Contents

Chapter 1

Introduction

The word "*cryptography*" derives from the Greek word κρυπτὸς, *kryptos*, meaning hidden. Its origin is usually dated back to 2000 B.C., and associated with the Egyptian practice of hieroglyphics which full meaning was only known to an elite few. Formally, cryptography is the science of protecting information exchanged by two or more parties. In the last decades, it has advanced exponentially by taking advantage of the most developed formalisations and by relying on hard mathematical problems for security. 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 mid 19-th century, Modern Cryptography is a wider field than classical cryptography since it also offers new additional properties vital to modern communications that can be summed up as follows:

- 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. This is the only property attempted to achieve by classical cryptography;
- Integrity - ensures the information sent is exactly the same information re-

ceived;

- 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), designed in 1997 to supersede its predecessor DES in terms of both efficiency and security. More on this protocol can be found in [18].
Public key encryption schemes use a pair of keys: the public key serves to encrypt messages, the private one to decrypt them. These special algorithms are also named 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: RSA [19] is the most world spread algorithm [29], while Elliptic Curve Cryptography (ECC) [20] provides the most recent and efficient schemes.

In parallel with cryptography's evolution, 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 quantum Turing machine, or universal quantum computer: it can simulate any quantum mechanical process with small overhead and independently of the substrate.
The basic idea behind a universal quantum computer is to exploit quantum bits, or qubits for short. As opposed to binary bits, qubits can achieve additional states in between the two binary states. This is defined as a superposition of the digital states. A full treatment about quantum computers is beyond the singular of this thesis but a curious reader can refer to [1, 2, 4].
It is natural wondering about whether quantum computers could solve classical com-

putational 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 dramatic: large enough quantum computers will break factorisation-based cryptosystems, such as RSA, as well as cryptosystems based on the discrete logarithm problem, such as elliptic curve cryptosystems.

1.1 Shor's algorithm

Theorised in 1994 by Peter Williston Shor, his algorithm is one of the most important in 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 represented as sinus waves to factorise big integers. Using number theory, the problem of integer 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.

1.2 Post-Quantum Cryptography

With the term "*Post-quantum cryptography*" we refer to the cryptography's branch aiming at protecting information from both quantum and classical attacks, as well as to the collection of schemes that accomplish this task. Unfortunately, the transition to post-quantum cryptography is not cost-free. The hard problems exploited by post-quantum cryptography, except for hash inversion, have been studied less than integer factorisation and the discrete logarithm problem. Consequently a post-quantum hard problem inevitably conveys a lighter 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 from quantum computers require cryptosystems with far greater memory and bandwidth impeding their adoption for low-cost devices.

1.2.1 NIST's Standardisation Competition

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.
The above mentioned effort was also triggered by "*The Post-Quantum-Cryptography Standardization Challenge*" which is a competition started by the National Institute of Standards and Technology (NIST) in April 2016 accepting proposals for post-quantum protocols. It entered in the 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 five different levels, summarised below:

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

As a final note, most published post-quantum schemes can be divided into the following families [1]:

- Hash-based cryptography (e.g. Merkle's hash-tree signature system [30]);
- Multivariate cryptography (e.g. HFE signature scheme [31]);
- Lattice-based cryptography (e.g. NTRU encryption scheme [34]);
- Code-based cryptography (e.g. McEliece encryption scheme [33], Niederreiter encryption scheme [32]).

A new family, called isogeny-based cryptography, recently originated and it is the central point of this thesis.

1.3 This Thesis

In this thesis we want to present a C implementation of a post-quantum protocol introduced in [9] by Vanessa Vitse. The proposed algorithm relies on the supersin-

gular isogeny problem, which is quantum resistant, for its security, it exploits a zero knowledge protocol to exchange some data between the parties without revealing unnecessary information.
By implementing this protocol first we reiterate that classical computers are central in post-quantum cryptography or, viceversa, quantum computers are not compulsory when implementing post-quantum algorithms. Most importantly we adapted an open source library and to produce the first public implementation, as far as our knowledge goes, of the above mentioned algorithm. This thesis is organised as follows.

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

Chapter 2

Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is an alternative to RSA-based public key scheme based on the mathematics of elliptic curves. Cryptographic algorithms within 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 fall in the area of encryption via public key but it can also be used in 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 on elliptic curves to familiarise with them.

2.1 Elliptic Curves

An elliptic curve [22] is a cubic curve defined over a field $\mathbb{K}$ and having a specified point $\mathcal{O}$. Whenever the field $\mathbb{K}$ has characteristic $char(\mathbb{K}) \neq \{2, 3\}$, an elliptic curve can be defined as the locus of points satisfying an affine equation which comes with a smoothness condition:

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

together with the point at infinity $\mathcal{O}$ as expressed in [23].
Points of an elliptic curve E form a group $(E, +)$ [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 now briefly list some of the most important properties for an elliptic curve from a cryptographic 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:

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

- The above formula cannot be used for the Point Doubling, i.e. for summing P, Q when $P = Q$. In that case 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 $R = P + Q = Q + P$;

- The operation of adding a point P to itself k times is written as $[k]P$, or simply kP, and sometimes called scalar multiplication (between the scalar k and the point P;

- Given a point P of an elliptic curve, its order is the minimum number k such that $[k]P = \mathcal{O}$. All the points $\{G, [2]G, [3]G, \dots, [k-1]G, [k]G\}$ form an elliptic curve E denoted as $E : \langle G \rangle$ and has cardinality k;

- An elliptic curve, defined over a finite field $\mathbb{F}_p$ with p prime greater than 3, having $p + 1$ points with coefficients in $\mathbb{F}_p$, is said *Supersingular*, otherwise is said Ordinary. The set of all the points of the elliptic curve E having coefficients in $\mathbb{F}_p$ is denoted by $E(\mathbb{F}_p)$ and it is called the set of rational points. $E(\mathbb{F}_p)$ is a subgroup of $(E, +)$;

- 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 is an important identifier for an elliptic curve and, for this reason, it 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 computers 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 $P = [k]G$ if it exists.

In figure 2.1 we can see two common affine representations of an elliptic curve defined over the field $\mathbb{R}$: on the left $a = -3$, $b = 1$, on the right $a = -2$, $b = 2$.

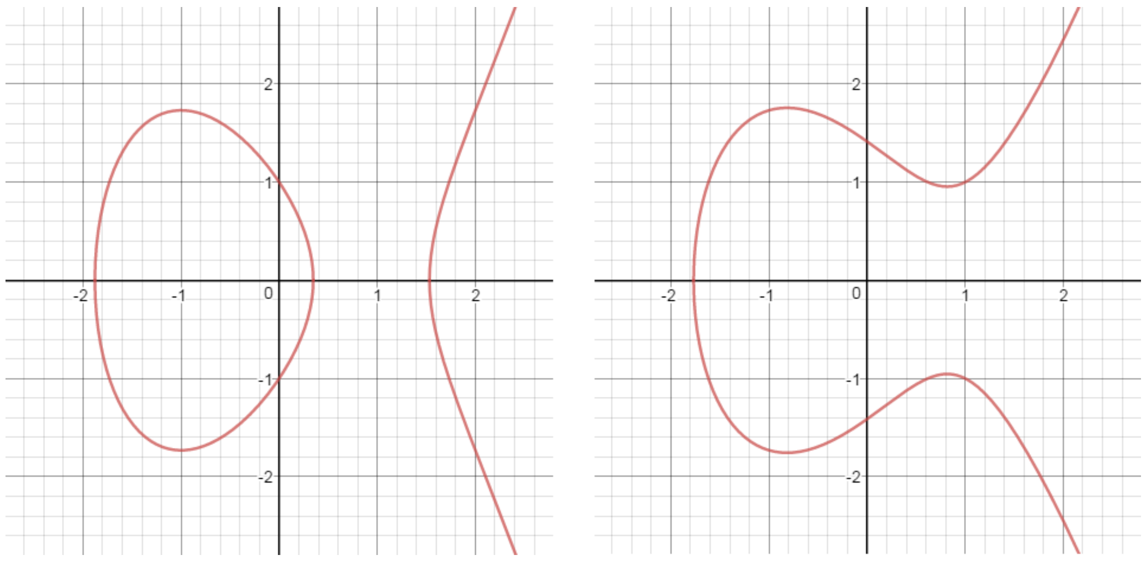

Figure 2.1: Two affine elliptic curves

2.2 Important concepts

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

2.2.1 m-torsion groups

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:

$$\{P \in E \mid [m]P = \mathcal{O}\}$$

This set forms a subgroup for which holds:

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

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 a consequence of the 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 a group homomorphism if it 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*, we write $G \simeq H$ and the two groups have the same algebraic structure.
Since the points of an elliptic curve form a group [25], it is natural to extend the concepts of homomorphism and isomorphism to elliptic curves. In particular, it turns out that two elliptic curves over $\mathbb{K}$ are isomorphic over the 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 which is expressed by rational functions. Moreover, given two elliptic curves E, E' and the map $\phi : E \to E'$, the following are equivalent when the isogeny ϕ satisfies an extra property, named separability, ϕ is a group homomorphism with finite kernel. 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: two elliptic curves are isogenous over an extension of $\mathbb{F}$ or $\mathbb{K}$ if and only if $\#E(\mathbb{F}) = \#E'(\mathbb{F})$, where $E(\mathbb{F})$ denotes the subgroup of points with coefficients in $\mathbb{F}$.

Isogenies can be computed via Velù's formulae: let $E : y^2 = x^3 + ax + b$ be an elliptic curve over a finite field $\mathbb{F}_q$, P a generic point of E, $\langle K \rangle$ a cyclic group generated by K; let $\phi : E \to E'$ be the isogeny with kernel $\langle K \rangle$. The arriving curve E' is usually denoted as $E/\langle K \rangle$. The isogeny $\phi(P) = (x_\phi, y_\phi)$ has equation [6]:

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

$$y_\phi = \left(y(P) + \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(y(P+Q) - y(Q) \Big) \right) \tag{2.3}$$

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

$$a' = a - 5 \cdot \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(3x(Q)^2 + a \Big) \tag{2.4}$$

$$b' = b - 7 \cdot \sum_{Q \in \langle K \rangle \setminus \{\mathcal{O}\}} \Big(5x(Q)^3 + 3ax(Q) + b \Big) \tag{2.5}$$

The operations in 2.2, 2.3, 2.4 and 2.5 require a small constant number of computations 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, defined over $\mathbb{F}_{11}$ by $y^2 = x^3 + x$ is mapped on the right curve E', defined over $\mathbb{F}_{11}$ by

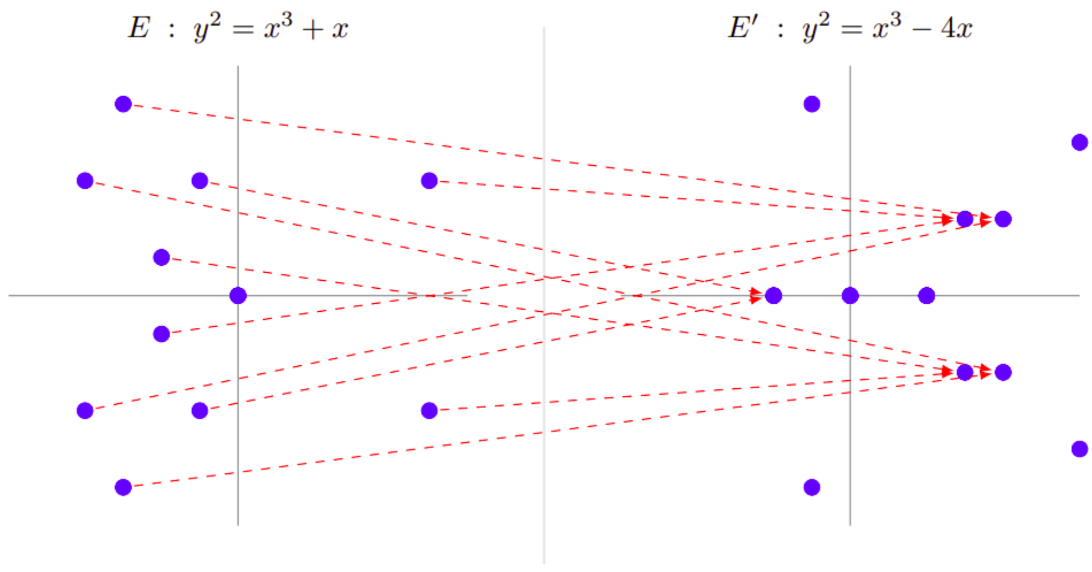

Figure 2.2: Isogeny mapping between E and E' [10]

$y^2 = x^3 - 4x$. The kernel is the point $(0, 0)$ on E and the isogeny is defined by:

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

Degree

Following [6,26], we define the degree of a separable isogeny $\phi : E \to E'$ as the finite cardinality of its kernel. It is common in literature to call an isogeny ϕ of degree $deg(\phi) = d$ as d-isogeny and so we will do hereafter.
Let E be an elliptic curve and G a finite subgroup of E. Then there is a *unique* elliptic curve E' and a unique separable isogeny ϕ such that $ker\ \phi = G$ and $\phi : E \to E'$. Consequently, the notation $E' = E\big/G$ uniquely denotes the range of ϕ.

Dual Isogenies

For any isogeny $\phi : E \to E'$ between two 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 to note that $deg\,\widehat{\phi} = deg\,\phi$, and $\widehat{\widehat{\phi}} = \phi$.

2.2.4 Endomorphisms

Isogenies from a curve to itself are called *endomorphisms*. Given an elliptic curve 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 finite field $\mathbb{F}_q$ with q elements Its Frobenius endomorphism is the map $\pi : P = (x, y) \to (x^q, y^q)$.
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 field $\mathbb{F}_q$, with $q = p^n$, and an integer $m \neq p$, we know that $E[m] = \mathbb{Z}_m \times \mathbb{Z}_m$. A basis for the $E[m]$ is a pair $(P, Q) \in E[m]$ of linearly independent points. Indeed, the set of all their linear combinations is equal to $E[m]$. In that case we write $\langle P, Q\rangle = E[m]$. To be more precise one can say that two points P, Q of an elliptic curve are linearly independent if and only if $[v]P + [u]Q = \mathcal{O}$ if and only if $v = u = 0$.
It is possible to check the linear independency of two points using 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 $\bar{\mathbb{K}}$, which 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 a bilinear map $e_n : E[n] \times E[n] \to \mu_n$.

A bilinear map is a function that combines elements of two vector spaces, $E[n]$ in our case, returns an element of a third vector space, $E[n] \times E[n]$, and is linear in both its arguments.
For a basis (P, Q) of $E[n]$ the Weil pairing $e_n(P, Q)$ is a primitive n-th root of unity. Consequently, to check if (P, Q) is a basis for $E[m]$ is sufficient to check the order of $e_m(P, Q)$, in particular if it is n.

2.3 Mathematical improvements

This section aims to sum up some of the many improvements to the concepts seen in section 2.2 commonly 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:

$$M_{(A,B)} : By^2 = x(x^2 + Ax + 1) \tag{2.6}$$

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:

$$M_{(A,B)} : BY^2Z = X(X^2 + AXZ + Z) \tag{2.7}$$

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

Many computational improvements come from choosing a Montgomery curve over a general elliptic curve given by its Weierstrass equation. 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 a 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.

Several variants of the Montgomery curve constants are used when implementing elliptic curves cryptographic schemes [11] to achieve better performances. Let E_a be the curve $E - A/\mathbb{F}_{p^2} : y^2 = x^3 + ax^2 + x$, we write the pair $(A : C)$ to denote the equivalence $(A : C) \sim (a : 1)$ in $\mathbb{P}^1(\mathbb{F}_{p^2}$. Furthermore, we define:

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

2.3.2 Weil Pairing

The base algorithm for computing the Weil pairing e_n 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 computation as described in [7].
Let P, Q be two points both of order mn of an elliptic curve E, so that they are in $E[mn]$; the n-th power of the pairing $e_{mn}(P, Q)$ can be computed as follows [7,25]:

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

We now consider an example: let $m = 4$, $n = 2^{370}$ such that $mn = 2^{372}$, then:

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

Together with the assertion that P and Q both have order $mn = 2^{372}$, the assertion that the Weil pairing $e_4([2^{370}]P, [2^{370}]Q)$ is non-trivial, proves that P and Q have Weil pairing of order 2^{372}. If indeed P and Q have order 2^{372}, the points $P' = [2^{370}]P$ and $Q' = [2^{370}]Q$ both have order 4. In that case, $e_4(P', Q') \neq 1$ if and only if $P' \neq Q'$.
To prove this last step we can exploit another speed up: all the described operations are possible considering only the x coordinates of input points P, Q. 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 such that $x(P) = \frac{X(P)}{Z(P)}$.
At this point it is mandatory to check $P' \neq Q'$ that is $x(P') \neq x(Q')$. 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 obtain $X(P')Z(Q') \neq X(Q')Z(P')$ 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} and are linear independent. 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 subgroup of points of an elliptic curve E, the worst drawback of using Velù's formulae for computing an isogeny from E having $\langle K \rangle$ as kernel, is that they require $\mathcal{O}(\# \langle K \rangle)$ operations hence making their implementation intractable when $\# \langle K \rangle)$ is big. 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 fundamental for practical implementations.

Costello et al. [7] write about efficiently computing large degree isogenies via small degree isogenies. We now describe the idea of composing isogenies.
Given an elliptic curve E defined over $\mathbb{F}_{p^2}$, a cyclic subgroup $\langle R \rangle \subseteq E[l^e] \subseteq E\mathbb{F}_{p^2}$ 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], obtaining intermediate curves $E_{i+1} = E_i/\langle [l^{e-i-1}]R_i\rangle$ for $0 \leq i < e$. Note that the point R_i is an l^{e-i}-torsion point and so the point $[l^{e-i-1}]R_i$ has order l. The resulting isogeny is then $\phi_R = \phi_{e-1} \circ \ldots \circ \phi_0$ having degree l^e as required. Since the composition of two separable isogenies is a separable isogeny, the degree of $\phi \circ \varphi$ is $deg(\phi)\, deg(\varphi)$.
Even though we may compute the curve E_{i+1} and its isogeny ϕ_i via Velù's formulae, this strategy is still too onerous. A better strategy is a particular isogeny walk, as shown in figure 2.3 for the case $e = 6$ [12]. Bullets "•" 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.

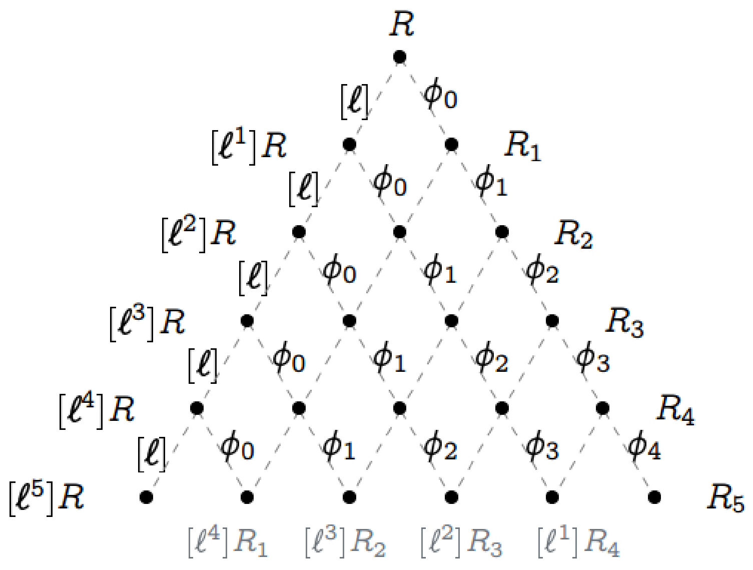

Figure 2.3: l-isogeny walks up to degree l^5

At this point we need a "*strategy*" [7,12] to minimise the operations in descending the graph in figure 2.3 down to the bottom leaves. We may define a strategy as a binary tree topology: a tree in which each node has only one parent node (directed from above in oriented trees with the root on top) and two or zero child node (directed to the bottom direction). Each of its nodes can be decomposed into two sub-trees, or 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 from the root on top along the edges to the roots of the sub-strategies. Given the costs of all sub-strategies, one can select the a strategy by choosing the one with minimum cost hence named *optimal*.

A strategy can be stored in a simple list L of integers, with L having 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 graph there are seven optimal formed strategies as shown in figure 2.4 [12].

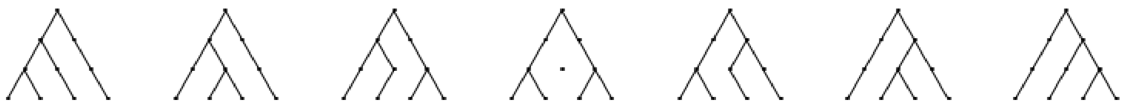

Figure 2.4: Optimal strategies for $e = 4$

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

Small degree isogenies

In this section we show the formulae alternative to that of Velù's used in real applications. There, only two different isogenies are usually used: 2- and 3-isogenies. In the former case, a further speed up is possible if we consider 4-isogenies which halves the total operations needed [11].

2-isogenies

Let $E_{A,B}$ be an elliptic curve, $(x_2, y_2) \in E_{A,B}$ 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 originating from $E_{A,B}$ with kernel $\langle (x_2, y_2) \rangle$. The coefficients of the image curve can be computed as follows:

$$\left(A', B'\right) = \left(2 \cdot \left(1 - 2x_2^2\right), \quad Bx_2\right)$$

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

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

4-isogenies

Let $E_{A,B}$ be an elliptic curve, $(x_4, y_4) \in E_{A,B}$ 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 originating from $E_{A,B}$ with kernel $\langle (x_4, y_4) \rangle$. The coefficients of the image curve can be computed as follows:

$$\left(A', B'\right) = \left(4x_4^4 - 2, \quad -x_4(x_4^2 + 1) \cdot B/2\right)$$

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

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

3-isogenies

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

$$\left(A', B'\right) = \left(\left(Ax_3 - 6x_3^2 + 6\right) x_3, \quad Bx_3^2\right)$$

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:

$$x_{\phi_3} = \frac{x_P (x_P x_3 - 1)^2}{(x_P - x_3)^2}$$

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

2.4 Isogeny-based Cryptography

Traditional Elliptic Curve Cryptography (ECC) relies for its security on the Discrete Logarithm Problem (DLP). That said, Shor's algorithm represents a threat to DLP-based cryptography since it can compute discrete logarithms in polynomial time. A quantum-resistant alternative 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 $\phi : E \to E'$*". For the given problem does not exist any efficient quantum attack thus making it post-quantum resistant.

Finally 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 also more computationally efficient.

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 coupling these two protocols together it is possible to construct a post-quantum multi-party scheme.
First things first, we separately explain how SIDH [7,11,12] and OT 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 [11] 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 [7].
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 in deeper details.

All the supersingular elliptic curves used in SIDH are defined over $\mathbb{F}_{p^2}$ where p is a prime of the form $p = l_A^{e_A} l_B^{e_B} \pm 1$, l_A and l_B are small primes, e_A and e_B are

integers such that $l_A^{e_A} \approx l_B^{e_B}$. All SIDH implementations consider $l_A = 2$, $l_B = 3$ and so we will henceforth. Typical choices for e_A fall within the range $[100, 500]$ according to the security level desired. In order to simplify the notation we will be using n instead of e_A and m instead of e_B.
The initial public parameters are:

- A supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$;
- A bases (U, V) for $E[2^n]$ which lies in $E(\mathbb{F}_{p^2})$;
- A bases (P, Q) for $E[3^m]$ which lies in $E(\mathbb{F}_{p^2})$.

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 $R_A = x_A U + y_A V$	Computes $R_B = x_B P + y_B Q$
Computes the curve $E_A = E/\langle R_A \rangle$	Computes the curve $E_B = E/\langle R_B \rangle$
Computes the isogeny $\phi_A : E \to E_A$ having $\langle R_A \rangle$ as kernel	Computes the isogeny $\phi_B : E \to E_B$ having $\langle R_B \rangle$ as kernel
Computes $P' = \phi_A(P)$, $Q' = \phi_A(Q)$	Computes $U' = \phi_B(U)$, $V' = \phi_B(V)$

$$\xleftrightarrow{\text{Exchange curves } E_A,\ E_B \text{ and points } U',\ V',\ P',\ Q'}$$

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

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

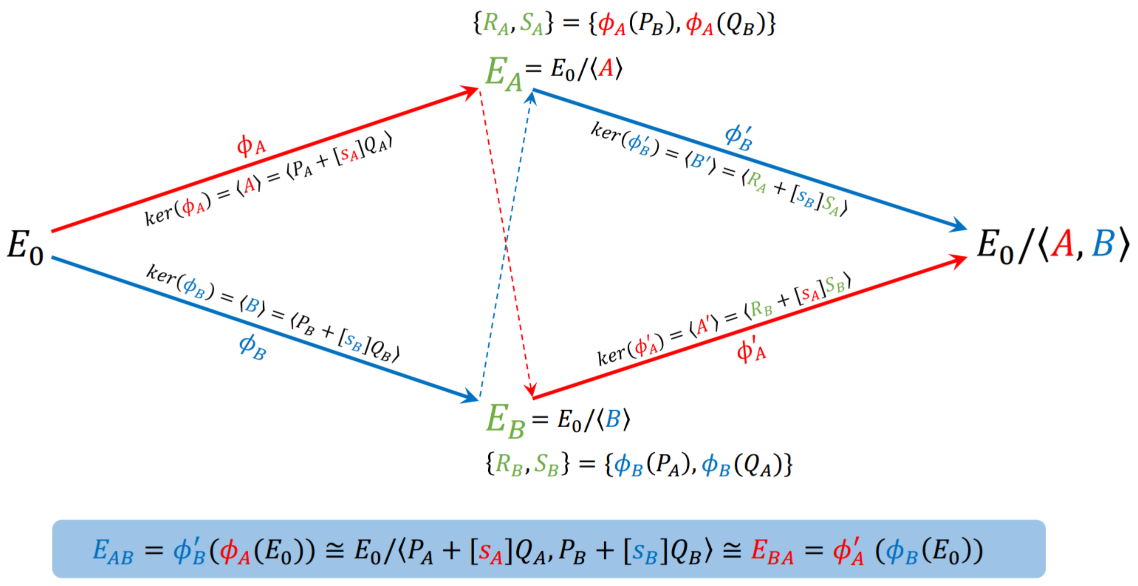
$\{R_A, S_A\} = \{\phi_A(P_B), \phi_A(Q_B)\}$
$E_A = E_0/\langle A\rangle$
ϕ_A
$ker(\phi_A) = \langle A\rangle = \langle P_A + [s_A]Q_A\rangle$
ϕ'_B
$ker(\phi'_B) = \langle B'\rangle = \langle R_A + [s_B]S_A\rangle$
E_0
$E_0/\langle A, B\rangle$
$ker(\phi_B) = \langle B\rangle = \langle P_B + [s_B]Q_B\rangle$
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

This brief section shows how Alice and Bob compute two curves isomorphic to one another thus yielding the same j-invariant.

Alice computes:

$$\begin{aligned}
E_{BA} &= E_B/\left\langle x_A U' + y_A V'\right\rangle \\
&= E_B/\left\langle x_A \phi_B(U) + y_A \phi_B(V)\right\rangle \\
&= E_B/\left\langle \phi_B(x_A U + y_A V)\right\rangle \\
&= E_B/\left\langle \phi_B(R_A)\right\rangle \\
&= \left(E/\left\langle R_B\right\rangle\right)/\left\langle \phi_B(R_A)\right\rangle \\
&= E/\left\langle R_B, R_A\right\rangle
\end{aligned}$$

Bob computes:

$$\begin{aligned}
E_{AB} &= E_A / \langle x_B P' + y_B Q' \rangle \\
&= E_A / \langle x_B \phi_A(P) + y_B \phi_A(Q) \rangle \\
&= E_A / \langle \phi_A(x_B P + y_B Q) \rangle \\
&= E_A / \langle \phi_A(R_B) \rangle \\
&= \big(E / \langle R_A \rangle\big) / \langle \phi_A(R_B) \rangle \\
&= E / \langle R_A, R_B \rangle
\end{aligned}$$

Therefore $E_{BA} = E/\langle R_B, R_A\rangle = E/\langle R_A, R_B\rangle = E_{AB}$ proving that Alice and Bob now share a curve isomorphic to their party's curve.

3.1.2 Security

Before introducing the problem which SIDH's security relies on 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 $\phi_A : E \to E_A$ if it exists".
The problem exploited for SIDH's security is similar to the *Standard Isogeny* problem but it adds some information to the attacker. One can state the problem as follows [13]: "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 $\phi_A : E \to E_A$ if it exists **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 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 [7].

SIDH's primes p were initially selected considering only the running time of the Claw Finding attack without considering its space complexity amounting to $\mathcal{O}(\sqrt[6]{p})$ qubits.
A helpful notation for the following disclosure is to express an n-bit length prime number p is p_n.
As an example [17], in NIST's Competition's Round 1 introduced in 1.2.1, 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 $p_{751} = 2^{372}3^{239} - 1$ was proposed for the 128-bit classical security level and the Quantum Level I. Moreover the 964-bit prime $p_{964} = 2^{486}3^{301} - 1$ was proposed for the 160-bit classical security level.
For NIST's Competition 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.
More details about attacks and examples can be found in [14–16].

3.2 Oblivious Transfers

An *oblivious transfer*, or **OT**, is a multi-party cryptographic scheme in which two or more parties are involved. A typical OT application is the secure function evaluation where every party holds a part of 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 piece of information. Correctness of the protocol is usually proved with a zero knowledge proof.
There exist different instantiations of the oblivious transfer primitive 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 instantiation is the *Rabin OT* [35] 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 [28] 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 B, receives one out of two (alternatively n) pieces of information but the sender party A, which owns both pieces, 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 ran-

dom bit c. At the end of the transfer B will have knowledge of the piece b_c coming from A. It is fundamental to observe that b_0 and b_1 are masked, and that B is only able to recover one of them, i.e. b_c.

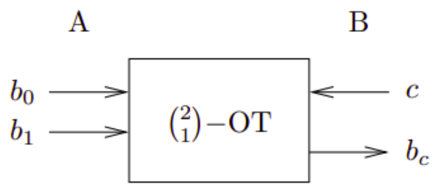

Figure 3.2: Simple $\binom{2}{1}$-OT

3.3 SIDH-based OT

In this chapter we introduce the SIDH-based oblivious transfer protocol proposed by Vanessa Vitse in [9]. The goal here is to construct isogeny-based, post-quantum oblivious transfer protocols that are also analogue to SIDH settings (section 3.1). The initial public parameters are:

- A supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$ with p prime of the form $2^n3^m \pm 1$;
- A bases (P, Q) of $E[3^m]$ with P, $Q \in E(\mathbb{F}_{p^2})$;
- A secure symmetric encryption protocol *Enc* for which we use the notation: $c = Enc(m, k)$, $m = Enc^{-1}(c, k)$, where m is the plain text message, k is the encryption key, c is the cipher text;
- A key derivation function *KDF* such that k = *KDF(seed)* for a given initial input *seed*.

For simplicity we consider 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 for $E[2^n]$:
$(R_0, T_0), (R_1, T_1)$

Computes *two* curves and *two* isogenies:
$E_{A,0} \simeq E\big/\left\langle R_0 \right\rangle,\ \phi_{A,0} : E \to E_{A,0}$
$E_{A,1} \simeq E\big/\left\langle R_1 \right\rangle,\ \phi_{A,1} : E \to E_{A,1}$

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 $b \in \mathbb{Z}\big/3^m\mathbb{Z}$

Computes the curve and j-invariant:
$E_B = E\big/\left\langle P + bQ \right\rangle,\ j_B = j(E_B)$

Computes *two* bases for $E[2^n]$:
$(U_0,\, V_0), (U_1,\, V_1)$
having the same Weil pairing:
$e_{2^n}(U_0,\, V_0) = e_{2^n}(U_1,\, V_1)$

Computes the curve and isogeny:
$E'_B \simeq E_{A,k}\big/\left\langle P_k + bQ_k \right\rangle,\ \phi'_B = E_{A,k} \to E'_B$

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 $x_i,\, y_i \in \mathbb{Z}$, with $i = 0, 1$:
$(x_0,\, y_0)$ *s.t.* $\phi_{A,0}(T_0) = x_0 U_0 + y_0 V_0$
$(x_1,\, y_1)$ *s.t.* $\phi_{A,1}(T_1) = x_1 U_1 + y_1 V_1$

Computes *two* curves and their correspondent j-invariants:
$F_0 = E'_B / \langle x_0 U'_k + y_0 V'_k \rangle$, $j_0 = j(F_0)$
$F_1 = E'_B / \langle x_1 U'_k + y_1 V'_k \rangle$, $j_1 = j(F_1)$

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

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

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

The protocol described above can be generalised to an $\binom{n}{1}$-OT with little adjustments: every "*two*" in the protocol should be changed to "n"; Alice's pairs $x_i,\, y_i \in \mathbb{Z}$, would have $i = 0, \dots, n-1$; Bob's k should be chosen in the finite set $\{0, n-1\}$.

3.3.1 Correctness

How can Bob 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_k must be isomorphic to the curve E_B computed by Bob. We now show the correctness of the protocol. For the sake of simplicity we assume $k = 0$ without loss of generality.

$$\begin{aligned} F_0 &= E'_B / \langle x_0 U'_0 + y_0 V'_0 \rangle \\ &= E'_B / \langle x_0 \phi'_B(U_0) + y_0 \phi'_B(V_0) \rangle \\ &= E'_B / \langle \phi'_B(x_0 U_0 + y_0 V_0) \rangle \\ &= E'_B / \langle \phi'_B(\phi_{A,0}(T_0)) \rangle \\ &= \left(E_{A,0} / \langle P_0 + bQ_0 \rangle \right) / \langle \phi'_B(\phi_{A,0}(T_0)) \rangle \\ &= \left(E_{A,0} / \langle P_0 + bQ_0 \rangle \right) / \langle \phi_{A,0}(T_0) \rangle \\ &= E_{A,0} / \langle P_0 + bQ_0, \phi_{A,0}(T_0) \rangle \end{aligned}$$

$$
\begin{aligned}
&= E_{A,0}/\left\langle \phi_{A,0}(P) + b\phi_{A,0}(Q), \phi_{A,0}(T_0)\right\rangle \\
&= E_{A,0}/\left\langle \phi_{A,0}(P + bQ), \phi_{A,0}(T_0)\right\rangle \\
&= \big(E/\left\langle R_0\right\rangle\big)/\left\langle \phi_{A,0}(P + bQ), \phi_{A,0}(T_0)\right\rangle \\
&= \big(E/\left\langle R_0\right\rangle\big)/\left\langle P + bQ, T_0\right\rangle \\
&= E/\left\langle R_0, P + bQ, T_0\right\rangle \\
&\simeq E/\left\langle R_0, T_0\right\rangle/\left\langle P + bQ\right\rangle \\
&= E/E[2^n]/\left\langle P + bQ\right\rangle \\
&= E/\left\langle P_0 + bQ_0\right\rangle \\
&= E_B
\end{aligned}
$$

3.3.2 Security

For this section it is important to first introduce few hard problems in order to analyse the security of the protocol introduced in the previous section. We will follow [9,12].

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 (U, V) is a basis for $E[2^n]$ and (U', V') is a basis for $E'[2^n]$, 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 $\phi(V) = V'$.

DSSI - *Decisional Supersingular Isogeny* problem

Given the prime p of the form $l_A^{e_A} l_B^{e_B} \pm 1$ where l_A and l_B are small primes, e_A and e_B are integers, the supersingular curves E_0 and E_A defined over $\mathbb{F}_{p^2}$, decide whether E_A is $l_A^{e_A}$-isogenous to E_0.

CSSI - *Computational Supersingular Isogeny* problem

Given a supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$, the bases (U, V) for $E[2^n]$ and (P, Q) for $E[3^m]$, 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 E_0 be an elliptic curve and 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 $E_{AB} = E_0 / \big\langle [m_A]P_A + [n_A]Q_A,\ [m_B]P_B + [n_B]Q_B \big\rangle$.

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 as in the *SSCDH* problem and $E_{AB} = E_0 / \big\langle [m_A]P_A + [n_A]Q_A,\ [m_B]P_B + [n_B]Q_B \big\rangle$;
- $\Big(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_C\Big)$, where all the parameters are as in the *SSCDH* problem except for $E_C = E_0 / \big\langle [m'_A]P_A + [n'_A]Q_A,\ [m'_B]P_B + [n'_B]Q_B \big\rangle$ where 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 denoted by $\phi_0 : E \to E_0$, $\phi_1 : E \to E_1$. 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 kernel 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 the *2-inv-CSSI* problem for the following parameters: curves E, E_0, E_1, points $P, Q, U_0, V_0, U_1, V_1, \phi_0(P), \phi_0(Q), \phi_1(P), \phi_1(Q)$. We consider the following game between a party Bob and a challenger oracle:

- Bob initially knows E, E', U', V', and can only query the oracle;
- Bob sends to the challenger oracle a supersingular elliptic curve E' and the basis (U', V') for $E'[2^n]$;
- the oracle chooses four integers x_0, y_0, x_1, y_1 and the random points W_0 and W_1 of $E'[2^n]$. Then it computes the supersingular curves $F_0 = E'/\langle x_0U' + y_0V'\rangle$, $F_1 = E'/\langle x_1U' + y_1V'\rangle$, $F_0' = E'/\langle W_0\rangle$, $F_0' = E'/\langle W_1\rangle$;
- the oracle chooses 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 guess whether $b_0 = b_1$ or $b_0 \neq b_1$.

Bob's advantage in this game is defined as "$\mathcal{P}(c)$ *- 1/2*", where $\mathcal{P}(c)$ 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 introduced all the necessary hard problems and can finally approach the security model for the protocol. Starting with Alice, we must ensure that she cannot discover Bob's secret k otherwise she would know which of her secrets Bob would know hence breaking the security offered by the oblivious transfer. In a similar way, we must ensure that it is infeasible for Bob to discover the other secrets of Alice. This case is worse than the latter since it would completely exposes Alice messages if Bob were successful.
The two cases described are named Malicious Alice and Malicious Bob respectively. We will now describe them both and prove that both cases are computationally infeasible thus making the SIDH-based OT a secure post-quantum protocol.

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 formalised as the *Decisional Supersingular Isogeny* (*DSSI*) problem and is expected to be computationally intractable [12]. Nevertheless Alice has more information at her disposal: $\phi'_B(U_k)$, $\phi'_B(V_k)$ and the *two* pairs (U_i, V_i). It is possible now to consider a Weil pairing's property such that:

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

Now recall when in (section 3.3) Bob was required to compute *two* bases (U_0, V_0), (U_1, V_1) for $E[2^n]$. The bases were required to be chosen such that they have the same Weil pairing e_{2^n}. What would happen if Bob calculated his *two* bases with different pairings? Alice would be able to check all bases until she finds the correct one for which the condition above holds, as a consequence she would know Bob's k. It is then mandatory for Bob to choose bases having the same Weil pairing.
In order to prevent this scenario, [9] considers the *XDSSI* problem and its computational intractability. Considering the protocol settings, the *XDSSI* is equivalent to the 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 in the random oracle model relies on the hardness of the *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 *2-inv-CSSI*. A malicious Bob should solve the *2-inv-CSSI* without computing x_0, y_0, x_1, y_1, that is, solving the *CSSI*. 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). The problem described is precisely

how the oblivious transfer works although it 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 may choose a smaller basis thus limiting the possible values of $x_iU' + y_iV'$. For a small enough basis, it would be feasible for Bob to compute all the possible values of $x_iU' + y_iV'$. He would then be able to compute the curves all the possible curves of the form $F_j = E'_B / \langle x_iU' + y_iV' \rangle$, in particular he would find the curves F_0, F_1, the same computed by Alice! At this point Bob would need to compute the j-invariant for every curve F_j and use them as *seeds* for the *KDF*. By applying each seed to each of Alice's secrets, Bob would eventually be able to decrypt all of her secrets.
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). In this case Bob is in the situation to guess whether his chosen secret is b_0 or b_1 hence he must solve the *2-inv-DSSI* problem. Even though this problem is expected to be easier than its computational version *2-inv-CSSI* there is no known reduction to the *SSDDH* problem therefore the *2-inv-DSSI* can be considered computationally intractable.

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:

$$(X' : Z') = \Big(X\big(2X_4Z_4Z - X(X_4^2 + Z_4^2)\big)(X_4X - Z_4Z) : Z\big(2X_4Z_4X - Z(X_4^2 + Z_4^2)\big)(Z_4X - X_4Z)\Big)$$

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 $m' \in \{1, 2, \ldots, 2^{371} - 1\}$ and set $R_A = P_A + [2m']Q_A$.
A similar approach is used to sample 3^m-order points: choose a random integer $m' \in \{1, 2, \ldots, 3^{238} - 1\}$ and set $R_B = P_B + [3m']Q_B$.

Bibliography

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

[2] Alan Szepieniec, (2018). Mathematical and Provable Security Aspects of Post-Quantum Cryptography.

[3] Michele Mosca, (2015). Cybersecurity in an era with quantum computers: will we be ready? *Cryptology ePrint Archive.*

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

[5] Peter W. Shor, (1997). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing.*

[6] Luca De Feo, (2017). Mathematics of Isogeny Based Cryptography. *ArXiv.*

[7] Craig Costello, Patrick Longa, and Michael Naehrig, (2016). Efficient algorithms for supersingular isogeny Diffie-Hellman. *CRYPTO 2016.*

[8] Craig Costello, Benjamin Smith, (2017). Montgomery curves and their arithmetic, The case of large characteristic fields. *Journal of Cryptographic Engineering*

[9] Vanessa Vitse, (2019). Simple oblivious transfer protocols compatible with Kummer and supersingular isogenies. *IACR Cryptology ePrint Archive.*

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

[11] David Jao, (2019). Supersingular Isogeny Key Encapsulation. `https://sike.org/`.

[12] Luca De Feo, David Jao, Jérome Plut, (2014). Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. *PQCrypto 2014.*

[13] Federico Pintore, (2019). Cryptographic primitives from elliptic curve isogenies. *De Cifris Athesis*

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

[15] Steven D. Galbraith, Christophe Petit, Barak Shani, Yan Bo Ti, (2016). On the Security of Supersingular Isogeny Cryptosystems. *ASIACRYPT 2016*

[16] Patrick Longa, (2018). A Note on Post-Quantum Authenticated Key Exchange from Supersingular Isogenies. *IACR Cryptology ePrint Archive.*

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

[18] Joan Daemen, Vincent Rijmen, (2002). The Design of Rijndael. *Information Security and Cryptography.*

[19] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, (1997). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *CACM.*

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

[21] David Deutsch, (1985). Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences.*

[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. *Graduate texts in mathematics*

[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. ASIACRYPT 2017.*

[28] Shimon Even, Oded Goldreich, Abraham Lempel, (1982). A Randomized Protocol for Signing Contracts. *Communications of the ACM*. 28. 205-210. 10.1145/3812.3818.

[29] Kute, Vivek Paradhi, P Bamnote, Gajendra. (2009). A SOFTWARE COMPARISON OF RSA AND ECC. *International Journal Of Computer Science And Applications*. 2.

[30] Bosamia, Mansi Patel, Dharmendra. (2018). Current Trends and Future Implementation Possibilities of the Merkel Tree. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*. 6. 294-301.

[31] Wenbin Zhang and Chik How Tan., (2015). MI-T-HFE, A New Multivariate Signature Scheme. *In Proceedings of the 15th IMA International Conference on Cryptography and Coding* - Volume 9496 (IMACC 2015). Springer-Verlag, Berlin, Heidelberg, 43–56.

[32] Nicolas Sendrier, (2011). Niederreiter Encryption Scheme. *Encyclopedia of Cryptography and Security*. 842-843.

[33] Bhaskar Biswas, Nicolas Sendrier, (2008). McEliece Cryptosystem Implementation: Theory and Practice. *Post-Quantum Cryptography*. 47-62.

[34] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman (1998). NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory*. 267-288.

[35] Michael Rabin, (1981). How To Exchange Secrets with Oblivious Transfer. *IACR Cryptology ePrint Archive*. 2005. 187.