Isogeny-Based Cryptography Tutorial

ISARA Corporation

August 1, 2019

www.isara.com

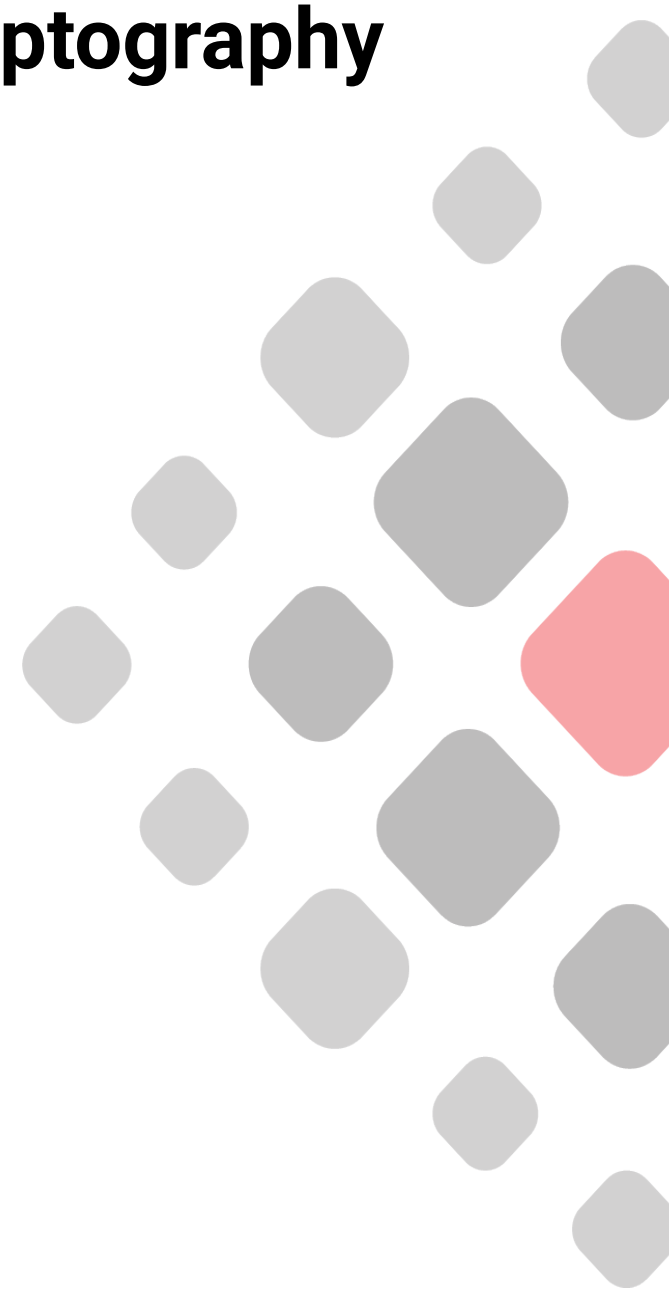

Table of Contents

1 Introduction 2
2 Elliptic Curves 2
3 Elliptic Curve Diffie-Hellman (ECDH) 6
4 Isogenies 8
5 Explicit Constructions 13
6 Supersingular Isogeny Diffie–Hellman (SIDH) 16
7 About ISARA Corporation 18

1 Introduction

In this tutorial, we provide a gentle introduction to supersingular isogeny-based cryptography. By the end of this document, the reader will have obtained a high-level understanding of the math behind this promising area of cryptography and will have surveyed the most important key establishment primitive based on isogeny-based cryptography.

More specifically, we organized this tutorial as follows. We will start with a short description of elliptic curves and their scalar multiplication, which forms the foundation of Elliptic Curve Diffie-Hellman (ECDH). Those comfortable with the basics of ECDH should feel free to skip to Section 4, which covers the definition of an isogeny and how an isogeny can be determined via its kernel. We will then discuss how to randomly choose kernels, and how to use the resulting isogenies to replace the scalar multiplication in ECDH to get a new quantum-safe key establishment algorithm called Supersingular Isogeny-Based Diffie-Hellman (SIDH).

2 Elliptic Curves

Historically, one of the most important problems in mathematics was to describe solutions to polynomial equations. One standard problem is to find points on a curve in a plane described by a polynomial $f(x, y)$.

Definition 2.1. By *curve* we will always mean the solutions of a polynomial equation in two variables. A curve of *degree* d will mean that the highest degree of any term in the polynomial has degree d.

If the degree of this polynomial is one, then the curve is a line and finding the points on a line is a trivial problem. Solving the degree two case involves investigating conic sections, and this problem has a well-known solution.

The degree three case, however, results in some remarkably complex and interesting mathematics. This mathematics gives rise to hard problems that are at the heart of well-known classical and quantum-safe cryptographic algorithms. It is these degree three polynomials that we will be discussing in this document.

Definition 2.2. An *elliptic curve* is a curve defined by an equation of the form $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$.

The condition $4a^3+27b^2 \neq 0$ guarantees that there are no troublesome (singular) points like those given in Figure 1.

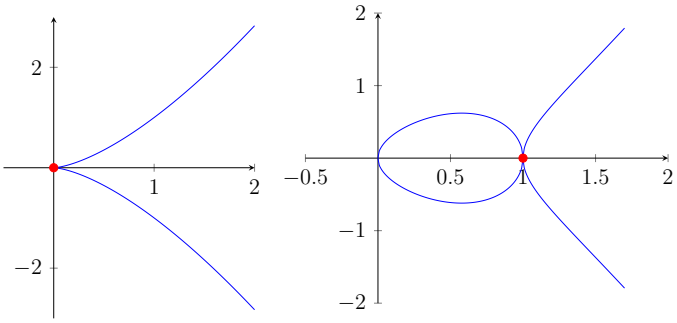

Figure 1: A cusp and a point of self-intersection

Lemma 2.3. *Given an elliptic curve $y^2 = x^3 + ax + b$, where a and b are real numbers, there are either one or three real roots of $x^3 + ax + b = 0$.*

The two cases covered in Lemma 2.3 are illustrated in Figure 2.

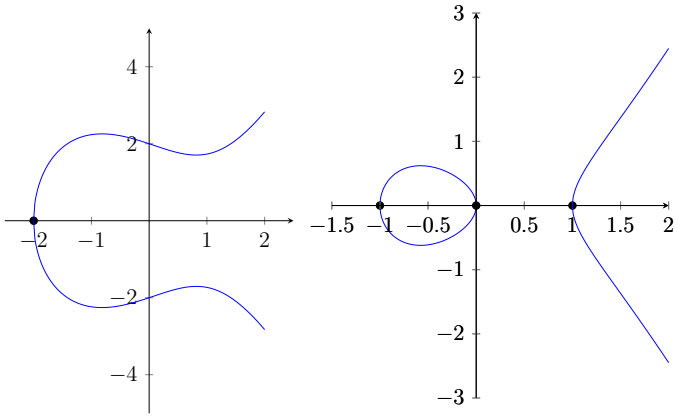

Figure 2: Elliptic curves with one and three real roots

Given two points on an elliptic curve, we can get a third point on the curve in a natural way, which we will call the sum of these two points. To understand this "addition" of points it is important to understand the way lines and elliptic curves intersect. Let us begin with the most common case.

Case 1: Suppose we are given two different points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on an elliptic curve E, where the line between them is non-vertical and intersects E at a distinct third point. Let $R = (x_3, y_3)$ denote the third intersection point between the line and E. We define the addition $P + Q$ to be the reflection of R with respect to the x-axis, namely, $(x_3, -y_3)$—see Figure 3.

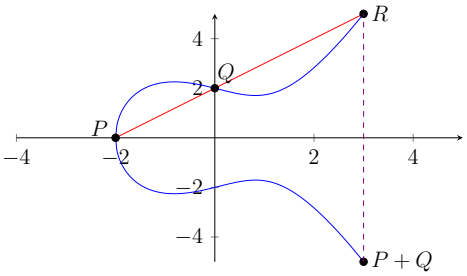

Figure 3: Case 1

The next case covers the situation where we want to double a point, or in other words define $P + P$.

Case 2: Choose $P = (x_1, y_1)$ on an elliptic curve E. The *tangent line* of E at P is the line that passes through P in a way where the line has the same slope as the curve E at the point P—see Figure 4. It is well-known that if this tangent line is non-vertical, then it will intersect E at one other point $R = (x_3, y_3)$. We define $P + P = (x_3, -y_3)$—see Figure 4.

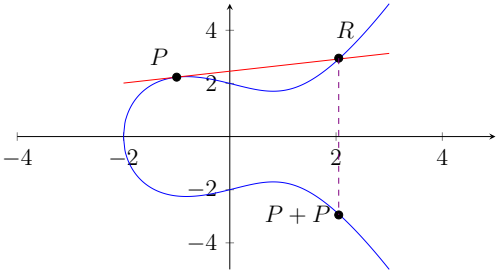

Figure 4: Case 2

If the line between two different points does not intersect at a third point, then it can be shown that the line is a tangent line to one of these two points, as is shown in Figure 5. This happens in the final case involving a non-vertical line.

Case 3: Suppose we are given two different points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on an elliptic curve E, where the line between them is non-vertical and intersects E at the point P. This happens when the line is tangent to E at P. For mathematical reasons, we think of the line being a tangent line of E at P as implying that the line and E have a double intersection at P, much in the same way that as we think of $y = x^2$ as having a double root at $(0, 0)$. Let $R = P$, (the “third" point of intersection). We define $P + Q = (x_1, -y_1)$—see Figure 5.

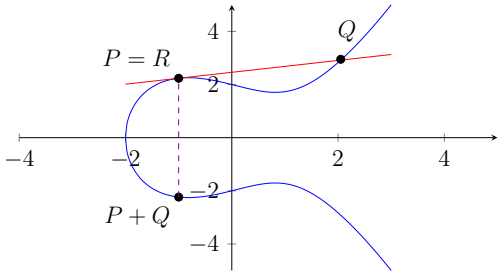

Figure 5: Case 3

The above cases cover all the situations where the line between two points (or the tangent line at a point) is non-vertical. We will briefly discuss the situation where the line is vertical.

Case 4: The case where the line is vertical is taken care of by introducing a special point $\mathcal{O}$ to the points on the elliptic curve, which is called the *identity of* E (or the *point at infinity* of E) and satisfies the property $P + \mathcal{O} = P$ for any point $P \in E$. This extra point does not lie in the plane. We can think of this special point as being like 0, in the sense that $x + 0 = x$, for any real number x. The only time when the line between two points is vertical is if the points have the same x-coordinate. Thus, we define $(x_1, y_1) + (x_1, -y_1) = \mathcal{O}$. For this reason, if $P = (x_1, y_1)$, then $-P = (x_1, -y_1)$. This is depicted in Figure 6.

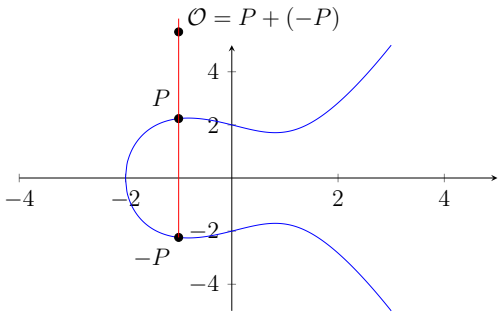

Figure 6: Case 4

This addition operation is sometimes referred to as being defined by the “chord and tangent rule." The main take-away is that there is a simple method to add points on an elliptic curve that covers all the possible cases. Note that we can easily find explicit formulas to compute addition of points in Cases 1 to 4. For more information, see [6, Section 1.4] or [5, Section 3.2].

Definition 2.4. Given a point P on an elliptic curve and a natural number n, we have a notion of (*scalar*) *multiplication by n*, which is simply summing P a total of n times and is denoted by $[n] \cdot P$. In other words

$$[n] \cdot P = \underbrace{P + \cdots + P}_{n}.$$

Additionally, we let $[0] \cdot P = \mathcal{O}$ and $[-n] \cdot P = [n] \cdot (-P)$. For a fixed natural number n and elliptic curve E, we call the map $[n] \cdot E \to E$ that maps a point P to the point $[n] \cdot P$ the *scalar multiplication by n map*.

Instead of working with elliptic curves whose points have real or rational x and y values, classical elliptic curve cryptography assumes the points on the elliptic curve have x and y values that are in a finite space $\mathbb{F}_p$ for some prime p. (The coordinates a and b that define the elliptic curve $y^2 = x^3 + ax + b$ are also in $\mathbb{F}_p$.) We can think of $\mathbb{F}_p$ as simply being the numbers $0, 1, \ldots, p-1$ where addition and multiplication is done modulo p. Recall, two numbers are the same modulo p if their difference is divisible by p.

Example 2.5. If we are working in $\mathbb{F}_7$, then

$$3 + 2 \cdot 5 = 3 + 10 = 13 = 6.$$

The last step works because 13 is equal to 6 modulo 7.

The formulas that describe how to add points on the elliptic curves when the points are real numbers still hold when we are working with arithmetic modulo a prime p. Using this we can now explain Elliptic Curve Diffie-Hellman (ECDH), a prevalent classical key establishment algorithm.

3 Elliptic Curve Diffie-Hellman (ECDH)

First presented in 1976, the Diffie-Hellman key exchange is one of the earliest practical examples of public key cryptography and allows two parties to securely establish a shared secret over an insecure channel. Initially proposed in terms of natural numbers and exponents, we now present its more modern elliptic curve variant which relies upon the scalar multiplication by n map.

Using the scalar multiplication by n map—where the points have x and y coordinates in $\mathbb{F}_p$—we will now describe the ECDH algorithm.

Public Parameters: An elliptic curve E defined over $\mathbb{F}_p$ for some prime p, and a point P on the elliptic curve (with some nice properties).

Key Generation: Alice begins by choosing a random number n_A, and calculates $[n_A] \cdot P$. Her private/public key pair is then $(n_A, [n_A] \cdot P)$. Similarly, Bob begins by choosing a random number n_B, and calculates $[n_B] \cdot P$. His private/public key pair is $(n_B, [n_B] \cdot P)$. Alice and Bob exchange public keys $[n_A] \cdot P$ and $[n_B] \cdot P$.

Shared Secret: Both parties can now calculate a shared secret, namely $[n_A n_B] \cdot P$. Using the point $[n_B] \cdot P$ that she receives from Bob, Alice computes $[n_A] \cdot ([n_B] \cdot P)$.

Similarly, using the point $[n_A] \cdot P$ that he receives from Alice, Bob computes $[n_B] \cdot ([n_A] \cdot P)$. Since the following equation holds:

$$[n_A] \cdot ([n_B] \cdot P) = [n_A n_B] \cdot P = [n_B n_A] \cdot P = [n_B] \cdot ([n_A] \cdot P),$$

we see they have a shared secret.

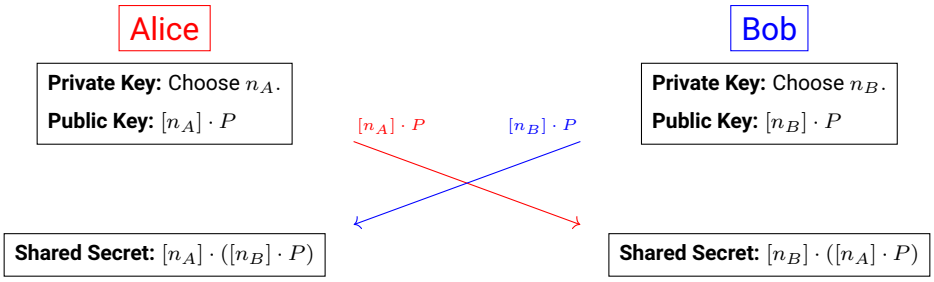

Figure 7: ECDH

The ECDH algorithm is summarized in Figure 7. We can also use a mathematical representation to describe ECDH—see Figure 8 . Alice calculates the arrows in red, and Bob calculates the arrows in blue. In Figure 8, the dotted lines represent the parties exchanging the points $[n_A] \cdot P$ and $[n_B] \cdot P$.

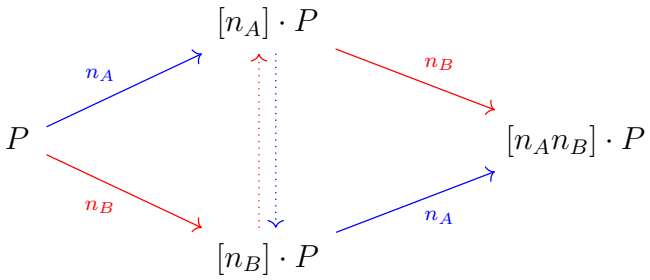

Figure 8: ECDH

The security of ECDH is based on the hardness of the following problem:

Definition 3.1. The *elliptic curve discrete log problem (ECDLP)* is the problem of finding a number n such that $[n] \cdot Q = P$, for given points P and Q on an elliptic curve E over $\mathbb{F}_p$, provided that P is indeed a multiple of Q.

For elliptic curves over $\mathbb{F}_p$ where p and n are large, ECDLP is considered to be an intractably hard problem even with access to today's most powerful supercomputers. Thus, revealing P and Alice's public key $[n_A] \cdot P$, does not reveal Alice's private key n_A. Similarly, revealing P and Bob's public key $[n_B] \cdot P$, does not reveal Bob's private key n_B. This means that given $P, [n_A] \cdot P$, and $[n_B] \cdot P$, it is computationally

infeasible to calculate the shared secret $[n_A n_B] \cdot P$. In this sense, the security of ECDH relies on the hardness of ECDLP.

However, if an adversary had access to a large-scale quantum-computer, then they could run an algorithm invented by the mathematician Peter Shor which could efficiently solve ECDLP [4]. Although this type of large-scale quantum-computer is still years away, due to its reliance on the hardness of ECDLP, ECDH is not a long-term solution for internet security.

For this reason we introduce isogeny-based cryptography. As we will see, isogenies provide a means to create a quantum-safe key establishment algorithm. A *quantum-safe (or post-quantum)* cryptographic algorithm is an algorithm that is believed to still be resistant to an adversary with access to a large-scale quantum computer.

4 Isogenies

The rest of this document will focus on a quantum-safe key establishment algorithm titled Supersingular Isogeny-Based Diffie-Hellman (SIDH). In this section we provide a comparison between the mathematical objects used in ECDH and SIDH. These are summarized in Table 1, and we will explain them as we proceed. In Section 5 we will explain how to explicitly construct these objects. We will describe the full SIDH protocol in Section 6.

	ECDH	SIDH
Quantum-Safe	No	Yes
Parameters	Elliptic curve E over $\mathbb{F}_p$, Point P	(Supersingular) elliptic curve E over $\mathbb{F}_{p^2}$, Points P_A, Q_A, P_B, Q_B
Private Key	Scalar multiplication map $[n_A]$ (or $[n_B]$)	Isogeny ϕ_A (or ϕ_B)
Public Key	Point $[n_A] \cdot P$ (or $[n_B] \cdot P$)	Image elliptic curve $\phi_A(E)$ (and two points) (or $\phi_B(E)$ and two points)
Shared Secret	$[n_A] \cdot [n_B] \cdot P = [n_B] \cdot [n_A] \cdot P$	$j(\psi_A(E_B)) = j(\psi_B(E_A))$ (ψ_A and ψ_B are determined by the private keys)
Hard Problem	Given P and $[n] \cdot P$, find n	Given E and $\phi(E)$ (and two points), find ϕ

Table 1: Comparison between ECDH and SIDH

Private Keys—Replacing Scalar Multiplications by Isogenies

We would like to have a protocol similar to ECDH that is quantum-safe. We can think of scalar multiplication as the function of the base of ECDH—see Figure 8. Unfortunately, a quantum enabled adversary could break cryptosystems based on scalar multiplication (such as ECDH), and so we want to use more general functions to construct a similar-looking algorithm that is also quantum-safe.

We want these more general functions to map elliptic curves to elliptic curves. However, unlike with scalar multiplication, the starting and ending elliptic curves of these more general functions need not be the same. We will focus on the next type of function, because these functions have the property that the image of an elliptic curve is still an elliptic curve.

Definition 4.1. A *rational map* is a function between two curves where the coordinate functions are each defined by a ratio of polynomials. More formally, a rational map has the form

$$\phi(P) = \phi(x, y) = \left(\frac{p_1(x, y)}{q_1(x, y)}, \frac{p_2(x, y)}{q_2(x, y)} \right),$$

for any point $P = (x, y)$ on the (domain) curve, where p_1, p_2, q_1, q_2 are polynomials in two variables.

We also want our functions to preserve addition; in other words, if we add two points before we apply the function, we get the same result as if we apply the function of the two points and then add them (that is, $\phi(P + Q) = \phi(P) + \phi(Q)$).

Definition 4.2. An *isogeny* ϕ between elliptic curves E_0 and E_1 is a rational map $\phi : E_0 \to E_1$ that preserves addition (in technical terms, is a group homomorphism).

From now on, the only functions (or maps) that we will be dealing with will be isogenies. Far from being mysterious, isogenies have been studied extensively. In the following example we will see that we have already been working with them.

Example 4.3. For a fixed natural number n, the scalar multiplication by n map is an isogeny. For example, multiplication by 2 for the elliptic curve $y^2 = x^3 + ax + b$ can be described explicitly as follows:

$$[2] \cdot (x, y) = \left(\tfrac{x^4 - 2ax^2 - 8bx + a^2}{4(x^3 + ax + b)}, \tfrac{(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - a^3 + 8b^2)y}{4(x^3 + ax + b)^2} \right).$$

This description of the scalar multiplication map $[2]$ as a rational map follows from the description given in Cases 1-4 in Section 2, as shown in [5]. Furthermore, the multiplication by 2 map preserves addition, in the sense that

$$[2] \cdot (P + Q) = [2] \cdot P + [2] \cdot Q.$$

Therefore, since it can be defined in terms of rational maps and it preserves addition, the scalar multiplication by 2 map is an isogeny.

As we mentioned earlier, we would like to replace the scalar multiplication maps in ECDH with more general isogenies. In particular, looking at Figure 8, we want to replace the scalar multiplication map $[n_A]$ by isogenies ϕ_A and ψ_A, and we want to

replace the scalar multiplication map $[n_B]$ by isogenies ϕ_B and ψ_B. As we will see later the isogenies ψ_A and ψ_B are equivalent to the isogenies ϕ_A and ϕ_B, respectively, but they start from different curves. This is depicted in Figure 9.

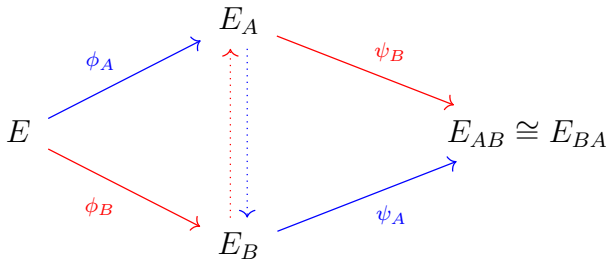

Figure 9: SIDH

Alice's private key will essentially be the isogeny ϕ_A and Bob's private key will be the isogeny ϕ_B. Alice's private key will allow her to calculate ψ_A and Bob's private key will allow him to calculate ψ_B.

Public Keys—Replacing Points by Image Curves

In ECDH, the public keys and shared secret are each a single point that is the image under a scalar multiplication map. For example, Alice's public key is the point $[n_A] \cdot P$, which is the image of P under the scalar multiplication map $[n_A]$. In isogeny-based cryptography, the public keys and shared secret will no longer be a single image point like in ECDH. The public keys will each include the image curve under an isogeny map (the shared secret is a bit more involved, as we will see later). For this reason it is important to know what the image of an isogeny looks like.

Proposition 4.4. *Every isogeny maps an elliptic curve onto the image elliptic curve. (In technical terms, every isogeny is surjective.)*

Thus, as the image of an isogeny is itself an elliptic curve, in isogeny-based cryptography, the public keys will be (image) elliptic curves—see Table 1. For a technical reason that we will explain in Section 5, each party's public key will also include a couple of extra points.

Shared Secret—Replacing a Point by a j-Invariant

Recall that in ECDH, the two parties calculate exactly the same point ($[n_A] \cdot [n_B] \cdot P = [n_B] \cdot [n_A] \cdot P$ in Figure 8) to extract the shared secret. Unfortunately, in isogeny-based cryptography, the image elliptic curves (E_{AB} and E_{BA} in Figure 9) that the two parties calculate during the shared secret step might not be identical. Although the image elliptic curves are different—in the sense that they are defined by different polynomial equations—they are still structurally essentially the same. We can formalize this notion of "structurally identical" using the following definition.

Definition 4.5. An isogeny between two elliptic curves $\phi : E_0 \to E_1$ is called an *isomorphism* if it is a one-to-one function, (also known as an injection). Two elliptic curves E_0 and E_1 are *isomorphic* if there is an isomorphism between them.

Remark 4.6. It is a well-known fact that two curves are isomorphic if and only if they have an inverse function between them that is also an isogeny. This fact has the consequence that two isomorphic elliptic curves have the same curve structure and addition structure.

We now introduce a number, called the j-invariant, that we attach to each elliptic curve. This number can be used to determine if two elliptic curves are isomorphic.

Definition 4.7. Given an elliptic curve $E : y^2 = x^3 + ax + b$, the *j-invariant* of E is the value

$$j(E) = \frac{6912a^3}{4a^3+27b^2}.$$

Proposition 4.8. *Two elliptic curve are isomorphic (over the algebraic closure) if and only if they have the same j-invariant.*

Although, the image elliptic curves (E_{AB} and E_{BA}) that the two parties calculate in SIDH during the shared secret step might not be identical, they are always isomorphic. Since Proposition 4.8 gives a simple way to detemine if two elliptic curves are isomorphic, the shared secret in isogeny-based cryptography is chosen to be the j-invariant of the image elliptic curves that the two parties calculate—see Table 1 and Figure 10.

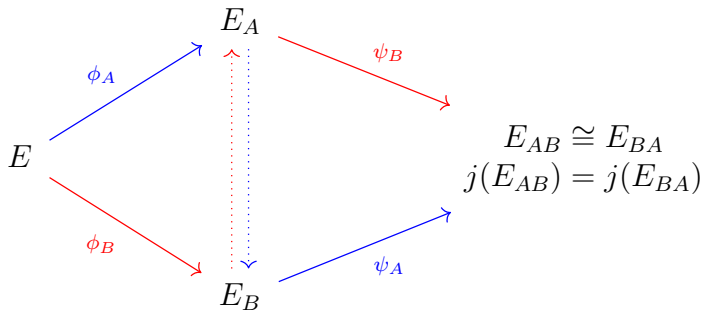

Figure 10: SIDH

Constructing the Private/Public Keypair

To use isogenies in a cryptographic algorithm, we need to be able to choose random isogenies as private keys. Efficiently randomizing these private keys in isogeny-based cryptography is done using the notion of the kernel of an isogeny, which we will now introduce.

Definition 4.9. The *kernel* of an isogeny $\phi : E_0 \to E_1$, denoted $\ker \phi$, is the set of points that map to the identity of E_1, namely,

$$\ker \phi = \{P \mid P \in E_0 \text{ and } \phi(P) = \mathcal{O}\}.$$

Intuitively, we can think of the kernel as the inverse image of $\mathcal{O}$.

As multiplication by n is a very important isogeny, we give a separate definition for its kernel.

Definition 4.10. Let E be an elliptic curve. The set of points P (in the algebraic closure) that lie on E which satisfy $[n] \cdot P = \mathcal{O}$ is called the *n-torsion group* and is denoted $E[n]$.

By the definition of the scalar multiplication map, $\ker[n] = E[n]$.

As we will see in Figure 11, the kernel is important because it gives a simple way to define maps that are not scalar multiplication. This is possible because of the following lemma.

Lemma 4.11. *If two isogenies have the same kernel, then the image curves are isomorphic.*

The object in the following definition will allow Alice and Bob to generate isogenies.

Definition 4.12. The set *generated* by points $P_1, \ldots, P_k$ on an elliptic curve E is the set of points of the form $[m_1] \cdot P_1 + \cdots + [m_k] \cdot P_k$, for some integers $m_1, \ldots, m_k$. We denote this set by $\langle P_1, \ldots, P_k \rangle$. An isogeny whose kernel is generated by a single point is called a *cyclic* isogeny.

To compute her SIDH key pair, Alice will first randomly choose a point R_A of a certain “size" (technically, by “size" we mean order). Alice then calculates a cyclic isogeny ϕ_A whose kernel is $\langle R_A \rangle$ using well-known explicit formulas [7, Section 12.3]. The point R_A, or the associated isogeny ϕ_A, will be Alice’s private key. The image E_A of ϕ_A will be part of Alice’s public key. See Figure 11 for a step-by-step description.

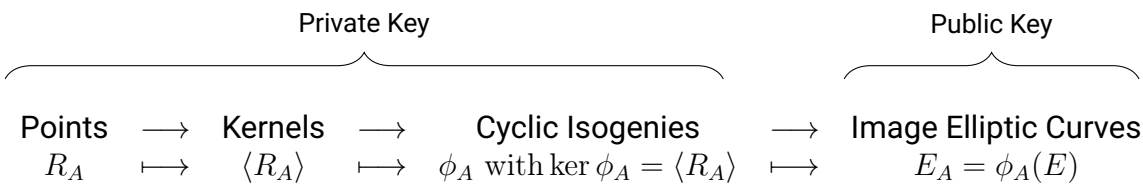

Figure 11: Generating Isogenies from a Point

Bob also chooses a random point R_B and uses this point to construct an isogeny ϕ_B whose kernel is $\langle R_B \rangle$. The point R_B, or the associated isogeny ϕ_B, will be Bob's private key. The image E_B of ϕ_B will be part of Bob's public key.

The question now remains, "how can we randomly choose a point?" To understand the answer to this question it is important to understand the structure of the points on the elliptic curve.

5 Explicit Constructions

In this section we would like to explicitly construct private/public key pairs and shared secrets. This requires us to understand the set of points on an elliptic curve which we can use to generate an isogeny.

Points in a Finite Field on an Elliptic Curve

Elliptic curve cryptography is done over finite spaces where we can perform arithmetic. These spaces are called finite fields. The finite fields that are used in this document are $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, where p is a prime number. Classical elliptic curve cryptography generally uses $\mathbb{F}_p$, whereas quantum-safe isogeny-based cryptography uses $\mathbb{F}_{p^2}$.

Recall that $\mathbb{F}_p$ is the set of integers modulo p, for some prime p. However, $\mathbb{F}_{p^2}$ is not precisely the integers modulo p^2. Intuitively, $\mathbb{F}_{p^2}$ is a type of 2-dimensional version of $\mathbb{F}_p$. More explicitly, if p is congruent to 3 modulo 4, then we can think of $\mathbb{F}_{p^2}$ as simply being the numbers $a + bi$ where $i^2 = -1$ and a and b are numbers in $0, 1, \ldots, p-1$ modulo p. (In isogeny-based cryptography we will always choose p such that $p \equiv 3 \pmod 4$.) For example, in $\mathbb{F}_{7^2}$ we see that

$$(3i)(3+2i) = 9i + 6(i^2) = 9i + 6(-1) = -6 + 9i = -6 + 2i.$$

In isogeny-based cryptography, we will work with an elliptic curve E over $\mathbb{F}_{p^2}$, for some prime p of the form $p = 2^a 3^b - 1$, where a and b are natural numbers. For example, $p = 2^{372}3^{239} - 1$ is one of the primes used in isogeny-based proposal to the NIST quantum-safe cryptography project [3].

As we are interested in isogeny-based cryptography, we would like to have a method to describe all the points with x and y values in $\mathbb{F}_{p^2}$ on an elliptic curve E. We actually only need to understand the points in $E[2^a]$ and $E[3^b]$ whose x and y values are in $\mathbb{F}_{p^2}$. We will discuss the reason for picking points in these torsion groups in Remark 5.1.

Both the sets $E[2^a]$ and $E[3^b]$ have a nice structure. In particular, there exist two points P_A and Q_A on E that generate the set $E[2^a]$. This means that all the points in $E[2^a]$ can be described as $[n] \cdot P_A + [m] \cdot Q_A$, for some n and m in $\mathbb{F}_{p^2}$. (Actually there are many choices for P_A and Q_A, so we just fix one choice as system parameters).

Similarly, there are two points P_B and Q_B on E that generate the set $E[3^b]$. (Again, there are many choices for P_B and Q_B, so we just fix one choice).

Constructing the Private/Public Keypair from a Seed

Suppose that Alice and Bob would like to use isogeny-based cryptography to agree on a secret key. Recall from Figure 11 that Alice and Bob each wish to generate an isogeny from a random point. In particular, Alice will choose a random seed $0 \leq r_A < 2^a$. Using this she will construct the (random) point $R_A = P_A + [r_A] \cdot Q_A$ in the torsion subgroup $E[2^a]$. (Varying the m in description $[n] \cdot P_A + [m] \cdot Q_A$ of a general 2^a-torsion point gives enough possible points that we do not need to vary n as well.) Alice's public key will include the image curve E_A of the isogeny ϕ_A whose kernel is $\langle R_A \rangle$—see Figure 12. Alice's private key can be thought of as r_A, or the associated isogeny ϕ_A.

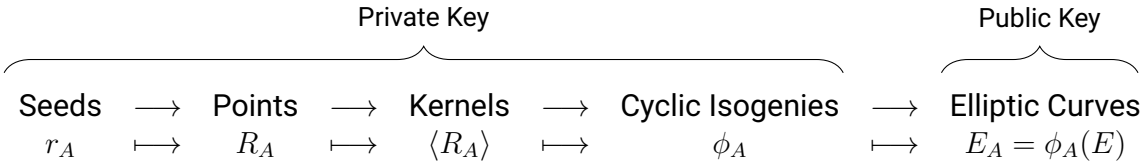

Figure 12: Generating Isogenies from a Seed

Similarly, Bob chooses a random seed r_B and uses it to compute a (random) point $R_B = P_B + [r_B] \cdot Q_B$. Bob's public key will include the image curve E_B of the isogeny ϕ_B whose kernel is $\langle R_B \rangle$. Bob's private key can be thought of as r_B, or the associated isogeny ϕ_B.

Remark 5.1. The kernels of the isogenies are of size 2^a and 3^b, because it is easier to calculate isogenies if the size of their kernels is a power of a small prime. Additionally, for security we also want the sizes of Alice's and Bob's kernels to be relatively prime and reasonably large (there is a correspondence between the sizes of a and b and the corresponding amount of security of the protocol).

Computing the Shared Secret

We want to construct a system of maps as illustrated in Figure 10, where $j(E_{AB}) = j(E_{BA})$. In particular, if we let ψ_A be the isogeny with kernel $\langle \phi_B(R_A) \rangle$, then a simple mathematical argument proves

$$\psi_A \circ \phi_B = \langle R_A, R_B \rangle.$$

Similarly, if we let ψ_B be the isogeny with kernel $\langle \phi_A(R_B) \rangle$, then

$$\psi_B \circ \phi_A = \langle R_A, R_B \rangle.$$

By Lemma 4.11, as $\ker \psi_A \circ \phi_B = \ker \psi_B \circ \phi_A$, this proves

$$E_{AB} = \psi_A \circ \phi_B(E) \cong \psi_B \circ \phi_A(E) = E_{BA}.$$

Therefore, by Proposition 4.8, $j(E_{AB}) = j(E_{BA})$—see Figure 10. Thus, if Alice and Bob follow this procedure to create ψ_A and ψ_B, respectively, then they will end up with a shared secret, namely, $j(E_{AB}) = j(E_{BA})$.

In order for Alice to construct ψ_A, she will need to know $\phi_B(R_A)$. For this reason, in addition to E_B, Bob's public key also includes $\phi_B(P_A), \phi_B(Q_A)$. More explicitly, having been given $\phi_B(P_A), \phi_B(Q_A)$, Alice can construct

$$\phi_B(R_A) = \phi_B(P_A) + [r_A] \cdot \phi_B(Q_A).$$

Next Alice computes the image E_{AB} of the isogeny ψ_A whose kernel is $\langle \phi_B(R_A) \rangle$. Finally, she calculates the j-invariant of E_{AB}, which will be the shared secret. This process is summarized in Figure 13.

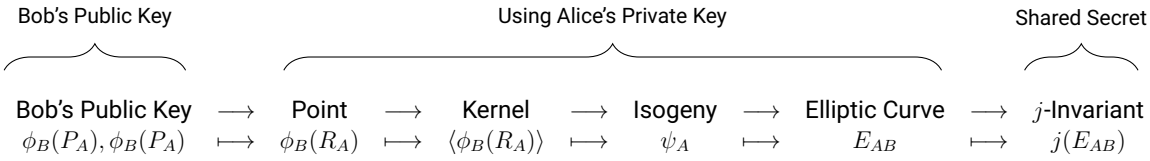

Figure 13: Generating a Shared Secret

Similarly, Alice's public key is $E_A, \phi_A(P_B), \phi_A(Q_B)$. Having been given $\phi_A(P_B)$ and $\phi_A(Q_B)$, Bob can construct

$$\phi_A(R_B) = \phi_A(P_B) + [r_B] \cdot \phi_A(Q_B).$$

Next Bob computes the image E_{BA} of the isogeny ψ_B whose kernel is $\langle \phi_A(R_B) \rangle$. He calculates the j-invariant of E_{BA}, which is the shared secret. This more detailed protocol is depicted in Figure 14.

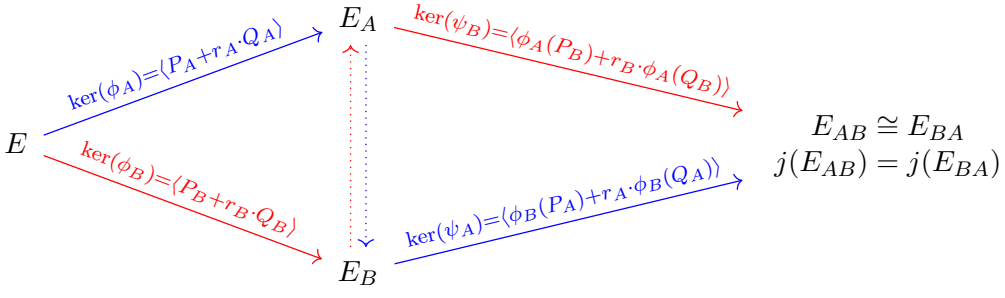

Figure 14: SIDH

Combining these ideas we can describe the full SIDH protocol.

6 Supersingular Isogeny Diffie–Hellman (SIDH)

In 2011, David Jao and Luca De Feo introduced SIDH, an isogeny-based version of ECDH [2], [1]. To instantiate the protocol, first fix the elliptic curve E. This elliptic curve is chosen so that the following algorithm is not susceptible to the same type of quantum attack—like the one by Shor—that makes ECDH vulnerable. In particular, we can choose $E : y^2 = x^3 + ax + b$, defined in terms of a field $\mathbb{F}_{p^2}$ for some prime $p = 2^a 3^b - 1$ (although any supersingular elliptic curve would do—see [5] for a definition of supersingular). To agree on a shared secret using SIDH, Alice and Bob complete the following steps:

Public Parameters: A (supersingular) elliptic curve $E : y^2 = x^3 + ax + b$ defined in terms of a field $\mathbb{F}_{p^2}$ for some prime $p = 2^a 3^b - 1$, where a and b are natural numbers. A basis P_A, Q_A of $E[2^a]$ and a basis P_B, Q_B of $E[3^b]$.

Key Generation: Alice begins by choosing a random number r_A between 0 and $2^a - 1$. She calculates the isogeny ϕ_A whose kernel is generated by $R_A = P_A + [r_A] \cdot Q_A$ using well-known explicit formulas [7, Section 12.3]. Alice then sends the image curve $E_A = \phi_A(E)$ to Bob. She also sends $\phi_A(P_B)$ and $\phi_A(Q_B)$, which will allow Bob to construct his next map ψ_B. Alice's private key is the seed r_A, or equivalently, the isogeny ϕ_A. Alice's public key is $(E_A, \phi_A(P_B), \phi_A(Q_B))$.

Similarly, Bob chooses a random number r_B between 0 and $3^b - 1$. He calculates the isogeny ϕ_B whose kernel is generated by $R_B = P_B + [r_B] \cdot Q_B$. Bob's private key is the seed r_B, or equivalently, the isogeny ϕ_B. Bob's public key is $(E_B, \phi_B(P_A), \phi_B(Q_A))$.

Shared Secret: Using the points that she receives from Bob, Alice computes an isogeny ψ_A on E_B whose kernel is generated by $\phi_B(R_A) = \phi_B(P_A) + [r_A] \cdot \phi_B(Q_A)$. Alice then calculates the j-invariant of the image curve $E_{AB} = \psi_A(E_B)$. Similarly, using the points that he receives from Alice, Bob computes an isogeny ψ_B on E_A whose kernel is generated by $\phi_A(R_B) = \phi_A(P_B) + [r_B] \cdot \phi_A(Q_B)$. Bob then calculates the j-invariant of the image curve $E_{BA} = \psi_B(E_A)$. Since E_{AB} and E_{BA} are isomorphic, they have the same j-invariant. More specifically,

$$j(\psi_B(E_A)) = j(E_{BA}) = j(E_{AB}) = j(\psi_A(E_B)).$$

Thus, they can use the j-invariants as a shared secret.

Figures 15 summarizes SIDH. However, we can also think of it more mathematically, see Figure 14. These diagrams are the analogue of Figure 7 and Figure 8, respectively. In both figures, Alice calculates the arrows in red, and Bob calculates the arrows in blue. In Figure 9 , the dotted lines represent the parties exchanging the public keys $\phi_A(E), \phi_A(P_B), \phi_A(Q_B)$ and $\phi_B(E), \phi_B(P_A), \phi_B(Q_A)$.

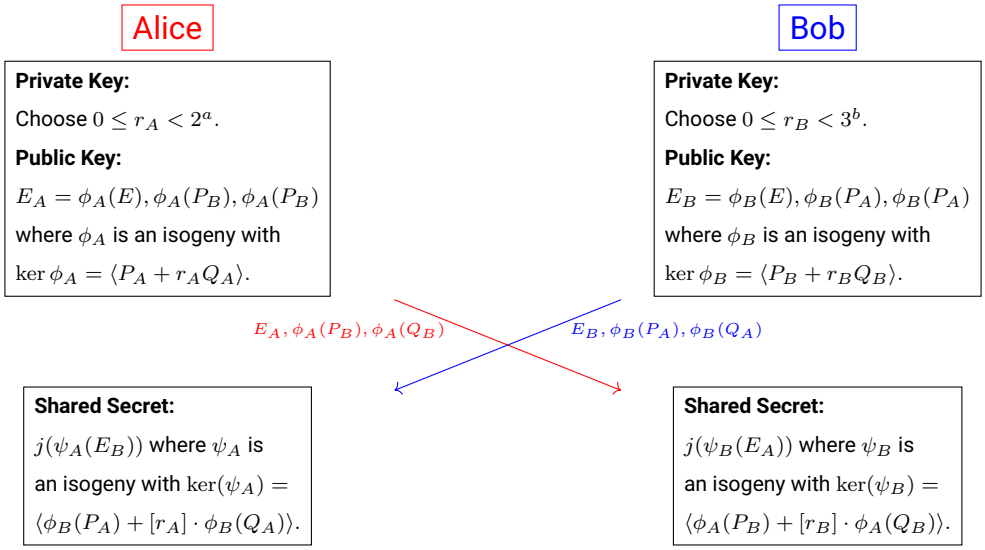

Figure 15: SIDH

For SIDH to be secure the following problem must be hard:

Definition 6.1. Suppose there exists an isogeny ϕ between two elliptic curves E and E', where the kernel of this isogeny has size l^e. The *l^e-isogeny problem* is the problem of finding the kernel of ϕ given only E and E'.

To ensure an adversary cannot compute Alice's private key from Alice's public key, it is important for the 2^a-isogeny problem to be hard. Similarly, to ensure an adversary cannot compute Bob's private key from Bob's public key, it is important for the 3^b-isogeny problem to be hard. However, the l^e-isogeny problem has been studied by number theorists for over 20 years, and is believed to be hard for large enough powers of l, for any natural number l.

As the public keys include extra points, the security of SIKE is actually based on a variant of the l^e-isogeny problem. This variant is still believed to be hard for large powers of l.

Unlike ECDLP, the l^e-isogeny problem is believed to be secure even if an adversary had access to a large-scale quantum-computer. A key encapsulation mechanism (KEM) based on SIDH titled Supersingular Isogeny-Based Key (SIKE) is currently a promising candidate for the NIST's standardization process for post-quantum cryptography. For further details on this proposal, see [3].

One concern to keep in mind is that, unlike with ECDH, if both parties use long-term (static) public keys, then SIDH is insecure. Since the framework of KEMs only requires one party to have a long-term private/public key pair, this concern does not apply to SIKE. For this and other security reasons we recommend using the specifications given in [3].

In this document, we presented the introductory mathematics behind isogeny-based cryptography, and we showed how this mathematics could be used to provide a key establishment protocol (SIDH) that is similar in structure to ECDH. Despite the expected downfall of classical cryptography including ECDH, isogeny-based cryptography demonstrates that there are promising alternatives to provide security into the quantum age.

7 About ISARA Corporation

ISARA Corporation, the world's leading provider of agile quantum-safe security solutions, leverages decades of real-world cybersecurity expertise to protect today's computing ecosystems in the quantum age. With our partners, we're clearing the path to quantum-safe security for enterprises and governments by delivering practical, standardized solutions for a seamless migration. Co-founded in 2015 by former BlackBerry security executives, our team has launched several first-of-its-kind solutions such as a quantum-safe cryptographic library, integration tools for developers, and agile technologies. With an emphasis on interoperability, we proudly collaborate on international standards-setting efforts. For more information, visit www.isara.com or follow @ISARACorp on Twitter.

References

[1] 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.

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

[3] David Jao et al. Supersingular isogeny key encapsulation. *NIST Round 1 Submissions for Post-Quantum Cryptography Standardization*, November 30, 2017.

[4] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing*, 26(5):1484–1509, 1997.

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

[6] Joseph H Silverman and John Torrence Tate. *Rational points on elliptic curves*, volume 9. Springer, 1992.

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