BACHELOR'S THESIS

Degree in Mathematics

Title: Isogeny-based post-quantum key echange protocols

Author: Ernest Sorinas Capdevila

Advisor: Jorge Luis Villar Santos

Department: Mathematics

Academic year: 2019-2020

UNIVERSITAT POLITÈCNICA DE C.
BARCELONATECH
Facultat de Matemàtiques i Estadís

Universitat Politècnica de Catalunya

Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

Isogeny-based post-quantum key exchange protocols

Ernest Sorinas Capdevila

Supervised by Jorge Luis Villar Santos

January, 2020

I would really much like to thank my supervisor, Jorge, for introducing me to this complex but very interesting topic, for guiding and advising me throughout the project and for all the time spent solving my doubts in our long meetings. I would also like to thank my friend Zaira for supporting me all this time and reviewing the work.

Abstract

The goal of this project is to understand and analyze the supersingular isogeny Diffie Hellman (SIDH), a post-quantum key exchange protocol which security lies on the isogeny-finding problem between supersingular elliptic curves. In order to do so, we first introduce the reader to cryptography focusing on key agreement protocols and motivate the rise of post-quantum cryptography as a necessity with the existence of the model of quantum computation. We review some of the known attacks on the SIDH and finally study some algorithmic aspects to understand how the protocol can be implemented.

Keywords

post-quantum, elliptic curve, isogeny, key agreement

Contents

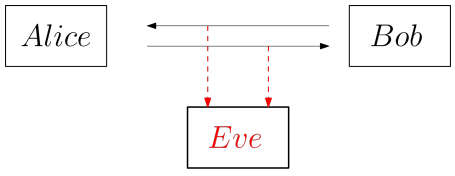
1. Introduction to cryptography

The purpose of this chapter is to introduce the reader to the topic of cryptography. We will define several concepts that we will need in the following chapters, such as public key cryptography, key agreement and security notions. Finally we will introduce the concept of post-quantum cryptography to motivate the appearance of the protocol we aim to understand, and we will briefly talk about the main post-quantum crytography active research fields.

We will assume the reader to have a background of elemental mathematics, algebraic structures -specially groups and finite fields- and some notions of algebraic geometry.

1.1 Basic concepts and definitions

Here we will assume that two users called *Alice* and *Bob* want to communicate with each another. There exist, of course, cryptographic protocols designed for more than two parties, but we will not study that. To determine the security of our scheme, we will assume the existence of a third user called *Eve* who will try to deduce the secret messages in the conversation. We will assume that Eve is "listening" to the conversation between Alice and Bob, receiving all of the information that Alice and Bob are sharing to one another. If we just let Eve listen, we will be studying how secure is our protocol to *passive attacks*. However, we will also consider the possibility that Eve does not only listen but also act in the middle of the conversation trying to corrupt it. For example, Eve could modify Alice's messages to Bob, or even re-write them, in order to obtain information from Bob's answer. Attacks where Eve is able to act directly on the information exchanged will be called *active attacks*, and have to be considered as well. We are still not considering all kinds of attack, but in this theoretical study it will be enough. We will talk about security in 1.4.



Alice needs to somehow *encrypt* the message that she wants to share with Bob in a way that Eve does not get much information (or none at all) about it. Of course, Bob has to be able to *decrypt* the message. Let us put some names over all these concepts.

First of all, let us define M as the set of all possible messages we could want to share in our scheme, and assume this set to be finite. Usually M will be all the possible n-bit chains, $M = \{0,1\}^n$, but sometimes we will assume M to be elements of a group G, because we can always code elements of a finite group as bit chains. Messages before (respectively, after) encryption will be called *plaintexts* (respectively, *ciphertexts*).

We need an encryption function, which we will call *Enc* or simply E and a decrypt function, which we will call *Dec* or simply D. Of course, *Dec* has to be the inverse of *Enc*, meaning that for any plaintext $m \in M$ we have $Dec(Enc(m)) = m$. We will assume that these functions are both public and also computable in polynomial time (we will define this concept in 1.4) .

At this point it is clear that we need something else for this to work. If Eve and Bob have the same information, how could Bob be able to recover the message but not Eve? We need Alice and Bob to have one or more secret parameters -keys- and use them to encrypt and decrypt. However, two possibilities arise: do Alice and Bob have the same key, or do they have different keys?

1.2 Symmetric key encryption

In a symmetric key encryption scheme Alice and Bob have a common key k in a finite set of possible keys K. Alice will encrypt a message m using the key k and Bob will decrypt the ciphertext using the same key. Namely, Alice will compute $c = Enc(m,k) = E_k(m)$, she will share it to Bob and he will decrypt it computing $Dec(c,k) = D_k(c) = m$. In this process, Eve will have seen the ciphertext c, and remember we assume that *Enc* and *Dec* are both public. Thus, the only thing that remains hidden from Eve is the shared key k, and we need her to not be able to get information of m using everything except for k.

Example 1. Cæsar cipher

This is a very simple example of a symmetric key encryption. Let M be the words in the Latin alphabet up to n letters, for some n. Choose a natural number $k \leq |M|$. To cipher a word m simply shift each of its letters k positions to the right in the ordered alphabet. For example, if $k = 2$, the plaintext $m = hello$ would be encrypted as $c = Enc(m,k) = jgnnq$.

This is not a safe encryption method. First of all, an attacker could try to decrypt with each possible key until a readable message appears, and that would not take long because of the key size. It has many other weaknesses, such as the repeated letters in m being repeated as well in c.

Example 2. One Time Pad

There is one cipher that solves these problems, called One Time Pad. It is simple: using a key k of size equal than (or greater) than the size of the plaintext m, to cipher we add it to m (as bit

strings, with the XOR operation): $Enc(m, k) = c = m \oplus k$. To decrypt the ciphertext, since XOR operation is self-inverse we just need to add k again, $Dec(c, k) = c \oplus k = m$.

This protocol is safe. It is actually the safest we can find. Assuming that the key is completely random, when Eve reads the ciphertext she gets no information at all about m, just like she received a random bit string and she had to guess the message. However, there are weaknesses.

First, if Alice decides to send another message to Bob and uses the same key, then Eve will get significative information about the plaintexts. If Eve has two ciphertexts encrypted with the same key, she can compute $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$, the XOR or the two plaintexts, and that is clearly unsafe. Hence, this cipher is only safe when the key is unused, and that is why it is called *One Time*. Also, the key size needed grows as much as the message, making it really hard to share long messages.

1.3 Public key encryption

In symmetric key encryption we studied how can we safely send messages assuming that Alice and Bob share a secret key that no one else knows. In this section we will study protocols that do not need the parties to previously agree in a key and also the Diffie Hellman key agreement for them to do so.

In this subsection, the communication will be one-way, breaking the symmetric scheme we had until now. Alice will have a public key p_k and a secret key s_k, and Bob will use Alice's public key to encrypt a message for her $c = Enc(m, p_k)$, which Alice will be able to decrypt using her secret key computing $m = Dec(c, s_k)$. In other words, the public key will be used to encrypt (and therefore everyone will be able to encrypt) but only those who know the secret key (in our case, Alice) will be able to decrypt those messages.

Example 3. ElGamal

This public key encryption scheme has a very characteristic structure, which can be used in many other protocols. The idea to encrypt is to generate a shared key with the public key, and then send the ciphertext together with a "clue" with which the other party can recover the shared key and then decrypt the message. Its security is based on the Discrete logarithm problem, which we will talk about later, but let us introduce the protocol.

The public parameters are a finite cyclic group G, its size n and a generator g. To generate her public and secret keys, Alice chooses a random integer $s \in \mathbb{Z}/n\mathbb{Z}$ and compute $h = g^s$. Her secret key is $s_k = s$, and all the other information is public $p_k = (G, n, g, h)$.

To encrypt a message m, which we assume to be an element of G, Bob chooses a random integer

$r \in \mathbb{Z}/n\mathbb{Z}$ and computes $h^r = g^{rs}$. This is the shared key that they will use this session. Then he ciphers m as $c_1 = m \cdot h^r$. The "clue" that he gives to Alice in order for her to deduce the shared key is $c_2 = g^r$. Finally he sends $c = (c_1, c_2)$.

To decrypt $c = (c_1, c_2)$, Alice first computes the shared key as $c_2^s = g^{rs}$. Then she decrypts c_1 as $m = c_1 \cdot (c_2^s)^{-1}$.

1.4 Security in cryptography

Clearly, we will need to talk about the security of the schemes we define. In order to do that, we introduce now some basic concepts of security in cryptography, focusing in decisional problems.

When studying asymptotic complexities of algorithms we will classify them depending on their running times using the "big O" notation.

Notation. "Big O". Let $f, g : \mathbb{N} \to \mathbb{N}$. If there exist $N, c \in \mathbb{N}$ such that $g(n) \geq cf(n)$ for all $n > N$ we will say that $f = O(g)$.

Larger keys usually leads to higher security, at least in the sense that the brute-force attack[1] complexity gets higher. What we would like is that this growth was a fast as possible, in order to have high security with low key sizes. We will study the time required to attack a scheme in asymptotic terms over a security parameter which we will denote by λ. This parameter is usually the size of the key used in the protocol, or maybe linearly proportional to it.

Definition 1. We will say that an algorithm is polynomial (or runs in polynomial time) if its running time is upper bounded by a polynomial on λ or, equivalently, if there exists $k \in \mathbb{N}$ such that the running time is $O(\lambda^k)$.

When there is no known polynomial algorithm to solve a problem, we will say that this problem is *hard*. Otherwise we will say it is *easy*.

The same way that we talk about functions (in this case, the running time of an algorithm) not growing too fast, we also define what we will consider to be small enough that is *negligible*.

Definition 2. We will say that a function $f(\lambda)$ is negligible if it decays faster than the inverse of any polynomial in λ or, equivalently, it decays faster than the inverse of λ^k for all k. Formally, f is negligible if $\forall k > 0$ there exists λ_0 such that $|f(\lambda)| < \frac{1}{\lambda^k}$ for all $\lambda > \lambda_0$. In this case, we will write $f \in negl(\lambda)$.

[1] Any cryptosystem can be attacked using the brute force or exhaustive attack, which means trying to decrypt the ciphertext using each possible key.

We will also talk about *decisional problems*, problems which answer can be represented with a single bit. In other words, if the answer of a decisional problem is either "yes" or "no". We will use the concept of decisional problem several times, and in fact they are very important in cryptography. They are classified in classes, such as **BPP** or the famous **P** and **NP** classes[2].

For decisional problems, let us define the concept of *advantage*. Informally, the advantage of a probabilistic algorithm A in a decisional problem family determines how good is it at distinguishing yes-problems (problems which answer is yes) than no-problems. Given a yes-problem and a no-problem at *random* (not necessarily uniformly distributed, but random by specific distributions of probability), the advantage will measure the probability that A has to distinguish them. From now on, consider a fixed security parameter λ.

Definition 3. Given D_0, D_1, distributions of probabilities of yes-problems and no-problems respectively, in a decisional problem family, the advantage of a probabilistic algorithm A is

$$Avg(A) := |Prob(A(p) = 1 \mid p \leftarrow D_1) - Prob(A(p) = 1 \mid p \leftarrow D_0)|$$

where $\leftarrow$ denotes picking a random yes/no-problem using the distrbution $D_{1/0}$.

If the advantage of A is negligible, then we will say that the distributions are *indistinguishable by A*, and the indistinguishability will become a very relevant property. If the distributions are indistinguishable for all polynomial algorithms, we will say that they are *indistinguishable*. This property will usually be very hard to prove.

1.5 Diffie Hellman key agreement

In symmetric key encryption, we needed both parties to know a previously agreed secret value in order to both encrypt and decrypt messages. We saw how, using public key encryption, users can communicate without having any previous agreement. Key agreement protocols are designed to agree in a key (without, of course, any previous agreements) in a way that the exchanged information does not leak any clue of the key.

First, there are some public parameters, which can be chosen by a trusted third party or just published for everyone to see. Using those parameters and (maybe) a random oracle, Alice and Bob will do some polynomial time computations, between which they might exchange information any number of times (in the protocols we will see, there will be just one moment in which they exchange

[2] An algorithm is said to be in **P** if it can be solved in polynomial time. A problem is said to be in **NP** if it can be verified in polynomial time whether an answer is correct or not. It is obvious that $\mathbf{P} \subseteq \mathbf{NP}$, however no problem has been proved to be in **NP** but not in **P**. Therefore, it is an open problem to determine whether $\mathbf{P} = \mathbf{NP}$ or not.

of information), and they will eventually be able to compute a common value to use as a key.

Many public key encryption schemes can be converted into key agreement schemes and vice-versa. We will explain now the most famous and used key agreement protocol, the Diffie Hellman, named after its creators Whitefield Diffie and Martin Hellman.

1.5.1 Diffie Hellman key agreement

The security of this protocol, as usually, will rely on a hard problem in cryptography, which is the *discrete logarithm problem*. The protocol works as follows:

Public parameters are a cyclic group G of size $n = |G|$ and a generator of the group $g \in G$. Remember that an element $g \in G$ is a generator if and only if the cyclic group generated by g is G. In other words, if any element $x \in G$ is a power of g, $x = g^r$ for some $r \in \mathbb{Z}$.

Now, Alice and Bob each chooses a random[3] element of $\mathbb{Z}/n\mathbb{Z}$ which we will call a and b, respectively. This integers are secret and they will not share them to each other. Instead, they will compute g^a and g^b, respectively, and send this to the other party. Alice will receive g^b and will compute $(g^b)^a = g^{ab}$ and Bob will receive g^a and compute $(g^a)^b = g^{ab}$. Now they share a secret, g^{ab}.

Public parameters

A cyclic group G, its size n and a generator g.

Alice	**Bob**
Chooses random $a \in \mathbb{Z}/n\mathbb{Z}$ and computes g^a.	Chooses random $b \in \mathbb{Z}/n\mathbb{Z}$ and computes g^b.
Sends $U = g^a$ to Bob and receives Bob's message V.	Sends $V = g^b$ to Alice and receives Alice's message U.
Computes $V^a = g^{ab}$, the shared secret key.	Computes $U^b = g^{ab}$, the shared secret key.

1.5.2 Security of DH and associated problems

In the Diffie Hellman protocol, all the information available for an eavesdropper is G, n, g, g^a and g^b. Computing g^{ab} with that information is known as the *Computational Diffie Hellman* problem (CDH) and is believed to be hard for a generic group. It is worth mentioning that it is not equivalent

[3]When we say *random* element in a finite set S we will always assume we are using the uniform distribution where each element is chosen with probability $\frac{1}{|S|}$.

to the *Discrete Logarithm Problem* (DLOG) nor the *Decisional Diffie Hellman* (DDH) problem. Let us introduce them.

1. CDH: Given G, n, g, g^a and g^b compute g^{ab} where a, b are chosen randomly in $\mathbb{Z}/n\mathbb{Z}$.
2. DLOG: Given G, n, g and g^a compute a where a is chosen randomly in $\mathbb{Z}/n\mathbb{Z}$.
3. DDH: Win Game 1 with non-negligible advantage.

Game 1. In this game there will be two parties: the *challenger* and the *attacker*. The game proceeds as follows:

1. The challenger generates the public parameters of a DH key agreement protocol, $PP := (G, n, g)$. He also chooses random $a, b \in \mathbb{Z}/n\mathbb{Z}$ and computes g^a and g^b.
2. The challenger generates a random bit $b \in \{0, 1\}$. Then,
   - if $b = 0$, he chooses a random $r \in \mathbb{Z}/n\mathbb{Z}$ and sends PP, g^a, g^b and g^r to the attacker.
   - if $b = 1$, he computes g^{ab} and sends PP, g^a, g^b and g^{ab} to the attacker.
3. The attacker, with only the information he receives from the challenger and in polynomial time outputs a bit $b^* \in \{0, 1\}$.
4. The attacker wins if and only if $b = b^*$.

These problems are not known to be equivalent. However, there are some clear reductions. If the attacker can solve CDH, then he can solve DDH as well. Also, if he is able to solve DLOG, then he is also capable of solving both CDH and DDH. No general inverse reductions have been found, but it has not been demonstrated that they are not equivalent yet.

1.5.3 Group action Diffie Hellman

Definition 4. Let G be a group and X a set. An action of G on X is a map

$$G \times X \to X$$
$$(g, x) \longmapsto g * x$$

such that

- For all $g, h \in G$, $x \in X$ we have $g * (h * x) = (gh) * x$.

- For all $x \in X$ we have $1 * x = x$.

Example 4. Let G be a group (which will act as X in the definition). The group of integers $\mathbb{Z}$ (with the sum operation) acts on G as

$$\begin{aligned} \mathbb{Z} \times G &\longrightarrow G \\ (n, g) &\longmapsto g^n. \end{aligned}$$

It is clear that this is an action because given $n, m \in \mathbb{Z}$ and $g \in G$ we have $(g^m)^n = g^{mn}$ and $g^1 = g$.

In fact, what we did in the Diffie Hellman key agreement was using this group action on a cyclic group G, and the resulting key was the same precisely thanks to the action commutative property. More generally, given an action

$$\begin{aligned} a : G \times X &\to X \\ (g, x) &\longmapsto a(g, x) = g * x \end{aligned}$$

one can define a Diffie Hellman key agreement scheme as follows:

Public parameters A finite abelian group G, a set X, an element $x \in X$ and an action a of G over X.

Alice	**Bob**
Chooses a random element $g \in G$ and computes $g * x$.	Chooses a random element $h \in G$ and computes $h * x$.
Sends $U = g * x$ to Bob and receives Bob's message V.	Sends $V = h * x$ to Alice and receives Alice's message U.
Computes $g * V = g * (h * x) = (gh) * x$, the shared secret key.	Computes $h * U = h * (g * x) = (hg) * x$, the shared secret key.

It is easy to check that the DH agreement we defined previously is a particular case of this scheme, using the parameters of Example 4. Also, note that we are using the fact that the group G is commutative to say that $(gh) * x) = (hg) * x$. In fact, there are generalizations of this idea that do not require the group to be abelian, but in the end we need to somehow guarantee that the actions of gh and hg on x are equal.

1.6 Quantum and post-quantum cryptography

All we have seen up to this point is called classical cryptography. We have assumed the classical model of computation, in which a computer is essentially a machine with an internal state made of bits that periodically changes this state using logic gates. The protocol we aim to understand, however, is a post-quantum protocol, in which we assume the attacker to have a machine called quantum computer. It is not in our purposes to understand the quantum computation model, because we do not really need it to make post-quantum cryptography (PQC). In PQC, we only assume the attacker to have such machine and, hence, to be able to run all the known quantum algorithms. Let us show the most famous quantum algorithms and why they are a problem to classical cryptography.

1.6.1 Shor's and Grover's algorithms

In 1994, with the model of quantum computation arising, Shor [Sho94] shows how to efficiently factorize a number[4] and compute the discrete logarithm in a quantum computer. Two years later, another revolutionary result is given by Grover [Gro96]. Given a function $f : A \to B$ as a black-box and an output value $b \in B$, Grover's algorithm finds with high probability the unique element $a \in A$ such that $f(a) = b$ in $O(\sqrt{|A|})$ function evaluations.

Many of the problems which have been found to be easy in quantum computers can be seen as particular cases of a problem called *abelian hidden subgroup problem*.

Problem 1. (Hidden Subgroup Problem) Let $f : G \to R$ be a function given as a black-box with its domain G being a known finite group. Assume that there exists a subgroup $H \subset G$ for which the following holds $\forall x, y \in G$

$$f(x) = f(y) \Leftrightarrow xH = yH.$$

The *hidden subgroup problem* is to find a generating set of H given f, G, R. If the group G is abelian, the problem is called *abelian hidden subgroup problem*.

A generalization of Shor's algorithm solves the abelian HSP. In [Wan10] Wang explains this algorithm and how the DLOG and the factorizing problems can be seen as particular cases of abelian-HSP. However, it remains as an open problem if the HSP problem for arbitrary groups can be efficiently solved in a quantum computer .

It is clear how these algorithms affect classical cryptographic schemes. Most of the protocols

[4] Here, "efficiently" means that the algorithm runs in polynomial time on the size of the number to factorize (if the number is n, the running time in a quantum computer is bounded by a polynomial in $log(n)$).

used in classical cryptography rely on the hardness of these problems. For example, the security of one of the most used public key encryption schemes, the RSA, lies directly on the hardness of the factoring problem. Shor's and Grover's algorithms break most of the vastly used classical protocols.

General purpose quantum computers with enough power to break the standard protocols with these algorithms are not known to exist yet. However, with the model and the algorithms on the table, it is only a matter of time until classical protocols become unsafe. Hence, the necessity of schemes that resist an attack from a quantum computer arises as a major priority in the cryptographic world. This motivates the research of quantum-resistant hard problems and protocols based on them, and the protocol we aim to understand here is one of them. We will explain later the isogeny-finding problem, but let us show some examples of research fields in which quantum-hard problems have been found.

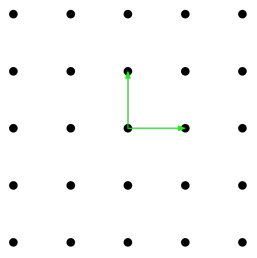
1.6.2 Lattice based cryptography

We will introduce lattice-based cryptography with some of the problems considered hard in the field. For first time readers of the topic we recommend [CCSKK15] and to go deeper in the cryptosystems that have been proposed using these problems we suggest [P+16].

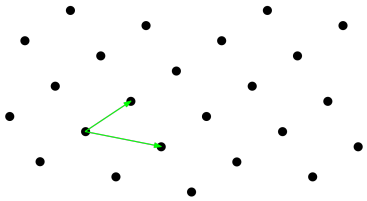
Definition 5. Let $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ be a collection of linearly independent vectors $\mathbf{b}_i \in \mathbb{R}^k$. We define the lattice generated by $\mathbf{B}$ as

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} \mathbf{b}_i c_i \text{ where } c_i \in \mathbb{Z} \right\}.$$

Example 5. One can think of a lattice as a collection of points in $\mathbb{R}^k$ with a particular structure. For example, taking $\mathbf{B} = \{(1,0),(0,1)\}$, the canonical base of $\mathbb{Z}^2$, the lattice $\mathcal{L}(\mathbf{B})$ is represented as in Figure 1a. Of course, the following figures show a part of the lattices, since they are infinite.



(a) Lattice defined by $\mathbf{B} = \{(1,0),(0,1)\}$.



(b) Representation of another lattice.

Figure 1

Let us show some problems that are considered hard in lattice-based cryptography.

Definition 6. *Shortest Vector Problem (SVP)*

Given an arbitrary basis **B** of a lattice $\mathcal{L}$, find a shortest non-zero vector (with the euclidean norm).

The norm of a shortest non-zero vector

$$\lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L} \setminus \{0\}} ||v||$$

is also known as the *minimum distance in* $\mathcal{L}$.

There are many variants of this problem

- *Decisional SVP*: the decisional version of SVP.
- *Approximate SVP*: Given a scalar $\epsilon \geq 1$ find a non-zero $v \in \mathcal{L}$ such that $||v|| \leq \epsilon \lambda_1(\mathcal{L})$.

Problem 2. *Closest Vector Problem (CVP)*

Given an arbitrary basis **B** of a lattice $\mathcal{L} \subset \mathbb{R}^n$ and a vector $v \in \mathbb{R}^n$ (not necessarily in $\mathcal{L}$), find the closest vector (with the euclidean norm) to v in $\mathcal{L}$.

Finally, let us introduce a lattice-related problem called *learning with errors*, which has inspired many post-quantum cryptographic protocols. Let $s \in \mathbb{Z}/q\mathbb{Z}$ be the secret vector and ξ a probability distribution which generates "short" elements, usually taken as the discrete Gaussian over $\mathbb{Z}$. We define an oracle O which does the following:

- Chooses $e \leftarrow \xi$ from the probability distribution ξ.
- Chooses a vector $a \leftarrow (\mathbb{Z}/q\mathbb{Z})^n$ at random.
- Outputs $(a, \langle a, s \rangle + e)$, where $\langle a, s \rangle$ denotes the inner product of a and s.

The computational learning with errors problem is to find s given access to polynomial many calls to the oracle O.

1.6.3 Code-based cryptography

Another important research field in post-quantum cryptography is code theory and its hard problems.

Definition 7. A linear (n, k)-code $\mathcal{C}$ over the finite field $\mathbb{F}_q$ is a subspace of dimension k of $\mathbb{F}_q^n$ as a vector space. A basis of $\mathcal{C}$ in matrix form, $C \in \mathcal{M}_{n \times k}(\mathbb{F}_q)$, is called a generating matrix of $\mathcal{C}$. Given $m \in \mathbb{F}_q^n$, we define its encoding as $c := mC \in \mathcal{C}$. Elements of $\mathcal{C}$ are called *codewords*.

We define the distance between two codewords as the *Hamming distance*, which is the number of positions in which the two words differ. A crucial property of codes is that, given a codeword c^*, the minimum distance between c^* and another codeword is independent from c, because

$$\max_{c \in C}\{dist(c^*, c)\} = \max_{c \in \mathcal{C}}\{dist(c^* - c, 0)\} = \max_{c \in \mathcal{C}}\{dist(c, 0)\}.$$

Let D be this maximum distance, and let $x = c + e$ with $c \in \mathcal{C}$ and $e \in \mathbb{F}_q^n$ with $||e|| < D/2$. Thanks to this property, we know that c the closest element of $\mathcal{C}$, and hence c is uniquely determined from x.

This induces a problem: Given an element x of the form $x = c + e$ as before, determine c. This operation is called *decoding*. It turns out that there is a particular type of codes, called Goppa codes, in which decoding can be computed efficiently.

In the McEllice cryptosystem [McE78], Alice chooses a Goppa code, a generating matrix G, and also a dense invertible matrix S and a random $n \times n$ permutation matrix P. She then publishes $G' = SGP$ as her public key, and keeps S and P as her secret. The matrix G' generates a code with the same minimum distance between points as G. The encryption of a message consists in encoding with G' and adding a random small enough error, $c = mG' + e$. Then, to decrypt c, Alice computes $cP^{-1} = mSG + eP^{-1}$, and then she can efficiently decode mS because G generates a Goppa code. Finally, multiplying this quantity by S^{-1} she recovers the original message.

The McEllise cryptosystem has inspired some post-quantum protocols, such as the Classic McEllise [BCL+17], competing to become a standard by NIST as the protocol we aim to study, SIKE.

2. Elliptic curves

Elliptic curves and their uses in cryptography is a very active field of study because they have a very interesting structure. They are defined as an algebraic variety as the solutions to a certain kind of equation, which gives them a geometric structure. However, they happen to have a very natural group structure as well. That gives a hint on the importance of elliptic curves, and also of isogenies, which will be a kind of operation between elliptic curves that preserve both aspects. As a reference on this topic we recommend the first chapter of Conell's book [Con96].

2.1 What is an elliptic curve?

There are different ways to define what is an elliptic curve, we will now introduce some of them. We will assume that the field on which we define the elliptic curve has characteristic greater than 3. This assumption makes sense in our context because most of the time we will work with curves over $\mathbb{F}_q$ for a large prime -or prime power- q.

For the change of variables from one form to another, more information and a deep analysis on the forms of elliptic curves see the reference book [Con96].

Definition 8. (Projective and short Weierstrass form) Given a, b in a field K (with $char(K) > 3$) such that $4a^3 + 27b^2 \neq 0$, we define an elliptic curve E as the locus in the projective plane $P^2(K)$ of the equation

$$Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{1}$$

.

For $Z \neq 0$ one can define the change of coordinates $x = X/Z, y = Y/Z$, which leads to the *short Weierstrass equation*:

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

.

Remember that points in the projective space are defined up to multiplication by a non-zero constant. Since the only projective point with $Z = 0$ that satisfies the equation 1 is the point $(0 : 1 : 0)$, one can also define the elliptic curve by its short Weierstrass equation plus the point $(0 : 1 : 0)$, which we will call $\mathcal{O}$ or *point at infinity*.

Observation 1. The condition of $4a^3 + 27b^2 \neq 0$ is more intuitive than one could think. We are defining the curve implicitly as the solutions of an equation or a multi-variable function being equal

to zero, hence we need to be able to invert the function. In order to do this, we need the partial derivatives not to be simultaneously zero, and that is exactly what the condition does. In other expressions for the elliptic curve the conditions will change, as we will see now.

Definition 9. (Weierstrass form) One can also define an elliptic curve in its (not reduced) *Weierstrass form* as the solutions of an equation of the form

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

plus the point at infinity $\mathcal{O}$. The coefficients a_i are taken in K and there is also a condition on them for the curve not to be singular (although a much longer one). Defining the coefficients

$$b_2 = a_1^2 + 4a_2,\ b_4 = 2a_4 + a_1a_3,\ b_6 = a_3^2 + 4a_6 \text{ and } b_8 = a_1^2a_6 - a_1a_3a_4 + 4a_2a_6 + a_2a_3^2 - a_4^2$$

the condition becomes

$$-b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6 \neq 0$$

It can be seen with a change of variables that an equation in Weierstrass form can be expressed in short Weierstrass form. That is why we will usually work with the second one.

Definition 10. (Montgomery form) Another common form for an elliptic curve is the *Montgomery form*. Any elliptic curve can be written as the locus of the equation

$$by^2 = x^3 + ax^2 + x$$

plus the point at infinity $\mathcal{O}$. The coefficients a, b are taken in K and the condition in this case becomes $b(a^2 - 4) \neq 0$.

Example 6. To get familiar with elliptic curves and see how they look in their common representation, let us introduce the curve $secp256k1$, which we will talk about later, to give an example.

Short Weierstrass form of the $secp256k1$: $y^2 = x^3 + 7$.

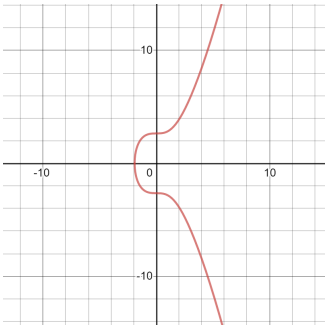


Figure 2: Geometrical representation of *secp*256*k*1.

This curve is typically used in a digital signature called ECDSA (Elliptic Curve Digital Signature Algorithm), used by the Bitcoin protocols. This example shows that we do not need very large parameters for a and b to make the protocols secure. Instead, what will become important will be the finite field K that we choose to set our curve in. For example, in the case of *secp*256*k*1, the field taken is $\mathbb{F}_p$ where $p = 2^{256} - 2^{32} - 977$. For more information about this protocol see [KD13].

2.1.1 Group law

To make cryptography we need some sort of mathematical structure. For now, elliptic curves are defined as algebraic varieties. In this subsection we talk about their group structure. Since we are working with elliptic curves over finite fields, this should not be a surprise for anyone. Given a finite field K and an elliptic curve $E = E(K)$ with n points, there is a natural bijection with $\{0, \dots, n-1\}$ and one can transport the group structure of $\mathbb{Z}/n\mathbb{Z}$ to E using this bijection. However, we need more than that to make cryptography. We need this group law to be relatively natural and, most importantly, efficiently computable. Fortunately, elliptic curves have a very intuitive group law, at least from a geometrical point of view. We will first see this intuitive geometrical approach and afterwards we will give the algebraic formal definition.

2.1.2 Geometrical approach

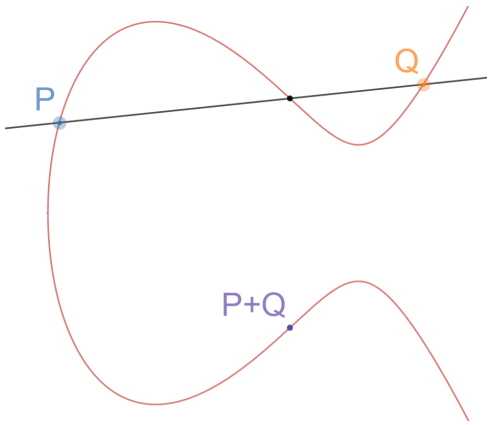
Let $E = E(K)$ be an elliptic curve over a finite field K and consider the geometrical representation of its short Weierstrass form in the plane. We define the group operation $+$ as follows.

First of all, we set the point at infinity $\mathcal{O}$ to be the neutral element. This means that for any P in E, $P + \mathcal{O} = \mathcal{O} + P = P$.

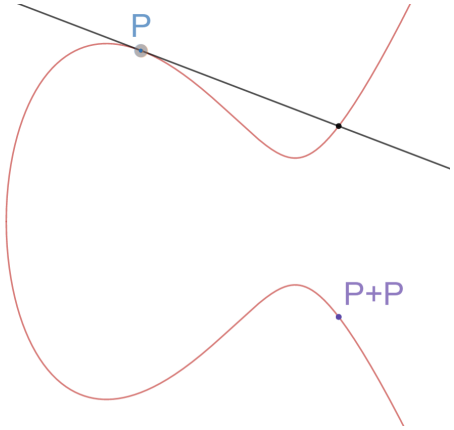
Now let P and Q be two different points in E different than $\mathcal{O}$. Joining this two points we get a line, which will intersect a third point of the curve. Note that this line could be a vertical line. In this case, it will intersect the curve at $\mathcal{O}$, and we define $P + Q = \mathcal{O}$. Otherwise we define $P + Q$

to be the x-axis reflection of this third point. See Figure 3a for an example.

Given $P \in E$ we still have to define $P + P$. In this case, take the tangent line to the point P. This line intersects the curve in exactly one more point. Note that this point could be $\mathcal{O}$. In this case, we define $P + P = \mathcal{O}$. Otherwise, we define $P + P$ as the x-axis reflection of that point. See Figure 3b for an example.



(a) Sum of two different points.



(b) Sum of one point with itself.

But why does this work? We are implicitly working in the projective space, and therefore we need to be careful when using intuition. In our definition we assumed that a line intersects the curve in exactly three points, up to multiplicity. For this to be mathematically rigorous we need to use Bezout's Theorem. Since the elliptic curve is a projective variety given by a polynomial of degree 3, and the line is a projective variety given by a polynomial of degree 1, Bezout's Theorem tells us that they will intersect in exactly $3 \times 1 = 3$ points, up to multiplicity.

2.1.3 Algebraic definition

Let E be an elliptic curve with short Weierstrass form $y^2 = x^3 + ax + b$.

Given P, Q two points of the elliptic curve, we define $P + Q$ as follows

- If $P = \mathcal{O}$ set $P + Q = Q$.
- If $Q = \mathcal{O}$, set $P + Q = P$.
- Otherwise, set $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$ and
  - If $x_P = x_Q$ and $y_P = -y_Q$, set $P + Q = \mathcal{O}$.

– Otherwise, define λ as

$$\lambda := \begin{cases} \frac{y_P - y_Q}{x_P - x_Q} \text{ if } P \neq Q \\ \frac{3x_P^2 + a}{2y_P} \text{ if } P = Q. \end{cases}$$

and set $P + Q = (x_S, y_S)$ where $x_S = \lambda^2 - x_P - x_Q$ and $y_S = \lambda(x_P - x_S) - y_P$.

Observation 2. When talking about the computational costs of cryptographic protocols involving elliptic curves, optimizing the costs of arithmetic operations will be crucial. It turns out that curves in Montgomery form have good properties in that sense. That is why, in the implementations of the protocol we aim to study, curves will usually be specified to be in Montgomery form.

2.2 Important concepts

Definition 11. Let E be an elliptic curve. For a non-negative integer m we define the *multiplication-by-m map* as

$$\begin{aligned} [m] : E &\longrightarrow E \\ P &\longmapsto mP := P + \overset{m}{\cdots} + P \end{aligned}$$

and for a negative integer $m < 0$ we define the multiplication-by-m as

$$\begin{aligned} [m] : E &\longrightarrow E \\ P &\longmapsto mP := [-m][-P] = (-P) + \overset{-m}{\cdots} + (-P). \end{aligned}$$

Notation. Using the natural embedding

$$\begin{aligned} \mathbb{Z} &\longrightarrow End(E) \\ n &\longmapsto [n] \end{aligned}$$

we will sometimes write the multiplication-by-n map as n instead of $[n]$ and consider $\mathbb{Z}$ as a subset of the endomorphism ring $End(E)$.

Given an integer l, we will be interested in for which points we have $[l]P = \mathcal{O}$. Let us define this concept.

Definition 12. Let $E = E(K)$ be an elliptic curve over a field K. For an integer l we define the *l-torsion subgroup* as

$$E[l] := \left\{ P \in E(\overline{K}) \mid [l]P = \mathcal{O} \right\}.$$

Proposition 1. *For any integer l, $E[l]$ is a subgroup of E.*

Proof. It is not hard to check that the group requirements hold for $E[l]$. However, it is easier and will be more helpful to note that $E[l] = ker([l])$. Since elliptic curves are abelian groups, the multiplication-by-m map is a group morphism, and therefore its kernel is a subgroup of E. □

There is one crucial result about the structure of the l-torsion groups that we will use several times.

Proposition 2. *([DF17], Proposition 4) Let $E = E(K)$ be an elliptic curve over a finite field K with characteristic p. Then, if $gcd(p, l) = 1$ we have $E[l] \cong \mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$.*

Elliptic curves can be classified in two categories: *supersingular* and *ordinary* ones.

Definition 13. Let E be an elliptic curve over $\mathbb{F}_{p^m}$. We will say that E is supersingular if $E[p] = \{\mathcal{O}\}$. We will say that E is ordinary if it is not supersingular.

There are many differences between ordinary curves and supersingular ones, and the protocol we aim to understand here, the SIDH, uses the latter. However, we will also be interested in the properties of ordinary curves because other cryptographic schemes have been proposed using them. We will talk about these protocols and the differences between ordinary and supersingular curves in 3.5.

Another morphism which we will be interested in is the Frobenius endomorphism.

Definition 14. Let E be an elliptic curve defined over $\mathbb{F}_q$. We define the *Frobenius endomorphism* as

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

Proposition 3. *([Sch87], Proposition 3.6) Let E be an elliptic curve over $\mathbb{F}_q$ and π_q the Frobenius endomorphism. Then, there exists a unique $t \in \mathbb{Z} \subset End(E)$ such that π_q satisfies the equation $\pi_q^2 - t\pi_q + q = 0$.*

Definition 15. We will call t from Proposition 3 the *trace* of the Frobenius endomorphism or simply *trace of Frobenius*.

Let us now introduce the j-invariant, which will play a crucial role in elliptic curve and isogeny-based cryptography.

Definition 16. Let E be an elliptic curve with short Weierstrass form $y^2 = x^3 + ax + b$. The *j-invariant* of E is

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

Of course, with the corresponding change of variables, the j-invariant can be defined for curves in Weiestrass form, Montgomery form, etc. The reason why this value will play such an important role is because the j-invariant characterizes a curve, up to isomorphism. In other words, two elliptic curves E, E' are isomorphic if and only if $j(E) = j(E')$. For a proof of this result see ([Sut17], Theorem 14.14). With this property, when making isogeny-based cryptography we will usually work with isomorphism classes of elliptic curves.

Observation 3. It is worth mentioning that, given a j-invariant (a quantity that is the j-invariant of some curve over a fixed field $\mathbb{F}_q$), one can also compute an elliptic curve with that j-invariant. In fact, it is easy to check that, given $j \neq 0, 1728$, the curve E with short Weierstass equation

$$y^2 = x^3 + \frac{3j}{1728 - j}x + \frac{2j}{1728 - j}$$

has $j(E) = j$ (other examples can be found in [Con96]). For $j = 1728$, we have the curve $E : y^2 = x^3 + x$ and for $j = 0$, the curve $E : y^2 = x^3 + 1$.

2.3 Structure and point counting

We now analyze some basics on the group structure of elliptic curves. The proofs of most of the results we use here can be found in [Sch87] and [Wat69]. A well known result is that an elliptic curve is isomorphic to either a cyclic group or a product of two cyclic groups (see [Sch87], Lemma 4.8). Let us show a remarkable result of point counting on elliptic curves, known as *Hasse's bound* or *Hasse's interval*.

Theorem 1. *(Hasse) Let E be an elliptic curve defined over $\mathbb{F}_{p^n}$ for a prime p and integer n. Then,*

$$|E(\mathbb{F}_{p^n})| = p^n + 1 - t_n \text{ with } |t_n| \leq 2\sqrt{p^n}.$$

For a proof of this result we refer to ([Sil09], Theorem V.1.1). It turns out that the quantity t_n is the trace of Frobenius and it is a multiple of p if and only if the curve E is supersingular ([Sch87], Proposition 3.6). Joining this results we deduce the following:

- For a supersingular curve defined over $\mathbb{F}_p$ for a prime p, the only possible value for t_1 with $|t_1| \leq 2\sqrt{p}$ is 0. Therefore, we have $|E(\mathbb{F}_p)| = p + 1$. In this case, $E(\mathbb{F}_p) \cong \mathbb{Z}/(p+1)\mathbb{Z}$.

- For a supersingular curve defined over $\mathbb{F}_{p^2}$, the theorem tells us that $t_2 \in \{-2p, -p, 0, p, 2p\}$.

More generally, a classical result from Waterhouse [Wat69] determines for which values of t_n there exists a supersingular elliptic curve over $\mathbb{F}_q = \mathbb{F}_{p^n}$ with Frobenius endomorphism of trace t_n.

Let us use Waterhouse's result to focus on the case that we will study ($q = p^2$), so we can be more specific.

Proposition 4. *([Wat69], Theorem 4.1) There exist supersingular elliptic curves over F_{p^2} with trace of Frobenius t_2 if and only if one of the following holds:*

- $t_2 = \pm 2p$.
- $t_2 = \pm p$ *and* $p \not\equiv 1$ *mod*(3).
- $t_2 = 0$ *and* $p \not\equiv 1$ *mod*(4).

Observation 4. In the SIDH, the conditions $p \not\equiv 1$ mod(3) and $p \not\equiv 1$ mod(4) happen to hold false for the primes chosen in many implementations, which are of the form $p = 2^{e_A}3^{e_B}d + 1$ for an integer d and large e_A, e_B. In this case, we then have that $|E(\mathbb{F}_{p^2})| = p^2 + 1 \mp 2p = (p \mp 1)^2$ and therefore $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(p \mp 1)\mathbb{Z} \times \mathbb{Z}/(p \mp 1)\mathbb{Z}$.

3. Isogenies and SIDH

3.1 Isogenies

Definition 17. An *isogeny* is a non-constant rational map[5] between elliptic curves $\phi : E(\overline{K}) \to E'(\overline{K})$ such that $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$. We will say that two curves are *isogenous* if there is an isogeny ϕ from E to E'.

One important thing that we will use several times about isogenies and that may not be clear from this definition is that they are morphisms of curves (see [Gal12], Lemma 7.3.6). Also, since a morphism of projective curves is either constant or surjective ([Sut17], Theorem 5.12) and we disallowed the constant functions, isogenies are surjective as well.

Example 7. The *negation map* defined as follows (taking E in the projective space as defined in Definition 8)

$$\begin{aligned} \phi : E &\longrightarrow E \\ (X : Y : Z) &\longmapsto (X : -Y : Z) \end{aligned}$$

that sends P to its opposite by the elliptic curve group law is an isogeny, since we have already defined it in a rational map form and $\phi(\mathcal{O}) = \phi(0 : 1 : 0) = (0 : -1 : 0) = (0 : 1 : 0) = \mathcal{O}$.

Actually we already know some functions that are isogenies. For example, the identity map is an isogeny, and the multiplication-by-n map $[n](P) = P + \overset{n}{\cdots} + P$ is an isogeny as well.

3.1.1 Degree and separability

Let $\phi : E \to E'$ be an isogeny defined over K, and let $K(E)$ and $K(E')$ be the function fields of E and E', respectively. Given a function $f \in K(E')$, we can compute $\phi^*(f) = f \circ \phi : E \to E'$ and we obtain $\phi^*(f) \in K(E)$ (see Figure 4).

[5] See [[Sut17], Definition 5.6] for the *rational map* definition.

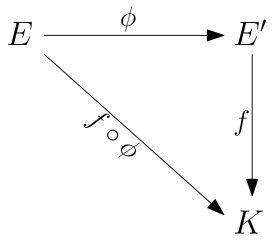


Figure 4: Diagram of the induced $\phi^*(f) = f \circ \phi : E \to K$.

Hence, since isogenies are injective we have that an isogeny ϕ induces a injective map

$$\begin{aligned} \phi^* : K(E') &\longrightarrow K(E) \\ f &\mapsto \phi^*(f) = f \circ \phi \end{aligned}$$

Additionally, this yields a field extension $K(E)/Im(\phi^*)$.

Definition 18. We define the *degree of* ϕ as the degree of this field extension, $deg(\phi) := [K(E) : Im(\phi^*)]$.

Definition 19. We will say that the isogeny ϕ is separable if the field extension $K(E)/Im(\phi^*)$ is.

For separable isogenies, we have an equivalent definition of the degree, which is easier to determine and compute.

Proposition 5. *([Sil09], Theorem 4.10) Given a separable isogeny ϕ, we have that $deg(\phi) = |ker(\phi)|$.*

It turns out that the degree has a very useful property, which we will use implicitly several times from now on: it is multiplicative under composition of isogenies.

Proposition 6. *([Sut17], Corollary 6.8) Given a composition of isogenies $\gamma = \psi \circ \phi$ we have*

$$deg(\gamma) = deg(\phi)deg(\psi).$$

3.1.2 Fundamental results

One of the reasons why we are interested in isogenies is that they preserve both facets of elliptic curves, the algebraic variety and the group structure. Let us show some important results over isogenies, some of which will be extremely necessary when constructing the SIDH protocol.

Proposition 7. *([Sut17], Lemma 5.21) Let E, E' be elliptic curves over K in short Weierstrass form and $\phi : E \to E'$ an isogeny between them. Then there exist polynomials $u(x), v(x), s(x), t(x)$ with*

$gcd(u, v) = gcd(s, t) = 1$ *such that*

$$\phi(x, y) = \left(\frac{u(x)}{v(x)}, \frac{s(x)}{t(x)} y \right).$$

This *standard* form for isogenies is a useful tool when algebraically working with them. It gives a lot of information, specially the roots of these polynomials, which will define the kernel of the isogeny. Also, with this definition we can give alternative (but equivalent) definitions for degree and separability.

Proposition 8. *([Sut17], Remark 5.27) Let ϕ be an isogeny in standard form, $\phi(x, y) = \left(\frac{u(x)}{v(x)}, \frac{s(x)}{t(x)} y \right)$,*

1. *The degree of ϕ is $deg(\phi) = max\{deg(u), deg(v)\}$.*
2. *The isogeny ϕ is separable if and only if the derivative of $\frac{u(x)}{v(x)}$ is nonzero.*

It will be useful in many situations to have such a natural expression for isogenies. However, this proposition tells us that, when it comes to computers, storing an isogeny in such form requires storing coefficients of these polynomials, one of which (at least) has degree $deg(\phi)$. Hence, the space required is $O(|deg(\phi)|)$.

Definition 20. Given an isogeny $\phi : E \to E'$ of degree n, one can find another isogeny in the opposite direction $\widehat{\phi} : E' \to E$, which we will call the *dual isogeny*[6]. It is worth noting that this isogeny is not the inverse of ϕ, meaning that $\widehat{\phi} \circ \phi \neq Id_E$. However, this composition is indeed a function we already know:

$$\widehat{\phi} \circ \phi = [n]_E$$

where $[n]_E$ denotes the multiplication-by-n map in E.

It is also important to have in mind that it is not a hard problem to determine if two curves are isogenous. This is thanks to Tate's theorem:

Theorem 2. *(Tate) Two elliptic curves E, E' defined over $\mathbb{F}_q$ are $\mathbb{F}_q$-isogenous if and only if $|E(\mathbb{F}_q)| = |E'(\mathbb{F}_q)|$.*

Since we will be working with isogenies between supersingular elliptic curves, the next result is crucial.

Proposition 9. *([Sut17], Theorem 14.1) Let $\phi : E(\overline{K}) \to E'(\overline{K})$ be an isogeny. Then, E is supersingular if and only if E' is supersingular.*

[6] A proof and detailed explanation of this can be found in ([Sil09], III.6).

The proof of this proposition is not hard. Considering the composition $[p]_{E_2} \circ \phi = \phi \circ [p]_{E_1}$, where $p = char(K)$ and $[p]_{E_i}$ denotes the multiplication-by-p in the curve E_i, one concludes that $deg([p]_{E_1}) = deg([p]_{E_2})$ and the proposition follows.

Finally, let us show a result that we will be using many times explicitly in the scheme: Vélu's formulas.

Theorem 3. *(Vélu) Let E be an elliptic curve in short Weierstrass form $y^2 = x^3 + ax + b$. Given a subgroup G of E, there exists a unique -up to isomorphism- elliptic curve E_G and a unique -up to isomorphism[7]- isogeny ϕ with $\phi : E \to E_G$ such that $ker(\phi) = G$.*

For $P \notin G$ the isogeny ϕ is given by the formula

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

where $x(A)$ denotes the x-coordinate of the point A and $y(A)$ its y-coordinate.

The short Weierstrass form of $Im(\phi) = E/G$ is $y^2 = x^3 + a'x + b'$ where

$$a' = a - 5 \sum_{Q \in G \setminus \mathcal{O}} \left[3x(Q)^2 + a \right]$$

$$b' = b - 7 \sum_{Q \in G \setminus \mathcal{O}} \left[5x(Q)^3 + 3ax(Q) + b \right].$$

Note that this theorem gives a bijection between subgroups of G -up to isomorphism- and separable isogenies with domain E -up to isomorphism. This is a very powerful result, and it will be crucial in the protocol. However, note that the formula for $\phi(P)$ needs $O(|G|)$ group operations. In the protocol, for security reasons, the size of the kernel of the secret isogenies will be exponential, and therefore we will not be able to use this formula directly.

Observation 5. The first isomorphism theorem states that, for a morphism of groups $f : G \to H$ one has

$$G/ker(f) \cong Im(f).$$

In particular for an isogeny $\phi : E \to E'$ we have $E/ker(\phi) \cong E'$ because ϕ is exhaustive. Now let $G \subset E$ be a subgroup, ϕ the isogeny given by Vélu's formulas using G and E_G the corresponding curve $\phi(E)$. Then we have $E/G \cong E_G$. This is why we will usually talk about quotient groups when using Vélu, although these formulas give a curve isomorphic to this quotient group.

[7] When talking about isogenies, "up to isomorphism" will mean "up to composition with isomorphisms".

3.1.3 Isogenies of prime power degree

Thanks to Proposition 6, we know that a composition of e isogenies of degree l will be an isogeny of degree l^e. Let us briefly study this last type of isogeny. In particular, we will focus on separable isogenies of degree l^e for some prime l and positive e, because it is the case that we will face in the SIDH protocol, proving the following theorem.

Theorem 4. *Let l be a prime and $e \in \mathbb{Z}$ an integer with $e > 1$. Let $\phi : E \to E'$ be a separable isogeny of degree l^e. Then, ϕ can be written as*

$$\phi = \phi_e \circ \cdots \circ \phi_1$$

where ϕ_i is an isogeny of degree l for all $i \in \{1, \dots, e\}$.

Proof. We will first see a way to decompose the l^e-isogeny into a composition of an l-isogeny and a l^{e-1}-isogeny. Then, repeating the same argument the theorem will follow.

Since ϕ is separable, we know that $|ker(\phi)| = l^e$. Now, given that $ker(\phi)$ is a subgroup of E because ϕ is a morphism of groups, we can use Cauchy's theorem[8] to say that there is an element $P \in ker(\phi)$ of order l.

Now, since P has order l, the cyclic group $\langle P \rangle$ has order l as well, and Vélu's formulas give a unique -up to isomorphism- isogeny $\phi_1 : E \to E_1$ where $ker(\phi_1) = \langle P \rangle$, and thus $deg(\phi_1) = l$. Now, we want to find an isogeny $\psi : E_1 \to E'$ so that $\psi \circ \phi_1 = \phi$ (as can be seen in Figure 5).

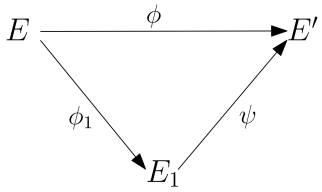


Figure 5: Commutative diagram of the decomposition of ϕ.

We know that kernels characterize isogenies up to isomorphism, thanks to Vélu's formulas. Hence, it is sufficient to find a ψ' such that $ker(\psi' \circ \phi_1) = ker(\phi)$. Therefore, we want to send $ker(\phi)$ to $\mathcal{O}_{E'}$ by $\psi' \circ \phi_1$, so we look at $\phi_1(ker(\phi))$, which is a subgroup of E_1, and we can use Vélu again to find $\psi' : E_1 \to E''$ with $ker(\psi') = \phi_1(ker(\phi))$. Then,

$$ker(\psi' \circ \phi_1) = \phi_1^{-1}(ker(\psi')) = \phi_1^{-1}(\phi_1(ker(\phi)).$$

[8] Let G be a finite group of order n. Cauchy's theorem states that for each prime p dividing n, there is an element $g \in G$ with order p.

Now we need to be careful, because $\phi_1^{-1}(\phi_1(ker(\phi))$ is not necessarily $ker(\phi)$. Instead,

$$\phi_1^{-1}(\phi_1(ker(\phi)) = ker(\phi) + ker(\phi_1).$$

However, in our case, since $ker(\phi_1) = \langle P \rangle \subset ker(\phi)$, we have that $ker(\psi' \circ \phi_1) = ker(\phi)$.

Therefore, $E'' \cong E'$ and $\psi' \circ \phi_1 = \alpha \circ \phi$ for some isomorphism α, as can be seen in Figure 6.

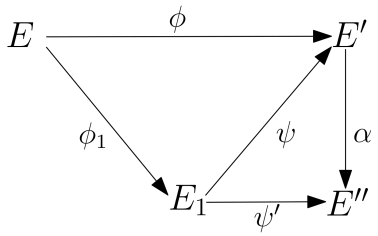


Figure 6

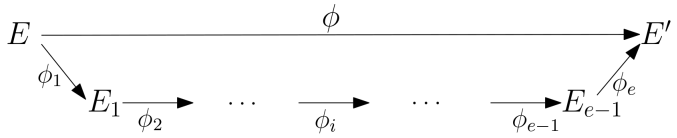
Now we can define ψ as $\psi = \alpha^{-1} \circ \psi' : E_1 \to E'$. It only remains to see that $deg(\psi) = l^{e-1}$, but that is easy to check using Proposition 6 as follows.

$$\phi = \psi \circ \phi_1 \Rightarrow deg(\phi) = deg(\psi \circ \phi_1) = deg(\phi_1)deg(\psi)$$

and using that we know the degrees of ϕ and ϕ_1 we can state

$$l^e = l \cdot deg(\psi) \Rightarrow deg(\psi) = l^{e-1}.$$

We now observe that the same argument can be used to decompose the l^{e-1}-isogeny ψ in an l-isogeny and a l^{e-2}-isogeny and so on, until we reach l^1, and this yields the desired chain $\phi = \phi_e \circ \cdots \circ \phi_1$.



□

Observation 6. More generally, if the prime factorization of the degree of an isogeny is $l_1^{r_1} ... l_n^{r_n}$ then it can be decomposed in $r_1 + \cdots + r_n$ isogenies of prime degree. It should be clear how the proof above can be adapted to fit this generalization. However, for our purposes it suffices with this theorem.

3.2 Supersingular Isogeny Diffie Hellman (SIDH)

3.2.1 Simplified SIDH

Before we jump to the full version of supersingular isogeny Diffie Hellman, let us study a simplified version that shows the main idea of the scheme.

In SIDH, instead of operating inside an elliptic curve using the group operation as in the regular Diffie Hellman key agreement we will operate over elliptic curves using isogenies. In fact, we will operate over isomorphism classes of elliptic curves, which are represented by j-invariants.

The main idea is the following. Alice and Bob will both choose a random point and its cyclic subgroup of the public supersingular elliptic curve E, namely $\langle a \rangle$ and $\langle b \rangle$, and calculate $E/\langle a \rangle$ and $E/\langle b \rangle$, respectively (in fact, they calculate curves isomorphic to $E/\langle a \rangle$ and $E/\langle b \rangle$), along with the corresponding isogenies using Vélu's formulas. They will send to the other party their quotient group and they will be able to calculate $E/\langle a, b \rangle$ up to isomorphism, sharing then a secret key $j(E/\langle a, b \rangle)$.

But Bob will have the following problem (and Alice the analogous): knowing $E/\langle a \rangle$ and $\langle b \rangle$ is not enough to calculate $E/\langle a, b \rangle$, because $\langle b \rangle$ is not in $E/\langle a \rangle$ but in E. We could solve this saying that Alice sends the image of $\langle b \rangle$ by the isogeny $\phi_A : E \to E/\langle a \rangle$, but that is where the security of the protocol lies. The hard problem that a passive attacker would have to face is that given two isogenous curves, it is hard to find the isogeny between them. Therefore, we would like to give as less information as possible about these isogenies.

Let us introduce a change: instead of using random points $\langle a \rangle$ and $\langle b \rangle$, we will choose two points for Alice, P_A and Q_A and two for Bob, P_B and Q_B and make them public. Then, they will use a random combination of their points to define their secret subgroups. However, these points will be taken such that $\langle P_A, Q_A \rangle = E[l_A]$ and $\langle P_B, Q_B \rangle = E[l_B]$ for some primes $l_A \neq l_B$. The reason why we can find such points is that from Proposition 2 we know that $E[l_A] \cong \mathbb{Z}/l_A\mathbb{Z} \times \mathbb{Z}/l_A\mathbb{Z}$ and $E[l_B] \cong \mathbb{Z}/l_B\mathbb{Z} \times \mathbb{Z}/l_B\mathbb{Z}$, and the reason why we do so will become clear later on.

With this change, now instead of sending the image of $\langle b \rangle$ by the isogeny, Alice will send the image of Bob's generators and, with that, Bob will be able to calculate the image of his random combination (because isogenies are morphisms).

Let us show a sketch of how the key exchange works. We will be omitting some crucial facts like what is the ground field, how are the primes l_A, l_B chosen or what are the restrictions on the random integers that both parties compute, but all these will be explained in forecoming sections when we see the full protocol and prove its correctness. For now, assume that

- E is an elliptic curve defined over a field $\mathbb{F}_q$.

- Integers l_A, l_B are fixed different primes and $gcd(l_A, q) = gcd(l_B, q) = 1$.
- Points $P_A, Q_A, P_B, Q_B \in E$ are chosen such that $\langle P_A, Q_A \rangle = E[l_A]$ and $\langle P_B, Q_B \rangle = E[l_B]$.

We now explain a sketch of how the protocol works from Alice's point of view.

1. Alice chooses two random integers $n_A, m_A \in \mathbb{Z}/l_A\mathbb{Z}$ and computes $R_A := n_A P_A + m_A Q_A$.
2. Let A be the cyclic group $\langle R_A \rangle$. Vélu allows her to compute unique -up to isomorphism- curve E_A and isogeny ϕ_A such that

$$\phi_A : E \to E_A \text{ and } ker(\phi_A) = A.$$

3. Evaluates Bob's points P_B, Q_B with ϕ_A and sends these values and her curve to Bob.
4. Receives Bob's curve E_B and evaluation of P_A, Q_A on Bob's isogeny $\phi_B(P_A), \phi_B(Q_A)$. Now she can compute $\phi_B(R_A)$ as

$$n_A\phi_B(P_A) + m_A\phi_B(Q_A) = \phi_B(n_A P_A + m_A Q_A) = \phi_B(R_A).$$

5. Again, taking the group $\langle \phi_B(R_A) \rangle \subset E_B$ Vélu's formulas determine unique -up to isomorphism- elliptic curve and isogeny E_{BA} and ϕ'_A with

$$\phi'_A : E_B \to E_{BA} \text{ and } ker(\phi'_A) = \langle \phi_B(R_A) \rangle.$$

Finally, the shared secret key is $j(E_{BA})$.

Bob will act analogously, ending in a curve E_{AB} and computing $j(E_{AB})$ as the secret key. For this protocol to work properly, we need the following theorem to hold.

Theorem 5. *In the previous protocol, $E_{AB} \cong E_{BA}$.*

Proof. This is a particular case of Theorem 6, which we will prove later on. □

3.2.2 Isogenies between supersingular elliptic curves

Let l be an integer such that $gcd(l, p) = 1$. We are interested in how many isogenies -up to isomorphism- are there $\phi : E \to E'$ such that $deg(\phi) = l$. To count them we will use the fact that for each subgroup G we have a unique -up to isomorphism- isogeny with G as kernel. In other

words, what we have is a bijection between subgroups of E and (separable) isogenies $\phi : E \to E$. In particular we have a bijection between subgroups of E of order l and isogenies $\phi : E \to E'$ of degree l.

Counting the subgroups of order l in E may not be easy. Let us try to fix that problem. Let $\phi : E \to E'$ be an isogeny with $deg(\phi) = l$. We know that there exists the dual isogeny $\hat{\phi}$ and $\hat{\phi} \circ \phi = [l]$. Now let $P \in ker(\phi)$ be a point of the kernel of ϕ. We have that

$$\phi(P) = \mathcal{O} \Rightarrow \left(\hat{\phi} \circ \phi\right)(P) = \mathcal{O} \Rightarrow [l](P) = \mathcal{O} \Rightarrow P \in E[l].$$

That means that the kernel of an isogeny ϕ of degree l is always in $E[l]$, namely $ker(\phi) \subset E[l]$. This will help us count the number of isogenies, because now we only have to count the subgroups of order l in $E[l]$, and we know that $E[l] \cong \mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$. Thus, we only need to count how many subgroups of order l are there in $\mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$.

Proposition 10. *Let l be a prime integer. There are exactly $l+1$ subgroups of order l in $\mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$.*

Proof. Let us count the subgroups of order l. First of all, note that any such group is cyclic, because it has prime order. Consider, for each pair $(x, y) \in \mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}$, the cyclic group $\langle (x, y) \rangle$. Since l is prime, this subgroup has order l if and only if $(x, y) \neq (0, 0)$.

Then, we have counted $|(\mathbb{Z}/l\mathbb{Z} \times \mathbb{Z}/l\mathbb{Z}) \backslash \{0\}| = l^2 - 1$ subgroups of order l. However, it is clear that we have counted each subgroup multiple times with this process. In fact, in a cyclic group of prime order l, each element has order l except for 0, meaning that each non-zero element of this group is a generator. Hence, we have counted each subgroup of order l a total of $l-1$ times. Finally, the number of subgroups of order l is $\frac{l^2-1}{l-1} = l + 1$. □

Corollary 1. *Given an elliptic curve E and a prime l, there are exactly $l + 1$ isogenies $\phi : E \to E'$ such that $deg(\phi) = l$.*

The importance of this fact is clear. In SIDH, the secrecy of the isogenies that both parties compute is essential to guarantee security. Therefore, we need the possibilities for those isogenies to be exponential, and what we just studied shows that this requires the degree of the isogeny to be exponential.

3.2.3 Supersingular isogeny graph

The idea of each supersingular elliptic curve having $l + 1$ isogenies originating from it leads to the concept of the supersingular isogeny graph. Consider the set of supersingular elliptic curves

isomorphism classes (or simply the set of supersingular j-invariants) as vertices and the l-isogenies between them as edges. This graph is called *supersingular isogeny graph*. In fact, what we have seen in the previous subsection is that this graph is $l+1$-regular[9].

A question may arise at this point: these j-invariants represent isomorphism classes of supersingular elliptic curves, but these curves may not be defined over $\mathbb{F}_p$. Not only we do not know where do we have to look to find all these classes, but we do not even know if there are a finite many of them. What is sure is that they all are defined over the algebraic closure $\overline{\mathbb{F}_p}$. Also, it turns out that given a prime p, all the isomorphism classes of supersingular curves have a representative in $\mathbb{F}_{p^2}$. In other words, each supersingular elliptic curve defined over $\overline{\mathbb{F}_p}$ is isomorphic to one defined over $\mathbb{F}_{p^2}$. That is why here we will always work in $\mathbb{F}_q = \mathbb{F}_{p^2}$.

In the same way that we counted isogenies (edges in the graph), one can also count j-invariants (vertices in the graph). Let us get straight to the result in this case. The number of j-invariants of curves defined over $\mathbb{F}_{p^2}$, usually represented by S_{p^2} is

$$S_{p^2} = \left\lfloor \frac{p}{12} \right\rfloor + \delta + \epsilon \text{ where } \delta = \begin{cases} 0 \text{ if } p \equiv 1 \mod(3) \\ 1 \text{ if } p \equiv 2 \mod(3) \end{cases} \quad \text{and } \epsilon = \begin{cases} 0 \text{ if } p \equiv 1 \mod(4) \\ 1 \text{ if } p \equiv 3 \mod(4) \end{cases}.$$

A study of supersingular isogeny graphs along with a proof of this result can be found in [KZ98]. For our purposes, what matters is that there are about $\lfloor \frac{p}{12} \rfloor$ isomorphism classes of supersingular elliptic curves, hence that many vertices on the graph.

Observation 7. Note that we chose the graph not to be directed although we defined the edges as isogenies from a curve to another. This is because we know that for each isogeny $\phi : E \to E'$ there exists the dual isogeny $\hat{\phi} : E' \to E$, and thus the existence of an edge from $j(E)$ to $j(E')$ implies the existence of another edge from $j(E')$ to $j(E)$.

With this notion, let us study what happened in the simplified version of SIDH. Alice chose a random l_A-isogeny, which is equivalent to taking a random step in the l_A-isogeny graph, and so did Bob in the l_B-isogeny graph. These graphs share the set of vertices, and in our protocol the two paths start in the same vertex, represented by $j(E)$, and end in the same vertex, represented by $j(E/\langle R_A, R_B \rangle)$. In the full version, what will happen is that these random steps will become random walks. Alice will compute a random $l_A^{e_A}$-isogeny in the l_A-isogeny graph, for a small prime l_A and a large enough e_A, which in fact will be the composition of e_A isogenies of degree l_A.

[9] Remember that a graph is k-regular if each vertex has exactly k neighbors (counting itself in case of loops).

3.2.4 SIDH

We are now able to look at the full SIDH protocol, detailed below. The protocol is due to David Jao and Luca de Feo and the original paper is [JDF11]. However, an extended version by Jao, de Feo and Plût can be found in [DFJP14] containing all information of [JDF11], so we will sometimes cite the extended version as the original paper as well.

Public parameters

Two different small primes l_A, l_B and two integers e_A, e_B such that $l_A^{e_A} \approx l_B^{e_B}$.

A prime p of the form $p = l_A^{e_A} l_B^{e_B} d \pm 1$ for a small factor d.

A supersingular elliptic curve E over $\mathbb{F}_q$ for $q = p^2$ with $(l_A^{e_A} l_B^{e_B} d)^2$ points.

Points P_A, Q_A that generate $E[l_A^{e_A}]$.

Points P_B, Q_B that generate $E[l_B^{e_B}]$.

Alice	**Bob**
Chooses two random integers $n_A, m_A \in \mathbb{Z}/l_A^{e_A}\mathbb{Z}$ not both divisible by l_A and computes $R_A := n_A P_A + m_A Q_A$. Let A be the cyclic group $\langle R_A \rangle$. Vélu's formulas determine unique -up to isomorphism- curve E_A and isogeny ϕ_A with	Chooses two random integers $n_B, m_B \in \mathbb{Z}/l_B^{e_B}\mathbb{Z}$ not both divisible by l_B and computes $R_B := n_B P_B + m_B Q_B$. Let B be the cyclic group $\langle R_B \rangle$. Vélu's formulas determine unique curve E_B and isogeny ϕ_B with
$\phi_A : E \to E_A$ and $ker(\phi_A) = A$.	$\phi_B : E \to E_B$ and $ker(\phi_B) = B$.
Evaluates Bob's points P_B, Q_B with ϕ_A and sends these values and her curve to Bob.	Evaluates Alice's points P_A, Q_A with ϕ_B and sends these values and his curve to Alice.
$\xrightarrow{E_A, \phi_A(P_B), \phi_A(Q_B)}$	$\xleftarrow{E_B, \phi_B(P_A), \phi_B(Q_A)}$
Receives Bob's curve E_B and evaluation of P_A, Q_A on Bob's isogeny $\phi_B(P_A), \phi_B(Q_A)$. Now she can compute $\phi_B(R_A)$ as	Receives Alice's curve E_A and evaluation of P_B, Q_B on Alice's isogeny $\phi_A(P_B), \phi_A(Q_B)$. Now he can compute $\phi_A(R_B)$ as
$n_A\phi_B(P_A) + m_A\phi_B(Q_A) = \phi_B(n_A P_A + m_A Q_A)$.	$n_B\phi_A(P_B) + m_B\phi_A(Q_B) = \phi_A(n_B P_B + m_B Q_B)$.
Again, taking the group $\langle \phi_B(R_A) \rangle \subset E_B$ Vélu's formulas determine unique -up to isomorphism- elliptic curve and isogeny E_{BA} and ϕ'_A with	Again, taking the group $\langle \phi_A(R_B) \rangle \subset E_A$ Vélu's formulas determine unique -up to isomorphism- elliptic curve and isogeny E_{AB} and ϕ'_B with
$\phi'_A : E_B \to E_{BA}$ and $ker(\phi'_A) = \langle \phi_B(R_A) \rangle$.	$\phi'_B : E_A \to E_{AB}$ and $ker(\phi'_B) = \langle \phi_A(R_B) \rangle$.
Finally, the shared secret key is $j(E_{BA})$.	Finally, the shared secret key is $j(E_{AB})$.

Theorem 6. *In the previous protocol, $E_{AB} \cong E_{BA}$.*

Proof. Let $\varphi : E \to E_{AB}$ the composition $\varphi = \phi'_B \circ \phi_A$ and $\psi : E \to E_{BA}$ the composition $\psi = \phi'_A \circ \phi_B$. The diagram in Figure 7 represents the situation in the scheme:

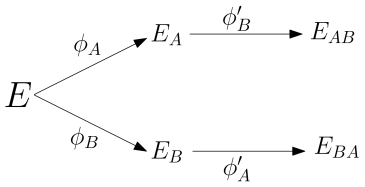


Figure 7

In the construction of the protocol we imposed that R_A and R_B were mapped to the neutral element $\mathcal{O}$ by both φ and ψ. In fact we imposed that the cyclic subgroups they generate, $\langle R_A \rangle$ and $\langle R_B \rangle$, were mapped to $\mathcal{O}$, which is necessary because isogenies are group morphisms. With this, we know that $\langle R_A, R_B \rangle \subseteq ker(\varphi)$ and $\langle R_A, R_B \rangle \subseteq ker(\psi)$. We will now show that these inclusions are, in fact, equalities.

Let us show that $ker(\varphi) = \langle R_A, R_B \rangle$, and analogously the same will hold for ψ. Note that

$$|ker(\varphi)| = deg(\varphi) = deg(\phi'_B \circ \phi_A) = deg(\phi_A) deg(\phi'_B).$$

First we observe that $deg(\phi_A) = |ker(\phi_A)| = ord(R_A)$ with $R_A = n_A P_A + m_A Q_A$. Now, the condition that n_A, m_A can not be both divisible by l_A becomes crucial in order to state that $ord(R_A) = l_A^{e_A}$. Let us prove this. Assume $\alpha R_A = \alpha n_A P_A + \alpha m_A Q_A = \mathcal{O}$ for some $\alpha \leq l_A^{e_A}$ and let us prove that α has to be $l_A^{e_A}$. Note that $\alpha n_A P_A + \alpha m_A Q_A = \mathcal{O}$ holds if and only if both $\alpha n_A P_A$ and $\alpha m_A Q_A$ are $\mathcal{O}$. This is true because $\langle P_A, Q_A \rangle \cong \mathbb{Z}/l_A^{e_A}\mathbb{Z} \times \mathbb{Z}/l_A^{e_A}\mathbb{Z}$ and hence $\langle P_A \rangle \cap \langle Q_A \rangle = \{\mathcal{O}\}$.

Therefore, we have that if $\alpha R_A = \mathcal{O}$ then

$$\begin{cases} \alpha n_A P_A = \mathcal{O} \\ \alpha m_A Q_A = \mathcal{O} \end{cases} \Rightarrow \begin{cases} ord(P_A) = l_A^{e_A} \text{ divides } \alpha n_A \\ ord(Q_A) = l_A^{e_A} \text{ divides } \alpha m_A \end{cases}.$$

Now, without loss of generality assume that n_A is not divisible by l_A. Then $l_A^{e_A}$ divides α, hence α must be $l_A^{e_A}$. Analogously, the same is true if m_A is not divisible by l_A. Therefore we can state that $ord(R_A) = l_A^{e_A} = deg(\phi_A)$.

We now need to calculate $deg(\phi'_B)$. Note that $deg(\phi'_B) = |ker(\phi'_B)| = |\langle \phi_A(R_B) \rangle|$ and

$$\langle \phi_A(R_B) \rangle = \{n\phi_A(R_B) | n \in \mathbb{Z}\} = \{\phi_A(nR_B) | n \in \mathbb{Z}\} = \phi_A(\langle R_B \rangle).$$

We will prove that $ord(\phi_A(R_B)) = ord(R_B)$ by proving that $\alpha\phi_A(R_B) = \mathcal{O}$ if and only if $\alpha R_B = \mathcal{O}$. First, if $\alpha R_B = \mathcal{O}$ then

$$\alpha\phi_A(R_B) = \phi_A(\alpha R_B) = \phi_A(\mathcal{O}) = \mathcal{O}.$$

In the other direction, if $\alpha\phi_A(R_B) = \mathcal{O}$ then $\phi_A(\alpha R_B) = \mathcal{O}$, so $\alpha R_B \in ker(\phi_A) = \langle R_A \rangle$ and thus $\langle \alpha R_B \rangle$ is a subgroup of $\langle R_A \rangle$. But αR_B is also a subgroup of $\langle R_B \rangle$, and since $ord(R_A) = l_A^{e_A}$ and $ord(R_B) = l_B^{e_B}$ are relatively primes, we have that $\langle \alpha R_B \rangle = \{\mathcal{O}\}$ and therefore $\alpha R_B = \mathcal{O}$.

With this we can conclude that

$$deg(\phi'_B) = |ker(\phi'_B)| = ord(\phi_A(R_B)) = ord(R_B) = l_B^{e_B}$$

and, finally, we can state that $deg(\varphi) = l_A^{e_A} l_B^{e_B}$.

Now, let us see that $|\langle R_A, R_B \rangle| = l_A^{e_A} l_B^{e_B}$ as well. We will use the following well known result (a corollary of the so-called *second isomorphism theorem*): for two finite subgroups H, K of a group G with H or K being normal[10], one has $|HK| = \frac{|H||K|}{|H \cap K|}$ (with multiplicative group notation).

In our case $\langle R_A \rangle$ and $\langle R_B \rangle$ have trivial intersection because they are cyclic groups and their orders are relatively primes. Therefore, we have that

$$|\langle R_A, R_B \rangle| = |\langle R_A \rangle + \langle R_B \rangle| = |R_A||R_B| = l_A^{e_A} l_B^{e_B}.$$

Since $\langle R_A, R_B \rangle \subseteq ker(\varphi)$ and $|\langle R_A, R_B \rangle| = |ker(\varphi)|$, we conclude that $\langle R_A, R_B \rangle = ker(\varphi)$. Analogously, one can see that $\langle R_A, R_B \rangle = ker(\psi)$ and, thus, $ker(\varphi) = ker(\psi)$.

Now it only remains to use the first isomorphism theorem and we have

$$E_{AB} = Im(\varphi) \cong E/ker(\varphi) = E/ker(\psi) \cong Im(\psi) = E_{BA}.$$

□

Although in the protocol description may not look like there is a substantial change from what we saw in 3.2.1, a lot has changed both theoretically and in practice. First, changing random steps for random walks increased the size of the kernels exponentially. However a new problem arises: Vélu's formulas give a way to compute the isogenies, but the complexity of the computations grows with as the size of the kernel, meaning that $\mathcal{O}(l^e)$ operations are required to compute an isogeny of degree l^e. Therefore, now that we have exponentially big kernels this may look like a problem to

[10] A subgroup H of a group G is called *normal* if $gN = Ng$ for all $g \in G$. In our case, all groups are normal because elliptic curves are abelian.

us. We will see how Alice computes the images of P_B, Q_B in 3.4.3, where Theorem 4 will play an important role.

Observation 8. In the SIDH protocol, the isogenies computed by Alice are ϕ_A and ϕ'_A, and Bob's are ϕ_B, ϕ'_B. This may be confusing for a first time reader, because the idea of getting to the same endpoint from a common starting point may lead to think that party A computes one path and party B computes another. In our case, one may think that Alice is doing the upper path $E \to E_A \to E_{AB}$ and Bob is doing $E \to E_B \to E_{BA}$. However, they are actually "exchanging" their roles in the middle step, as can be seen in Figure 8 below.

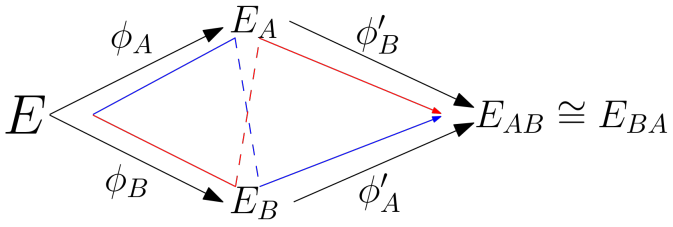


Figure 8: The blue line represent Alice's computations or "path", and the red one represents Bob's.

Observation 9. It is worth mentioning that this protocol is not a particular case of the group action Diffie Hellman we saw in 1.5.3. It may look like isogenies are acting on the set of j-invariants in a commutative way to reach a shared secret j-invariant. However, isogenies do not commute because, in fact, once we compute an isogeny we are not in the same curve anymore, meaning that it does not even make sense to think of commutativity.

3.2.5 Public key encryption from SIDH

The SIDH protocol is an unauthenticated key agreement. This means that an active attacker could interfere in the conversation between Alice and Bob, for example changing the message that one party receives by one of his or her choice. This is known as a *man-in-the-middle* attack, and clearly SIDH is vulnerable to them. In [Gal18], Galbraith studies possible SIDH-based authenticated key agreements (AKE) and proposes several AKE protocols.

SIKE is a key encapsulation protocol based on SIDH. In January of 2003, with the horizon of large quantum computers getting closer, NIST initialized a project for post-quantum cryptography standardization and SIKE is one of the candidates that passed to the second round of the selection[11]. We do not give details of the SIKE protocol because it has many technical details to guarantee fast implementations and additional security, and we are for now mostly working on the theoretical

[11]See the full list of round 2 submissions at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

approach. For all details, see the NIST submission [CCH+19] and also implementations can be found in the SIKE website.

As in many key agreement protocols, one can easily derive a public key encryption scheme from the SIDH key exchange. The idea is to use the secret shared key to encrypt and decrypt as in a symmetric key encryption scheme. Let us show an example (which is defined in the original paper [JDF11] and attacked in [GPST16]) with one-time pad (see Example 2) as symmetric key scheme. We will be using the same notation as in the SIDH protocol.

1. *Public parameters.* The public parameters are chosen as in SIDH and, additionally, a hash function H is also agreed as a public parameter. Then we have as public parameters

$$p, \mathbb{F}_{p^2}, E, P_A, Q_A, P_B, Q_B, l_A, e_A, l_B, e_B, H.$$

2. *Keys generation.* Alice generates (n_A, m_A) as in the protocol. This pair will be her secret key.

   She then computes her first step of the protocol and publishes $(\phi_A(P_B), \phi_A(Q_B), E_A)$. This will be her public key.

3. *Encryption.* Bob wants to cipher a message m. First, he chooses n_B, m_B as in the protocol and he computes the secret shared key k. He computes $H(k)$ and $c := m \oplus H(k)$.

   The encrypted message he sends to Alice is the ciphered m, which we called c, and the information that Alice needs to compute the secret key and then decrypt c, namely

$$Enc(m) = (E_B, \phi_B(P_A), \phi_B(Q_A), c).$$

4. *Decryption.* Upon receiving Bob's message, Alice can compute the secret shared key k as in the SIDH protocol and then recover the message as $m = c \oplus H(k)$.

3.3 Security

To study the security of SIDH and the attacks that have been proposed against this protocol and other isogeny-based schemes, first we should define some basic problems for the attacker to solve. Let us first define a basic and general one, which we will call the *pure isogeny problem*.

Problem 3. Pure isogeny problem. Given two elliptic curves E, E' over $\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = |E'(\mathbb{F}_p)|$, find an isogeny $\phi : E \to E'$.

Alternatively, one can also give the information of the degree of the isogeny. In the SIDH case that is also a known parameter so let us define a slightly different problem.

Problem 4. Given two l^e-isogenous elliptic curves E, E' over $\mathbb{F}_p$, find an l^e-isogeny $\phi : E \to E'$.

Obviously, an attacker that could solve any of these problems would easily break the SIDH protocol, recovering the isogenies and thus the kernels and the key. The good news is that no classical or quantum attack has been published that solves that problem in less than exponential time. But this is too general. Let us define the problem that a passive attacker would have to face in SIDH in order to find the isogeny, with the information being the public parameters of SIDH and the information that Alice shares with Bob. We will assume without loss of generality that the adversary attacks Alice's isogeny.

Problem 5. Computational Supersingular Isogeny problem (CSSI). Given

$$q, \mathbb{F}_q, E, l_A, l_B, e_A, e_B, P_A, Q_A, P_B, Q_B, \phi_A(P_B), \phi_A(Q_B) \text{ and } E_A$$

compute the isogeny $\phi_A : E \to E_A$.

It is clear that computing the isogeny is equivalent to computing a generator of A, so we could have also defined it the other way. Also, note that we do not give the attacker the information that Bob shares to Alice. In fact, that information should be completely irrelevant for him to compute Alice's isogeny, because Bob's random choice of integers n_B, m_B (and, thus, of isogeny) is independent from Alice's computations.

However, none of this problems gives enough security. It is good (and necessary) that an attacker can not find the isogeny between E and E_A, but what we really want to secure is the shared key $j(E_{AB}) = j(E_{BA})$. What if the attacker could compute that value without discovering the secret isogeny? That would be a problem as well. In fact, what we would like is that, for an attacker, the secret shared key is as similar to a random value as possible. In other words, we want the attacker to not be able to distinguish between the secret key and a random value. Let us formally define this concept with a game.

Game 2. Indistinguishability of SIDH.

We will be using the notation from subsection 3.2.4.

In this game there will be two parties: the *challenger* and the *attacker*. The game proceeds as follows:

1. First, the challenger generates the public parameters of a SIDH protocol, namely

$$PublicParams = (\mathbb{F}_q, E, l_A, e_A, P_A, Q_A, l_B, e_B, P_B, Q_B).$$

2. He computes a key k_1 using the SIDH protocol acting as two honest parties[12] Alice and Bob and also the information that they exchange

$$ExchangedInfo = (\phi_A(P_B), \phi_A(Q_B), E_A, \phi_B(P_A), \phi_B(Q_A), E_B).$$

3. Then, he generates a random bit $b \in \{0, 1\}$.

   - If $b = 0$, he computes a random value k_0 in the set of possible supersingular j-invariants of curves over $\mathbb{F}_q$. Then, he sends the *PublicParams*, *ExchangedInfo* and this value k_0 to the attacker.
   - If $b = 1$, he sends *PublicParams*, *ExchangedInfo* and the key k_1 to the attacker.

4. The attacker receives the public parameters and the value k, and then in polynomial time he outputs a bit $b^* \in \{0, 1\}$.

5. The attacker wins if and only if $b = b^*$.

To the date, no classical or quantum polynomial or subexponential algorithm is known that wins this game with non-negligible advantage.

3.3.1 Auxiliary points

A natural question about the security of SIDH is how much information does an attacker obtain from $\phi_A(P_B, Q_B)$ and $\phi_B(P_A, Q_A)$. One important observation is that not only we are giving the attacker the images of some particular points, but the image by ϕ_A of the whole $l_B^{e_B}$-torsion group and the image by ϕ_B of the whole $l_A^{e_A}$-torsion group. Let us explain why this is true.

Remember that P_A and Q_A generate $E[l_A^{e_A}]$. Thus, each point R of $E[l_A^{e_A}]$ can be written as $R = nP_A + mQ_A$ for some n, m. That means that the image of R by ϕ_B is

$$\phi_B(R) = \phi(nP_A + mQ_A) = n\phi_B(P_A) + m\phi_B(Q_A).$$

[12] By honest parties we mean parties that follow the specification as defined. For example, in SIDH a dishonest Alice could choose n_A and m_A in a specific way so that she can get information about Bob's secret subgroup, or even use a specific isogeny of her choice without following the protocol. Hence, in this case what we mean is that n_A, m_A, n_B, m_B as chosen randomly (in their respective possible values) and the following process is done as described in the specification.

All this was clear, but the problem comes to find the integers n, m. In fact, finding this numbers reminds us to the discrete logarithm problem, for which we have a polynomial quantum algorithm. Actually, since we are working in a group of order $l_A^{2e_A}$ or $l_B^{2e_B}$, all the prime factors are small and the discrete logarithm is easy even in classical computation thanks to the Pohlig-Hellman algorithm [PH78], and polynomial algorithms exist to compute this 2-dimensional logarithm (see the generalization of Pohlig-Hellman algorithm in [Tes99]).

Petit's work [Pet17] shows a way to exploit the auxiliary points given. Assuming some additional knowledge he shows a polynomial time attack to recover the secret isogeny. In SIDH, the attacker is revealed the image of $E[l_B^{e_B}]$ by ϕ_A, which is a $l_A^{e_A}$-isogeny. One of Petit's assumptions is the image of $E[N_2]$ by the N_1-isogeny with N_2 significantly larger than N_1. This is not the case of SIDH, where $N_1 = l_A^{e_A} \approx l_B^{e_B} = N_2$. He also assumes some non-scalar endomorphisms over the original curve to be known or easy to compute. The main idea of the attack is to consider $\psi := \theta_1\phi + \theta_2$ where ϕ is the secret isogeny and θ_1, θ_2 are known endomorphisms such that $deg(\psi) = N_1' N_2$ for some N_1'. Then one can decompose $\psi = \psi_{N_1'}\psi_{N_2}$ as a product of a N_1'-isogeny and a N_2-isogeny. Using that the image of ϕ in $E[N_2]$ is known, one can evaluate ψ in $E[N_2]$ and then he shows how to find $ker(\psi_{N_2})$ and, thus, ψ_{N_2}. He then uses a meet-in-the-middle attack to recover $\psi_{N_1'}$ and finally computes $\phi = \theta_1^{-1}(\psi_{N_1'}\psi_{N_2} - \theta_2)$. He also shows a way to use this technique without the given endomorphisms, adding some extra conditions and using scalar multiplications as endomorphisms.

3.3.2 Reusing keys

Here we talk about an important weakness of SIDH. When talking about a public encryption scheme it is natural to ask how long can the keys be reused before we lose too much security. Focusing on Alice, this can be asked in the SIDH key exchange talking about her secret combination (n_A, m_A). It turns out that the protocol loses security when Alice keeps reusing the same random combination (n_A, m_A) and, therefore, the same secret subgroup $\langle R_A \rangle$ and isogeny. Galbraith, Petit, Shani and Ti [GPST16] show how to attack the protocol in an active adaptive attack, acting as a dishonest Bob. Let (n_A, m_A) be the static pair that Alice is using. Their only assumption is to have access to an oracle

$$O(E, R, S, E') = \begin{cases} 1 \text{ if } j(E') = j(E/\langle n_A R + m_A S \rangle) \\ 0 \text{ otherwise.} \end{cases}$$

This assumption is realistic. If E is the curve computed by Bob ($E = E_B \cong E/\langle R_B \rangle$ in the SIDH between honest parties), then the oracle query is just a validation of the secret shared key being correct, which is reasonable to exist in order to guarantee authentication in the messages exchanged

after the key agreement. We do not show how the attack works, because it involves concepts that we have not explained here. The attack requires about $\frac{1}{2}log_2(p)$ interactions with Alice.

Countermeasures have been discussed to avoid this type of attacks against SIDH. Some of them were found vulnerable to the GPST, however there are countermeasures that prevent this attack, checking that the auxiliary points sent by Bob correspond to a honest user. In [GPST16] they discuss the usability of this countermeasures and remark that the ones that avoid this attack require an important amount of time in validations.

It is worth mentioning another attack defined in the same article [GPST16]. In this attack, a honest key-agreement takes places between Alice and Bob with Eve eavesdropping the session, hence obtaining E_A, $\phi_A(P_B)$, $\phi_A(Q_B)$, $E_B\phi_B(P_A)$, $\phi_B(Q_A)$ and then Eve acts as Bob to recover the shared j-invariant choosing her secret curve as $E_B/\langle R_C\rangle$ for some specific $\langle R_C\rangle$. In the attack they assume to obtain some partial information of the j-invariants computed by Alice, which could be obtained for example via a side-channel attack[13], and hence they obtain information on $j(E/\langle R_A, R_C\rangle)$ which they use to recover information from $j(E_{AB})$.

Of course, it is important to know what to do we mean by "partial information". They mount the attack based on two possible oracles for this partial information that we are mentioning.

Before we introduce them, note that the field $\mathbb{F}_{p^2}$ is an extension of $\mathbb{F}_p$ for some α with $\alpha^2 + A\alpha + B = 0$ with $A, B \in \mathbb{F}_p$. Then we can write $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$ and if $x \in \mathbb{F}_{p^2}$ one can write it as $x = a + b\alpha$ for some $a, b \in \mathbb{F}_p$. We will say that a and b are the *components* of x. One of the oracles they define return one of the two components of the j-invariant that Alice computes. The other one returns only the most significant bits of one of the components. To know how reasonable are this oracles we should do a completely different study, focusing on the side-channel weaknesses of the SIDH, which is not in the route of this work.

The result of the attack assuming to have the more powerful oracle (the one that outputs a full component) is that with only two queries they are able to recover the secret key with probability at least $\frac{1}{18}$ in the worst case (when none of the two components given by the oracle is the α-component).

3.3.3 Meet in the middle

To the date, the best known classical or quantum attack to the protocol using no extra information is a generic approach -meaning that it barely uses any property of the elliptic curves or isogenies-

[13]Informally, a *side-channel attack* does not attack the structure of the protocol itself, instead it uses physical observations and information on, such as detecting voltage spikes, calculating computation times or similar techniques on the honest users devices. For example, if a protocol requires significantly small computing time to encrypt or decrypt for some specific keys, a side-channel attack could be used to obtain some information.

that takes exponential time both in classical and quantum computers. The abstract attack is defined against the generic claw search problem. In this section we will assume that, when trying to recover the secret isogenies of SIDH, we do so against Alice's secret isogeny.

Problem 6. Let $f : A \to C$ and $g : B \to C$ be two functions with $|A| \leq |B|$. The *claw-search* finding problem is to find a pair $(a, b) \in A \times B$ such that $f(a) = g(b)$.

As mentioned in [CLN+19], the claw-search problem can be solved in $O(|A| + |B|)$ time and $O(|A|)$ space in classical computation, and Tani [Tan07] shows how to solve it in a quantum computer with time

$$O((|A||B|)^{1/3}) \text{ if } |A| \leq |B| < |A|^2$$
$$O(|B|^{1/2}) \text{ if } |B| \geq |A|^2.$$

Using the notation from the SIDH protocol, to reduce CSSI to the claw-search, let A be the order-$l_A^{e_A/2}$ cyclic subgroups of $E[l_A^{e_A}]$ and B be the order-$l_A^{e_A/2}$ cyclic subgroups of $E_A[l_A^{e_A}]$. For $G \in A$ define $f(G) = j(E/G)$ where $j(E/G)$ is obtained using Vélu's formulas. Similarly, for $H \in B$ define $g(H) = j(E_A/H)$. If we can find G, H such that $f(G) = g(H)$ then we can yield an isogeny from E to E_A as we will see step by step. First, we define the abstract idea of the attack.

1. Compute all possible paths in the $l^{e/2}$ supersingular isogeny graph starting from the vertex $j(E)$ and store the resulting edge and vertex.
2. Compute paths in the graph starting from $j(E')$ until a collision is find with one of the stored values.

Now let E, E' be supersingular elliptic curves and assume we want to find an isogeny $\phi : E \to E'$ of degree l^e. In the SIDH we would be trying to recover a $l_A^{e_A}$-isogeny, so one can think of l as l_A, e as e_A and E' as E_A, and thus $p = l_A^{e_A} l_B^{e_B} d \pm 1 \approx l^{2e}$ (remember that $l_A^{e_A} \approx l_B^{e_B}$). Let us assume that e is even for simplicity. The *basic meet-in-the-middle* attack proceeds as follows:

1. Compute all subgroups G of $E[l]$ of order $l^{e/2}$, and their corresponding isogenies and images $\phi_G : E \to E_G$. Store each pair $(G, j(E_G))$.
2. For each subgroup H of $E'[l]$ of order $l^{e/2}$, calculate its corresponding $\psi_H : E \to E_H$ and $j(E_H)$. Then, search in the stored values from Step 1 for a $G \subset E[l]$ with $j(E_G) = j(E_H)$. If one such G is found, one can find an isomorphism $\alpha : E_G \to E_H$ and yield

$$E \xrightarrow{\phi_G} E_G \xrightarrow{\alpha} E_H \xrightarrow{\widehat{\psi_H}} E'$$

where $\widehat{\psi_H}$ denotes the dual isogeny of ψ_H. Clearly we have the desired degree

$$deg(\widehat{\psi_H} \circ \alpha \circ \phi_G) = l^{e/2} \cdot 1 \cdot l^{e/2} = l^e.$$

Observation 10. In the algorithm above we computed a dual isogeny, and a question arises: can isogeny duals for SIDH-like isogenies be computed in polynomial time? The answer is yes. One can find how this can be done in [NR19], together with an optimization of the SIKE protocol using dual isogenies.

However, as we said before and is clear from the definition, the MITM does not run in polynomial time. There are $(l+1)l^{e/2-1}$ subgroups of order $l^{e/2}$ in $E[l^e]$. It follows that the algorithm needs to do, at least, $(l+1)l^{e/2-1}$ Vélu computations to get isogenies of degree $l^{e/2}$. In fact, assuming that the paths in the graph are random, the average case needs $1.5(l+1)l^{e/2-1}$ such computations, and the worst case $2(l+1)l^{e/2-1}$. Since $l^e \approx \sqrt{p}$, the quantity $(l+1)l^{e/2-1}$ is $O(p^{1/4})$. The space complexity of the attack is also huge, because we have to store the pairs $(G, j(E_G))$ for all subgroups $G \subset E[l]$. Again, since there are $(l+1)l^{e/2-1}$ such groups, the space required is $O(p^{1/4})$.

Many optimizations for this strategy have been found, for example using a depth-first-search (DFS) strategy. In [ACVCD+18] they define this optimization and show the following results.

			MITM-basic				MITM-DFS
			expected		measured	clock	clock
e_A	e_B	d	time	space	time	cycles	cycles
32	20	23	$2^{17.17}$	$2^{20.72}$	$2^{17.26}$	$2^{34.50}$	$2^{31.73}$
34	21	109	$2^{18.17}$	$2^{21.83}$	$2^{18.24}$	$2^{35.49}$	$2^{32.71}$
36	22	31	$2^{19.17}$	$2^{22.87}$	$2^{19.14}$	$2^{36.43}$	$2^{33.67}$
38	23	271	$2^{20.17}$	$2^{23.99}$	$2^{20.20}$	$2^{37.59}$	$2^{34.60}$
40	25	71	$2^{21.17}$	$2^{25.04}$	$2^{21.15}$	$2^{38.63}$	$2^{35.71}$
42	26	37	$2^{22.17}$	$2^{26.09}$	$2^{22.11}$	$2^{39.83}$	$2^{36.78}$
44	27	37	$2^{23.17}$	$2^{27.14}$	$2^{23.25}$	$2^{41.07}$	$2^{37.87}$

Figure 9: ([ACVCD+18], TABLE 1) Meet-in-the-middle attacks for finding a 2^{e_A}-isogeny between two supersingular elliptic curves over $\mathbb{F}_{p^2}$ with $p = 2^{e_A} \cdot 3^{e_B} \cdot d - 1$. For each p, 25 randomly generated CSSI instances were solved and the average of the results are reported. The "expected time" and "measured time" columns give the expected number and the actual number of degree-$2^{e_A/2}$ isogeny computations for MITM-basic. The space is measured in bytes.

In [ACVCD+18] they mention that, although the MITM strategy seems to be optimal for the CSSI problem, the space required is completely infeasible for cryptographic applications. They also

give one of the best optimizations, which uses Oorschot-Wiener's parallel collision search [VOW99], which can be found implemented with some improvements in [CLN+19].

3.4 Implementations and performance

In this section we analyze why the SIDH can be done efficiently, how efficient it is and what can be done to optimize the operations required in the protocol, such as large-degree isogeny computations and arithmetic in elliptic curves. Then, we talk about the experimental results on the performance of SIKE, the key-encapsulation SIDH-based protocol competing to become a standard by NIST and we compare it to other post-quantum protocols.

3.4.1 Parameters generation

When we were defining the SIDH protocol we made some implicit assumptions such as the fact that we were able to efficiently choose primes of a given form or random supersingular elliptic curves in a certain field. Let us briefly discuss why this is true, which can also be found in the original papers of SIDH ([DFJP14], [DFJP14]).

First, the small primes l_A, l_B are usually taken as $2, 3$, respectively. However, in some cases it can be useful to use 4 instead of 2 for computational reasons. Although 4 is not a prime, the protocol works properly because $gcd(3, 4) = 1$.

Now we need to choose e_A, e_B, d so that $l_A^{e_A} \approx l_B^{e_B}$ and $p = l_A^{e_A} l_B^{e_B} d \pm 1$ is a prime number. To guarantee that choosing random e_A, e_B, d (such that $l_A^{e_A} \approx l_B^{e_B}$, which is not hard to check) will often yield a prime p, we use the following theorem.

Theorem 7. *([Eve18], Theorem 6.4) Prime number theorem for arithmetic progressions. Let $\pi(x, n, a)$ denote the number of primes $p \leq x$ such that $p \equiv a \ mod(n)$. Then,*

$$\pi(x, n, a) \sim \frac{1}{\varphi(n)} \frac{x}{log(x)} \ as \ x \to \infty$$

where $\varphi(n)$ denotes Euler's totient function, the number of positive integers $1 \leq k \leq n$ such that $gcd(n, k) = 1$.

With this, taking $a = \pm 1$ and $n = 2^{e_A} 3^{e_B}$ we obtain that, when we look at very large numbers, there is a reasonable quantity of primes of the form $p = 2^{e_A} 3^{e_B} d \pm 1$.

Choosing a supersingular elliptic curve E over $\mathbb{F}_{p^2}$ with $(p \mp 1)^2 = (2^{e_A} 3^{e_B} d)^2$ points can be done in $O(log(q)^3)$ as shown in [Brö09], as long as such curve exists. To justify its existence, recall

from 2.2 that $|E(\mathbb{F}_{p^2})| = p^2 + 1 - t$ where t is the trace of Frobenius and also recall Proposition 4, which determined for which values of t there exist supersingular curves. Since $p = 2^{e_A}3^{e_B}d \pm 1$, choosing $t = \pm 2p$ (which Proposition 4 states as possible values for t) we obtain a curve with $|E(\mathbb{F}_{p^2})| = p^2 + 1 \mp 2p = (p \mp 1)^2 = (2^{e_A}3^{e_B}d)^2$. Then we can use Bröker's algorithm to compute the curve efficiently.

From the curve given by Bröker's algrithm, one can find a "random" curve by randomly walking in supersingular isogeny graphs. This random walks should be properly defined in order to formally justify the efficiency of parameter generation, we just give the idea and refer to [DFJP14] for further details.

It only remains to see how one can efficiently choose points P, Q that generate $E[l^e]$. Not only P, Q need to have maximal order l^e, but we also need them to be independent. However, the latter can be easily verified and will hardly ever happen, so the main difficulty is fining points of maximal order. Recall that $E[l^e] \cong \mathbb{Z}/l^e\mathbb{Z} \times \mathbb{Z}/l^e\mathbb{Z}$ and note that for a point $(a, b) \in \mathbb{Z}/l^e\mathbb{Z} \times \mathbb{Z}/l^e\mathbb{Z}$ to have maximal order l^e it suffices that one of its two components has order l^e. Also, an element $a \in \mathbb{Z}/l^e\mathbb{Z}$ has order l^e if and only if $gcd(a, l^e) = 1$. But $gcd(a, l^e) = 1$ if and only if $gcd(a, l) = 1$, so a has order l^e in $\mathbb{Z}/l^e\mathbb{Z}$ if and only if a modulo l has order l in $\mathbb{Z}/l\mathbb{Z}$. This means that the elements of maximal order in $\mathbb{Z}/l^e\mathbb{Z}$ are numbers of the form $n + kl$ with $gcd(n, l) = 1$ and $0 \leq k < l^{e-1}$. With $\varphi(l)$ choices for n and l^{e-1} choices for k, we obtain that there are $l^{e-1}\varphi(l)$ such numbers.

Finally with the primes chosen as 2 and 3 we can state that

- In $\mathbb{Z}/2^{e_A}\mathbb{Z}$ there are 2^{e_A-1} elements of maximal order (which is a half of all the elements). Then, a random point in $\mathbb{Z}/2^{e_A}\mathbb{Z} \times \mathbb{Z}/2^{e_A}\mathbb{Z}$ has maximal order with probability at least $\frac{3}{4}$.
- In $\mathbb{Z}/3^{e_B}\mathbb{Z}$ there are $2 \cdot 3^{e_B-1}$ elements of maximal order (which is $\frac{2}{3}$ of all the elements). Then, a random point in $\mathbb{Z}/3^{e_B}\mathbb{Z} \times \mathbb{Z}/3^{e_B}\mathbb{Z}$ has maximal order with probability at least $\frac{8}{9}$.

This justifies taking random $l_A^{e_A}$-torsion points to obtain maximal order points in $E[l_A^{e_A}]$. As a way to generate these random $l_A^{e_A}$-torsion points, in [JDF11] they suggest choosing random points in the curve and multiplying by $(l_B^{e_B}d)^2$. The same reasoning holds for $l_B^{e_B}$.

3.4.2 Multiplication map computation

Recall the multiplication-by-m map $[m]P = P + \overset{m}{\cdots} + P$, and let us take a look at how this operation should be done in an elliptic curve.

The naive approach is to compute the literal sum $P + \overset{m}{\cdots} + P$. However, as we have seen, the addition in elliptic curve requires an important amount of time (compared to the operations of other

groups used in cryptography, such as $\mathbb{Z}/p\mathbb{Z}$). This first approach requires $m-1$ sums.

Let us show a technique called *double-and-add*, which is commonly used to compute this type of operations. Assume we want to compute $[b]P$ and let $b = b_0 + 2b_1 + 2^2b_2 + \cdots + 2^nb_n$ be the binary decomposition of b. Then, $[b]P = [b_0]P + [2][b_1]P + \left[2^2\right][b_2]P + \cdots + [2^n][b_n]P$ and each $[2^s]$ is, in fact, $[2]^s = [2] \circ \cdots \circ [2]$. Now we use this decomposition of $[b]P$ to compute it term by term.

The idea is to iterate from 0 to n computing the corresponding term. Of course, we only do computations if $b_i = 1$, and in this case the $\left[2^i\right]P$ is computed as $[2]\left(\left[2^{i-1}\right]P\right)$ because we have stored $\left[2^{i-1}\right]P$ from the previous computation.

With this strategy we do $log_2(b)$ doubling operations and, at most, the same number of point adding operations. The worst case of this strategy then needs $2log_2(b)$ point adding operations, which is much better than the naive strategy. However, as mentioned in [JDF11], the double-and-add strategy is vulnerable to a side-channel attack known as Simple Power Analysis (SPA). Assuming an attacker has access to the power consumption of the machine computing $[b]P$ with this algorithm, it is not hard to determine the binary representation of b, distinguishing the case when the machine does not compute operations (when $b_i = 0$) and the case when it does (when $b = 1$), and thus consumes more power. This can be solved by adding noise to the signal or performing unnecessary or dummy operations.

In the SIDH case we want to compute $nP + mQ$ for $P, Q \in E[l^e]$ and $n, m \in \mathbb{Z}/l^e\mathbb{Z}$, which looks like two multiplication maps and a sum. However, since either n or m are invertible modulo l^e (we assume it is n), it suffices to compute $P + n^{-1}mQ$ (or $m^{-1}nP + Q$, in the other case), because they generate the same subgroup of $E[l^e]$. In ([JDF11], §4.2.1 Algorithm 1) they propose an algorithm for such purpose, which they claim to be much better than the other general approaches. This same algorithm is also used in the other two implementations we cite in this section [CLN16], [AFJ].

3.4.3 Large degree isogenies computation

As we mentioned right after the protocol specification, the isogenies used in SIDH are too large to use Vélu formulas. Let us explain how these isogenies can be computed efficiently. From Theorem 4 we know that l^e-isogenies are compositions of e isogenies of degree l. Let us show how to construct these l-isogenies in the case of SIDH. Assume we start in E_0 and we want to compute an isogeny $\phi : E_0 \to E$ with kernel $\langle R \rangle = \langle nP + mQ \rangle$, where $\langle P, Q \rangle = E_0[l^e]$ and $(n, m) \not\equiv (0, 0)\ mod(l)$. As we saw in the proof of Theorem 6, the condition on n, m implies that R has maximal order l^e.

Let $R_1 = [l^{e-1}]R \in E_0$. Since $ord(R) = l^e$, the order of R_1 is l, and hence we can efficiently

use Vélu to find $\phi_1 : E_0 \to E_1 \cong E_0/[l^{e-1}]R$ with $ker(\phi_1) = \langle R_1 \rangle$.

Let $R_2 = [l^{e-2}]\phi_1(R) = \phi_1([l^{e-2}]R) \in E_1$. Note that R_2 has order l, because $[l]R_2 = \phi_1([l^{e-1}]R) = \mathcal{O}$ by construction. Since $ord([l^{e-2}]R) = l^2$ and $ord(R_2) = l$ we can compute Vélu again to find $\phi_2 : E_1 \to E_2$ with $ker(\phi_2) = R_2$. Now note that we have $E_2 \cong E_0/[l^{e-2}]R$. This is easy to see using the same argument as in the proof of Theorem 6. We have that

- The composition $\phi_2 \circ \phi_1$ has degree l^2 because both isogenies have degree l.
- The point $[l^{e-2}]R$ of order l^2 is in the kernel of $\phi_2 \circ \phi_1$ by construction.

and that means that $ker(\phi_2 \circ \phi_1) = \langle [l^{e-2}]R] \rangle$ and hence $Im(\phi_2 \circ \phi_1) = E_2 \cong E/[l^{e-2}]R$

Similarly, any $1 \leq i \leq e$ we have that $R_i = [l^{e-i}](\phi_{i-1} \circ \cdots \circ \phi_1)R \in E_i$ has order l and using Vélu we obtain an isogeny $\phi_i : E_{i-1} \to E_i$ with $E_i \cong E_0/[l^{e-i}]R$. This ends with an isogeny chain $\phi_e \circ ... \phi_1 : E_0 \to E$ of degree l^e and $E \cong E_0/R$.

This way, instead of computing the isogeny of degree l^e using Vélu, which we know that requires $O(l^e)$ operations, we compute e isogenies of degree l, which can be done efficiently for small values of l like $2, 3, 4$. This strategy is enough for us to understand how this computation can be done efficiently, but much better strategies can be found in the literature for the computation of such isogenies. In [JDF11] they discuss deeply this aspect.

3.4.4 Performance

In [KRSS19] they test most of the Round 2 submissions of the NIST post-quantum standardization project (see 3.2.5) on a ARM Cortex-m4 processor. Remember that SIKE is a key encapsulation scheme, therefore the time is measured for key encapsulation and key decapsulation instead of encryption and decryption algorithms. The running times of the SIKE protocol are studied in [SJA19] and in [JAK+19], with the performance being analyzed from different possible processors. They also discuss the feasibility of SIKE working in mobile devices.

We will show some results of the tests on the performance of SIKE, focusing on "SIKEp503" and "SIKEp751", where the "pXXX" refers to the size of the prime p used. In the former it stands for $p = 2^{250}3^{159} - 1$ with $log_2(p) \approx 503$ and the latter for $p = 2^{372}3^{239} - 1$ with $log_2(p) \approx 751$. This sizes are chosen in order to achieve certain security levels defined by NIST. SIKEp503 claims to have security level II, which means that breaking it is as hard as breaking SHA256 (collision search) and SIKEp751 claims to have security level V, which means that breaking it is as hard as breaking AES256 (exhaustive key search).

[14] A clock cycle is a very small amount of time, corresponding to the latency with which processors compute

In the SIKE 2nd round specification [CCH+19] their results show that a full SIKEp503 protocol (with key generation, key encapsulation and decapsulation) runs in about 91.764.000 clock cycles[14] on an ARMv8 assembly, while the SIKEp751 needs 307.105.000 clock cycles. They also show a compressed version of SIKE protocols, which achieves $\approx 40\%$ compression of key sizes in exchange for a higher running time. The following table summarizes the memory needed in these protocols.

Scheme	secret key sk	public key pk	ciphertext ct	shared secret ss
SIKEp434	(44+330) 374	330	346	16
SIKEp503	(56+378) 434	378	402	24
SIKEp610	(62+462) 524	462	486	24
SIKEp751	(80+564) 644	564	596	32
SIKEp434_compressed	(43+196) 239	196	209	16
SIKEp503_compressed	(56+224) 280	224	248	24
SIKEp610_compressed	(62+273) 336	273	297	24
SIKEp751_compressed	(79+334) 413	331	363	32

Figure 10: ([CCH+19], Table 2.2) Memory of keys and texts in SIKE protocol, measured in bytes.

However, a compressed SIKE503 full protocol runs in 151.262.000 clock cycles in their tests and in the case of compressed SIKEp751 they get 524.448.000 clock cycles, which is about 40% slower than the non-compressed versions.

The numbers results obtained in [KRSS19] can not be compared to the ones we have shown, because they do not use the same optimizations and processors, but they do show some notable things. First, in their results, many versions of SIKE have the highest running times among their respective competitors in the same NIST security level. However, recent studies ([KAEK+], [SJA19], [JAK+19]) of optimization in the performance of SIKE already show the feasibility of the protocol in different devices. They also see how SIKE requires significantly less space than the average of candidates. For example, their executions of SIKEp751 are about 2500 times slower (in terms of clock cycles) than NewHope1024, a LWE-based protocol with the same NIST security level, but NewHope1024 needs double the space than its isogeny-based competitor. This may not look like a fair trade, but note that the speed can always be improved with faster processors, which execute more clock cycles per second, while the space required will always be the same. Another good example of this trade is the Classic McEllice (which we talked about in 1.6.3). While the protocol runs in relatively fast constant time (see [BCL+17]), the public key sizes range from 255 KiB to 1326 KiB, where 1 KiB = 1024 B. This is much higher than SIKE's public key sizes, that range from 330 B to 564 B in the non-compressed versions.

Although the precise numbers may differ between different comparison articles between post-

operations and hence devices change their internal state.

quantum protocols, the conclusion to which all the tests agree is that isogeny-based have the shortest key sizes and the largest running time. This is because the keys are usually few points in an elliptic curve, which can be stored in relatively short space, but arithmetic operations inside elliptic curves require more time than in many other structures. Apart from SIKE, 16 other key encapsulation methods have passed to the second round of NIST standardization project. From these, 9 are lattice-based and 7 are code-based, and the former seem to require less space both in key sizes and in ciphertexts than the latter. However, we will not analyze the performance differences between code and lattice based protocols because it is beyond the scope of this work.

3.5 Supersingular versus ordinary elliptic curves

The SIDH protocol is not the first one that uses the hard problem of finding isogenies between elliptic curves. Stolbunov [Sto10] presented a key agreement scheme (together with a public encryption scheme) that was based in the group action of isogenies on elliptic curves. However, his protocol used ordinary elliptic curves. This is not because he did not think of using supersingular elliptic curves, which he did (see [Sto10], p.221), but because the endomorphism ring of an ordinary elliptic curve is commutative, and that leads to the idea of group action. The scheme uses the commutativity of the endomorphism ring to define a group action, that implicitly defines a key agreement scheme as we saw in 1.5.3.

Stulbonov's protocol was found vulnerable to a quantum attack that recovered the secret key in subexponential time [CJS14]. Using precisely the fact that the endomorphism ring is commutative, they made a reduction to a particular hidden shift problem[15] (HSP), where the blackbox functions required action evaluation. They showed that this evaluation could be done in subexponential time and, with that, they also proved that the HSP could be solved in quantum-subexponential time. That is what motivated researchers to try to create cryptographic schemes based on the hardness of finding isogenies between supersingular elliptic curves[16].

[15] Let A be a finite group and S a finite set. Let $f, g : A \to S$ two black-box functions such that there is a shift $s \in A$ with $f(s) = g(s + x) \forall x \in A$. The HSP is to determine s with this information. In [CJS14], the group A is abelian and they also use the fact that f is injective.

[16] In fact they mention the possibility of using supersingular elliptic curves in cryptography and understanding the hardness of finding isogenies between them as an "interesting open problem" ([CJS14], p.3). In fact, one of the authors of the attack (D. Jao) is also part of the team that presented the SIDH protocol.

4. Conclusion

Assuming barely no knowledge on cryptography, our main goal was to understand the supersingular isogeny Diffie Hellman key exchange from its most mathematical and theoretical point of view. We have seen the basics of classical cryptography, empathizing on the Diffie Hellman key agreement, and motivated the rising of post-quantum protocols as a need in the sights of a computation model that will soon become a real threat to most of the classical cryptographic schemes.

We have briefly studied the basics of elliptic curves and its arithmetic, focusing on its rich algebraic structure, and from there isogenies appeared as a natural need for their structure-preserving property. Given that isogeny-finding is believed to be a hard problem for cryptography, we then focused on our main object of study, the SIDH, merging the Diffie Hellman structure we had seen together with this relatively new hard problem of finding isogenies. We then analyzed this problem from its mathematical perspective trying to understand the real hardness of it, reviewing some of the known attempts to solve it. Finally, we connected this theoretical idea of a key exchange protocol with the implementation of it that is competing to become a standard in the post-quantum cryptographic world and seen its performance in comparison with the other competitors.

In our way to understand SIDH we have proven some important mathematical results (such as isogeny decomposition or SIDH correctness) which are not usually proven in cryptographic articles and we believe this can help cryptographers who try to understand the mathematical fundamentals behind the schemes. We also think that this work can also be useful for mathematicians who want to introduce to the field of isogeny-based cryptography as we avoided any assumption of cryptographic knowledge and tried to follow a logical mathematical order and style.

As future work, we believe that an interesting path would be to study the structure of the endomorphism ring of elliptic curves (ordinary and supersingular) and how it affects isogenies. In particular, the relation between isogeny-finding problem and the problem of computing the endomorphism ring of a given curve. Another interesting research field in the direction of this work is the possible generalization of the notions we analyzed to hyperelliptic curves of genus 2 or higher.

References

[ACVCD+18] Gora Adj, Daniel Cervantes-Vázquez, Jesús-Javier Chi-Domínguez, Alfred Menezes, and Francisco Rodríguez-Henríquez. On the cost of computing isogenies between supersingular elliptic curves. In *International Conference on Selected Areas in Cryptography*, pages 322–343. Springer, 2018.

[AFJ] Reza Azarderakhsh, Dieter Fishbein, and David Jao. Efficient implementations of a quantum-resistant key-exchange protocol on embedded systems. Technical report, 2014 http://cacr.uwaterloo.ca/techreports/2014/cacr2014-20.pdf.

[BCL+17] Daniel J Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, et al. Classic mceliece: conservative code-based cryptography. *NIST PQC Competition Submissions*, 2017.

[Brö09] Reinier Bröker. Constructing supersingular elliptic curves. *J. Comb. Number Theory*, 1(3):269–273, 2009.

[CCH+19] Matthew Campagna, Craig Costello, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, David Urbanik, et al. Supersingular isogeny key encapsulation. 2019. https://sike.org/.

[CCSKK15] Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim, and Taewan Kim. Lattice based cryptography for beginners. *IACR Cryptology ePrint Archive*, 2015:938, 2015.

[CJS14] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.

[CLN16] Craig Costello, Patrick Longa, and Michael Naehrig. Efficient algorithms for supersingular isogeny diffie-hellman. In *Annual International Cryptology Conference. CRYPTO 2016, LNCS 9814,*, pages 572–601. Springer, 2016.

[CLN+19] Craig Costello, Patrick Longa, Michael Naehrig, Joost Renes, and Fernando Virdia. Improved classical cryptanalysis of the computational supersingular isogeny problem. *IACR Cryptology ePrint Archive*, 2019:298, 2019.

[Con96] Ian Connell. Elliptic curve handbook. *Preprint*, 1996.

[DF17] Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.

[DFJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

[Eve18] Jan-Hendrik Evertse. Analytic number theory lecture notes. *Liden University*, 2018. `http://pub.math.leidenuniv.nl/~vertsejh/ant-mastermath.html#coursenotes`.

[Gal12] Steven D Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[Gal18] Steven D Galbraith. Authenticated key exchange for sidh. *IACR Cryptology ePrint Archive*, 2018:266, 2018.

[GPST16] Steven D Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2016, LNCS 10031,*, pages 63–91. Springer, 2016.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[JAK+19] Amir Jalali, Reza Azarderakhsh, Mehran Mozaffari Kermani, Matthew Campagna, and David Jao. Optimized supersingular isogeny key encapsulation on armv8 processors. *IACR Cryptology ePrint Archive*, 2019:331, 2019.

[JDF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography, PQCrypto 2011, LNCS 7071*, pages 19–34. Springer, 2011.

[KAEK+] Brian Koziel12, A-Bon Ackie, Rami El Khatib, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Sike'd up: Fast and secure hardware architectures for supersingular isogeny key encapsulation. IACR Cryptology ePrint Archive, 2019:711, 2019.

[KD13] Cameron F Kerry and Charles Romine Director. Fips pub 186-4 federal information processing standards publication digital signature standard (dss). 2013.

[KRSS19] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. 2019. IACR Cryptology ePrint Archive, 2019:844, 2019.

[KZ98] Masanobu Kaneko and Don Zagier. Supersingular j-invariants, hypergeometric series, and atkin's orthogonal polynomials. *AMS IP Studies in Advanced Mathematics*, 7:97–126, 1998.

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[NR19] Michael Naehrig and Joost Renes. Dual isogenies and their application to public-key compression for isogeny-based cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security,ASIACRYPT 2019, LNCS 11922,*, pages 243–272. Springer, 2019.

[P+16] Chris Peikert et al. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.

[Pet17] Christophe Petit. Faster algorithms for isogeny problems using torsion point images. In *International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2017, LNCS 10625,*, pages 330–353. Springer, 2017.

[PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over gf (p) and its cryptographic significance (corresp.). *IEEE Transactions on information Theory*, 24(1):106–110, 1978.

[Sch87] René Schoof. Nonsingular plane cubic curves over finite fields. *Journal of combinatorial theory, Series A*, 46(2):183–211, 1987.

[Sho94] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[Sil09] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009.

[SJA19] Hwajeong Seo, Amir Jalali, and Reza Azarderakhsh. Sike round 2 speed record on arm cortex-m4. In *International Conference on Cryptology and Network Security, ANS 2019, LNCS 11829,*, pages 39–60. Springer, 2019.

[Sto10] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. in Math. of Comm.*, 4(2):215–235, 2010.

[Sut17] Andrew Sutherland. 18.783 elliptic curves lecture. *MIT*, 2017.

[Tan07] Seiichiro Tani. Claw finding algorithms using quantum walk. *arXiv preprint arXiv:0708.2584*, 2007.

[Tes99] Edlyn Teske. The pohlig–hellman method generalized for group structure computation. *Journal of Symbolic Computation*, 27(6):521–534, 1999.

[VOW99] Paul C Van Oorschot and Michael J Wiener. Parallel collision search with cryptanalytic applications. *Journal of cryptology*, 12(1):1–28, 1999.

[Wan10] Frédéric Wang. The hidden subgroup problem. *arXiv preprint arXiv:1008.0010*, 2010.

[Wat69] William C Waterhouse. Abelian varieties over finite fields. In *Annales scientifiques de l'École Normale Supérieure*, volume 2, pages 521–560, 1969.