to *SIKEp434* and *SIKEp503* are 330 and 378 (resp. 378 and 402) bytes, which is comparable to the 384-byte (3072-bit) modulus that is conjectured to offer 128 bits of classical security. Likewise, *SIKEp610* public keys (resp. ciphertexts) are 462 (resp. 486) bytes, and the largest of our parameter sets, SIKEp751, has 564-byte uncompressed public keys and 596-byte ciphertexts. On NIST's Challenge Round II there have been a further public key and ciphertext compression; this reduces all of the above numbers to roughly 60% of their former size, for performance overheads ranging from 139% to 161% during public key generation, 66% to 90% during encapsulation, and 59% to 68% during decapsulation. Finally there are reduced public key size by 41% and reduced ciphertext size by 39%.

The implementation offers efficient algorithms for isogeny computations and tree traversing strategies; elliptic curves computations using projective coordinates on Montgomery curves; scalar multiplication via 3-points Montgomery Ladder. Field operations on $\mathbb{F}_{p^2}$ make use of Karatsuba and Lazy Reduction techniques; multiprecision multiplication is implemented with a fully rolled version of Comba while the modular reduction with a Montgomery reduction. As a result, the field arithmetic is generic but very compact. Additionally this implementation is common to all the security levels hence offers great code reuse.
Protocol performances have been evaluated with a benchmark test on a 3.4GHz Intel Core i7-6700 processor running Ubuntu 16.04.3 LTS; TurboBoost disabled, clang 3.8.0 with the command "*clang -03*". Table 4.1 shows the comparison between the two implementations. For a memory analysis, in table 4.2 are reported the sizes in Bytes of secret and public keys, the ciphertext and the shared secret used in SIKE.

Scheme	*Key Generation*	*Encapsulation*	*Decapsulation*
Optimised implementations			
SIKEp434	56.264	92.180	98.335
SIKEp503	86.067	141.891	150.879
SIKEp610	160.401	294.628	296.577
SIKEp751	288.827	468.175	502.983
Compressed implementations			
SIKEp434_compressed	16.542	20.045	18.930
SIKEp503_compressed	23.395	27.543	25.534
SIKEp610_compressed	40.386	47.099	45.449
SIKEp751_compressed	62.347	78.748	72.774

Table 4.1: SIKE performances in thousands of clock cicles (rounded down)

Scheme	*Secret Key*	*Public Key*	*Ciphertext*	*Shared Secret*
SIKEp434	374	330	346	16
SIKEp503	434	378	402	24
SIKEp610	524	462	486	24
SIKEp751	644	564	596	32
SIKEp434_compressed	239	196	209	16
SIKEp503_compressed	280	224	248	24
SIKEp610_compressed	336	273	297	24
SIKEp751_compressed	413	331	363	32

Table 4.2: SIKE inputs and outputs sizes in Bytes

4.1.2 Implementation Choices

The vast majority of parameters used in our SIDH-based OT comes from the SIKE library due to its high performances and optimisation. Therefore, in this section we are going to describe SIKE's choices [7] and so, by reflection, ours.

Smooth Order Supersingular Elliptic Curves

Prime numbers in the form of $p = l_A^{e_A} l_B^{e_B} \pm 1$ have bases $l_A = 2$ and $l_B = 3$ fixed, the exponents are chosen such that the resulting numbers 2^{e_A} and 3^{e_B} have bit lengths slightly smaller than multiples of 64 while aiming to find $2^{e_A} \approx 3^{e_B}$ to ensure that attacking one party is not considerably easier than attacking the other. This also serves to balance the computational costs for each party without disadvantaging either.
This choice supports efficient arithmetic on many platforms and allows a large variety of optimisations. Among all, it is well-known that prime numbers having a special form (like the former just described above) can improve algorithm performances for the underlying modular arithmetic.
As a brilliant side effect, these primes ease construction of supersingular elliptic curves E over $\mathbb{F}_{p^2}$ with the consequence of having smooth order $(l_A^{e_A} l_B^{e_B})^2$. With these settings, we say that given $l \in \{l_A, l_B\}$ and $e \in \{e_A, e_B\}$, the full torsion group on E is defined over $\mathbb{F}_{p^2}$; furthermore l is coprime to p hence $E[l^e] \simeq (\mathbb{Z}/l^e\mathbb{Z})^2$. Now let $(P, Q) \in E[l^e]$ be a basis of order l^e such that we have the isomorphism $(\mathbb{Z}/l^e\mathbb{Z})^2 \to E[l^e]$, with map $(m, n) \to [m]P + [n]Q$. SIDH secret keys are the l^e-isogenies of E which are, in turn, isomorphic to cyclic subgroups of order l^e with kernel $\langle P, Q\rangle$. Finally, a point $[m]P + [n]Q$ has full order l^e if and only if at least m or n are coprime to l: there exist $l^{2e-2}(l^2 - 1)$ such points and $l^{e-1}(l + 1)$ distinct cyclic subgroups of order l^e.

Avoid Inversions

Montgomery projective curves are a combination of two techniques which aim to minimise the number of *inversions*, a high cost operation. First we consider elliptic curves in a projective space, then we exploit the fast arithmetic of Montgomery curves to efficiently compute points in $\mathbb{P}^1$. By coupling these techniques together it has been possible to code more compact algorithms ultimately achieving:

- faster point arithmetic by ignoring the Y projective coordinate;
- faster and more efficient isogeny arithmetic by ignoring the B curve constant and working with the pair $(A : C) \in \mathbb{P}^1$;
- key generation with a 3-way simultaneous inversion to normalise all the components of the key;
- shared secret (j-invariant) computation with only two inversions in the function $j\_inv : (A : C) \to j(E_{(A:B:C)})$.

Projective Isogenies

With this section we want to briefly summarise isogeny computations in SIKE library and also used in our implementation.
The fundamental isogenies are 3- and 4-degree evaluated with an isogeny walk and a traversing strategy. Starting with the 3-isogenies case we define:

- the supersingular elliptic curve $E_{(A:C)}$ expressed in projective Montgomery form;
- the affine x-coordinate $x(P) = (X_3 : Z_3) \in \mathbb{P}^1$ such that the point P has order 3 in $E_{(A:C)}$;
- the supersingular elliptic curve $E'_{(A':C')} = E_{(A:C)}/\langle P\rangle$;
- the 3-isogeny $\phi : E_{(A:C)} \to E'_{(A':C')}$;
- the point $Q \in E_{(A:C)} \setminus ker(\phi)$, its affine x-coordinate $x(Q) = (X : Z) \in \mathbb{P}^1$ and its image $x(\phi(Q)) = (X' : Z') \in \mathbb{P}^1$

At this point we have two functions at our disposal: *get_3_isog* which computes the isogenous curve $E'_{(A':C')}$ only and *eval_3_isog* to evaluate a single point in the image curve. The first function takes the curve $E_{(A:C)}$ and the point P's projective coordinates $(X_3 : Z_3)$ as input; the output is the 3-isogenous curve $E'_{(A':C')}$. This curve is computed with the expression:

$$(A' : C') = \big((AX_3Z_3 + 6(Z_3^2 - X_3^2))X_3 : CZ_3^3\big)$$

which is independent from $E_{(A:C)}$ coefficients and it requires 3 multiplications, 3 squaring, 8 additions. Using a well-known notation we could also write 3**M**+3**S**+8**a** where **M** stands for Multiplication, **S** for Squaring, **a** for Addition, eventually **I** for Inversion.
To evaluate the point Q we use:

$$(X' : Z') = \big(X(X_3X - Z_3Z)^2 : Z(Z_3X - X_3z)^2\big)$$

requiring 6**M**+2**S**+2**a**.

Passing on 4-isogenies we can reuse the above notations and only change the point P as a point of order 4 in $E_{(A:C)}$ with affine x-coordinate $x(P) = (X_4 : Z_4) \in \mathbb{P}^1$. In the isogeny walk used to compute a 4^e-isogeny an isomorphism is needed for every 4-isogeny step except for the first one. The following formulae consider a curve $E_{(A:C)}$ normalised so that $A = a$, $C = 1$.
For the first 4-isogeny, the image curve $E'_{(A':C')}$ is computed as:

$$(A' : C') = \big(2(a + 6) : a - 2\big)$$

while the image of a point Q is:

$$(X' : Z') = \big((X + z)^2(aXZ + X^2 + Z^2) : (2 - a)XZ(X - Z)^2\big)$$

requiring 4**M**+2**S**+9**a**.

For all the other 4-isogenies we get:

$$(A' : C') = \big(2(2X_4^4 - Z_4^4) : Z_4^4\big)$$

while the image of a point Q is:

$$\begin{aligned}(X' : Z') = \Big(&X\big(2X_4Z_4Z - X(X_4^2 + Z_4^2)\big)(X_4X - Z_4Z) \ : \\ &Z\big(2X_4Z_4X - Z(X_4^2 + Z_4^2)\big)(Z_4X - X_4Z)\Big)\end{aligned}$$

Bases and Torsion Points

Throughout the protocol, Alice and Bob make use of two particular bases: Alice's basis is (P_A, Q_A) of order 2^n and Bob's (P_B, Q_B) of order 3^m. To better explain how these bases had been computed we are going to follow the example in [7] thus we set $n = 372$, $m = 239$.
Let $P_A \in E_0(\mathbb{F}_p[2^{372}])$ be a point described as $[3^{239}](z, \sqrt{z^3 + z})$ where z is the smallest positive integer such that $\sqrt{z^3 + z} \in \mathbb{F}_p$ and P_A is of order 2^{372}. To compute the second point Q_A we use a distortion map $\tau : E_0(\mathbb{F}_p[2^{372}]) \to E_0(\mathbb{F}_p[2^{372}])$, $(x, y) \to (-x, iy)$ hence $Q_A = \tau(P_A)$. Bob's basis is found similarly.

Having now two distinct bases, one may exploit precise linear combination of basis points to sample a full order torsion point.
To sample a 2^{372}-order point $R_A \in E_0(\mathbb{F}_p[2^{372}])$, it is possible to choose a random integer $m' \in \{1, 2, \dots, 2^{371} - 1\}$ and set $R_A = P_A + [2m']Q_A$.
A similar approach is used to sample 3^m-order points: choose a random integer $m' \in \{1, 2, \dots, 3^{238} - 1\}$ and set $R_B = P_B + [3m']Q_B$.

Bibliography

[1] *Post Quantum Cryptography: Implementing Alternative Public Key Schemes on Embedded Devices*, by Stefan Heyse, October 2013.

[2] *Mathematical and Provable Security Aspects of Post-Quantum Cryptography*, by Alan Szepieniec, December 2018

[3] *Cybersecurity in an era with quantum computers: will we be ready?*, by Mosca M., Cryptology ePrint Archive, Report 2015/1075, 2015.

[4] *On the Development and Standardisation of Post-Quantum Cryptography. A Synopsis of the NIST Post-Quantum Cryptography Standardisation Process, its Incentives, and Submissions*, by Maja Worren Legernæs, Norwegian University of Science and Technology, Master of Science in Communication Technology, June 2018.

[5] *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, by Peter W. Shor, `https://arxiv.org/pdf/quant-ph/9508027.pdf`.

[6] *Mathematics of Isogeny Based Cryptography*, by Luca De Feo, École mathématique africaine, May 2017, Thiès, Senegal.

[7] *Efficient algorithms for supersingular isogeny Diffie-Hellman*, by Craig Costello, Patrick Longa, and Michael Naehrig, Microsoft Research, USA, 2016.

[8] *Montgomery curves and their arithmetic, The case of large characteristic fields*, by Craig Costello, Benjamin Smith, 2017.

[9] *Simple oblivious transfer protocols compatible with Kummer and supersingular isogenies*, by Vanessa Vitse, January 2019.

[10] *Isogeny based crypto: what's under the hood?*, by Luca De Feo, November 2018.

[11] *Supersingular Isogeny Key Encapsulation*, by David Jao, April 2019. SIKE project link: `https://sike.org/`. Library documentation link: `https://sike.org/files/SIDH-spec.pdf`.

[12] *TOWARDS QUANTUM-RESISTANT CRYPTOSYSTEMS FROM SUPERSINGULAR ELLIPTIC CURVE ISOGENIES*, by Luca De Feo, David Jao, Jérome Plut, 2014.

[13] *Cryptographic primitives from elliptic curve isogenies*, by Federico Pintore, 2019.

[14] *Faster Algorithms for Isogeny Problems using Torsion Point Images*, by Christophe Petit, 2017.

[15] *ON THE SECURITY OF SUPERSINGULAR ISOGENY CRYPTOSYSTEMS*, by Steven D. Galbraith, Christophe Petit, Barak Shani, Yan Bo Ti, 2016.

[16] *A Note on Post-Quantum Authenticated Key Exchange from Supersingular Isogenies*, by Patrick Longa, 2018.

[17] *Selected Areas in Cryptography – SAC 2018*, by Carlos Cid, Michael J. Jacobson Jr., 2019.

[18] *The Design of Rijndael*, by Joan Daemen, Vincent Rijmen, 2002.

[19] *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, by R.L. Rivest, A. Shamir, and L. Adleman, 1977.

[20] *An Elliptic Curve Cryptography (ECC) Primer*, by Certicom, June 2004.

[21] *Quantum theory, the Church–Turing principle and the universal quantum computer*, Deutsch D, 1985.

[22] Aleksandar Jurisic and Alfred J. Menezes, (2004). Elliptic Curves and Cryptography. *Guide to Elliptic Curve Cryptography*, 173-193.

[23] Joseph H. Silverman, (2009). *The Arithmetic of Elliptic Curves*

[24] Joseph H. Silverman, (2006). *An Introduction to the Theory of Elliptic Curves.*

[25] Christophe Ritzenthaler, (2014). *Introduction to elliptic curves.*

[26] Luca De Feo, (2019). *Isogeny Graphs in Cryptography.*

[27] Craig Costello and Huseyin Hisil, (2017). *A simple and compact algorithm for SIDH with arbitrary degree isogenies*

[28]