Supersingular Isogeny-based Cryptography: A Survey

Philipp STRATIL1, Shingo HASEGAWA2* and Hiroki SHIZUYA2

1Department of Informatics, Technical University of Munich, Germany
2Center for Data-driven Science and Artificial Intelligence, Tohoku University, Sendai 980-8576, Japan

With the advent of quantum computers that showed the viability of Shor’s Algorithm to factor integers, it became apparent that asymmetric cryptographic algorithms might soon become insecure. Since then, a large number of new algorithms that are conjectured to be quantum-secure have been proposed, many of which come with non-negligible trade-offs compared to current cryptosystems. Because of this, both research and standardization attempts are ongoing effort.

In this survey, we describe one of the most promising approaches to post-quantum cryptography: cryptosystems based on supersingular isogenies. Building on top of isogenies is promising not only because they have been a well-studied topic for many decades, but also because the algorithms proposed in recent literature promise decent performance at small key sizes, especially compared to other post-quantum candidates.

After introducing the basic mathematical backgrounds required to understand the fundamental idea behind the use of supersingular isogenies as well as their relation to elliptic curves, we explain the most important protocols that have been proposed in recent years, starting with the so-called Supersingular Isogeny Diffie–Hellman. We discuss the novel approaches to well-established protocols that supersingular isogeny-based schemes introduce, analyze why it is difficult to translate certain cryptographic schemes into the supersingular isogeny case and argue that while the discussed cryptographic schemes promise to be both performant and quantum-secure, they instead introduce a trade-off in the form of increased protocol complexity.

KEYWORDS: isogeny cryptography, Supersingular Isogeny Diffie–Hellman

1. Introduction

Today, almost all public-key cryptosystems depend in one way or another on the hardness of the discrete logarithm problem for their security. One of the most famous examples is the Diffie–Hellman protocol [12] for secure key exchange over an insecure channel, which is, without a doubt, one of the most important cryptographic protocols of our time. Its security is based on the so-called Diffie–Hellman problem (DHP), which, at its core, assumes computing an element \(g^x\) within a group \(G\) knowing only \(g, g^x\) and \(g^y\) to be infeasible with current technology. The Diffie–Hellman problem can easily be reduced to the discrete logarithm problem of finding an \(x\) such that \(g^x = y\) for a given generator \(g\) and a random element \(y\) of a group \(G\).

As long as it fulfills the underlying assumptions made by the protocol, the group \(G\) in the Diffie–Hellman protocol can be seen as interchangeable. Popular choices are the multiplicative group of integers \((\mathbb{Z}/p\mathbb{Z})^*\) modulo a prime \(p\) as well as cyclic subgroups of elliptic curves defined over finite fields \(\mathbb{F}_q\). While it is the discrete logarithm problem that is considered hard in the former group, the latter has its own elliptic curve discrete logarithm problem. While several classical algorithms to solve the discrete logarithm problem exist, none of them are efficient, with the best one, the general number field sieve, still requiring sub-exponential time.

In 1994, mathematician Peter Shor published a paper [28] in which he described an efficient algorithm for integer factorization, leveraging the power of a theoretical quantum computer. The algorithm, subsequently named Shor’s algorithm, works in two stages: a classical part that can be implemented on a normal computer, and a quantum part that is specifically designed to run on a quantum computer. The latter part works by reducing the problem of factoring an integer \(N\) to the problem of finding the order of random elements in the field \((\mathbb{Z}/N\mathbb{Z})^*\) and makes heavy use of the quantum Fourier transform. The requirements for Shor’s algorithm to work are quite simple: the group \(G\) that \(N\) belongs to has to be finite and abelian.

Not only can Shor’s algorithm be used to efficiently factor integers, it can also be adapted to solve the discrete logarithm problem in polynomial time. Since protocols like Diffie–Hellman require the group \(G\) to be cyclic, \(G\) is automatically abelian and hence vulnerable to Shor’s algorithm. The result is that, once large-scale quantum computer become a reality, all protocols relying on a variant of the discrete logarithm problem would no longer be secure. In
2001, researchers at IBM created a small-scale quantum gate computer with 7 qubits [33] that successfully factored the number $15 = 3 \times 5$, proving the algorithm's viability. In 2018, researchers successfully factored $376289 = 571 \times 659$ by a quantum annealing machine [23], the highest number to date factored with Shor's algorithm.

While these results might seem pointless when comparing them to the size of the numbers usually used in cryptographic contexts, it shows that it is only a matter of time until quantum computers large enough to break currently used protocols surface. Because of this, research in the area of post-quantum cryptography is an actively ongoing effort. In early 2017, the National Institute of Standards and Technology (NIST) announced a call for proposals of quantum-secure algorithms [25] in an attempt to create standardized and future-proof cryptographic specifications. This approach is similar to how Rijndael was selected to be what is today known as the Advanced Encryption Standard (AES), the most used symmetric block cipher in the world. Until the deadline in November 2017, 82 candidates were submitted and 69 were accepted for the first round. On January 30, 2019, 26 candidates were forwarded to the second round.

Among the candidates is the key encapsulation algorithm SIKE, which has proceeded to the second round. SIKE is one example of several recent cryptographic schemes based on the properties of homomorphisms between supersingular elliptic curves, so-called supersingular isogenies. While still based on top of the theory of elliptic curves, supersingular isogeny-based cryptography does not depend on the hardness of the elliptic curve discrete logarithm problem for security. In fact, the resulting commutative structure all protocols in this area take advantage of is neither cyclic nor abelian. In order to still uphold the commutativity aspect required by asymmetric protocols like Diffie–Hellman, supersingular isogeny-based schemes introduce a novel approach that has never been seen in previous protocols: in addition to the fixed public key parameters, the participants in the protocol compute and reveal certain auxiliary information, based on their own secret and the other parties' public parameters. This kind of approach naturally increases the attack surface of the protocol and opens the door to many new interesting types of cryptoanalysis.

Cryptosystems based on isogenies is proposed by Couveignes in [8]. His cryptosystem is based on computing isogenies between ordinary elliptic curves. Rostovtsev, Stolbunov [27] and Stolbunov [30] also considered cryptographic protocols using isogenies between ordinary elliptic curves. For the supersingular case, Charles, Lauter, Goren [5] first proposed a cryptosystem which uses isogenies between supersingular elliptic curves. They constructed a collision-resistant hash function whose collision resistance reduces to the difficulty of computing isogenies between supersingular elliptic curves. From the appearance of the novel key exchange protocol by Jao and De Feo [22], the supersingular isogeny-based cryptography attracted much more attention. Currently, there exists many cryptographic primitives in the supersingular isogeny-based cryptography. As major cryptographic primitives, public key encryption [11], zero-knowledge interactive proof [11], signature schemes [17] are constructed based on the supersingular isogeny.

The rest of this paper is structured as follows: in Sect. 2, we try to explain the bare minimum of mathematical backgrounds required to understand isogenies in general and in the supersingular case in particular. We also review some basic notions of cryptography used throughout the paper. In Sect. 3, we introduce the Supersingular Isogeny Diffie–Hellman (SIDH) due to Jao and De Feo. In Sect. 4, we describe the ElGamal-like public key encryption scheme due to De Feo, Jao and Plût that can be derived from SIDH. We additionally analyze why it is difficult to translate other public-key encryption schemes like Cramer–Shoup into the supersingular isogeny case. Finally, in Sect. 5, we introduce the Supersingular Isogeny Key Encapsulation (SIKE) protocol that was submitted to NIST's post-quantum cryptography contest and discuss how it tries to solve some of the shortcomings of both the SIDH and the derived PKE. Lastly, in Sect. 6, we draw our conclusions on the approaches that supersingular isogeny-based schemes take on already existing cryptosystems and what this means for post-quantum cryptography as a whole.

2. Preliminaries

In this section, we explain the bare minimum of required mathematical foundations such as algebra, number theory, elliptic curve arithmetic and public-key cryptography in a hopefully intuitive way and without any previous requirements. For a more complete but still reasonably short introduction to the topic we refer to De Feo’s Mathematics of Isogeny Based Cryptography [10]. As a resource for learning more about public-key cryptography we recommend [15], for more information about elliptic curve arithmetic we recommend [29]. Other useful resources include [7] and [3]. This section borrows heavily from the mentioned sources and for more in-depth information, we recommend the reader to refer to them instead.

2.1 Notions of security

Cryptographic schemes that find applications in most people’s day-to-day lives always try to solve at least one of three main problems: ensuring authenticity, confidentiality and integrity. Integrity ensures that a message has not been tampered with. An integrity-protecting scheme guarantees that the recipient of the message receives the message in the way the sender intended. An example of a cryptographic scheme trying to solve this problem are cryptographic hash functions.
**Authenticity** ensures that the message is from the person the sender is pretending to be. An authenticity-protecting scheme allows the recipient to confirm that the identity of the sender of a message. Signature schemes try to solve this issue.

**Confidentiality** ensures that a message cannot be read by unrelated third-parties on its way from sender to recipient. A confidentiality-protecting scheme allows sender and recipient to exchange messages over an unprotected channel without anyone else being able to understand their contents. This is what encryption schemes aim to do.

While all three properties are very closely related, a cryptographic scheme that only implements one of them does not automatically implement the other two.

Encryption schemes can further be divided into two different categories: symmetric and asymmetric encryption schemes. If the sender and the recipient are the same person, then symmetric schemes are often sufficient: in a symmetric scheme, also called private key scheme, only one static key exists, that is both used for encryption and decryption. Since this requires that sender and recipient both have common knowledge of the secret used to encrypt the data, symmetric schemes are unpractical in situations where two participants would have to first exchange the secret over an insecure channel.

Asymmetric schemes, also called public-key encryption schemes, on the other hand use different keys for encryption and decryption: one private key, which is only known to the recipient, and one public key, which is known to everyone as the name suggests. Some confidentiality-ensuring public-key encryption schemes can derive signature schemes that guarantee integrity and authenticity in an asymmetric manner.

Since asymmetric encryption schemes only work by disclosing information in form of the public key, implementing secure schemes that do not leak any information about the corresponding private key is a difficult task. Proving that a scheme is secure under these circumstances is even more difficult, if not impossible. Due to this, all asymmetric cryptographic schemes are based on the difficulty of hard mathematical problems. In cryptographic contexts, these problems are often used in the form of a “game” in which an adversary interacts with a black box and has to achieve a certain goal like forging a message or distinguishing two values. If the adversary’s probability of success in distinguishing two values is negligibly more than 50%, then he might guess the correct solution only, and the scheme is considered to be secure.

Since confidentiality, authenticity and integrity are closely related, most cryptographic applications used in the real world combine different cryptographic schemes in order to achieve all three goals. Since cryptographic schemes are complex systems, many different approaches in order to attack them exist. In fact, if an adversary has the ability to modify an encrypted message before it arrives at the recipient, it might give them an advantage in their attempt to decrypt the message. In other words, the lack of the integrity property in an encryption scheme might weaken its confidentiality property. In order to account for possible attacks of this kind, different kinds of attack models exist, all of which describe the resilience of an encryption scheme against an adversary with different capabilities. The procedure is always the same: the adversary interacts with a black box, the so-called oracle $O$, over two phases. In the first phase, the adversary prepares their attack by querying the oracle on arbitrary inputs. Based on the knowledge gained, the adversary then generates two plaintexts $m_0, m_1$, and hands them to the oracle. The oracle chooses a random $b \in \{0, 1\}$, encrypts the plaintext $m_b$, and gives the resulting ciphertext $c$ to the adversary, who now has to determine if $c$ is an encryption of $m_0$ or $m_1$. If the adversary’s success probability is only negligibly more than 50%, that is, the best he can do is guess randomly, then the encryption scheme is called indistinguishable in the chosen attack model.

The most commonly used attack models are as follows:

- **Chosen Plaintext Attack (CPA)** describes an attack model in which the adversary has the ability to query the oracle for encryptions of arbitrary plaintexts. They can only query each plaintext once, so as to not decrypt the challenge ciphertext in the first phase. This attack model essentially describes an adversary with access to the public key. It is obvious that a public key encryption scheme that is not at least CPA-secure provides no security at all.

- **Chosen Ciphertext Attack (CCA1)**, also called lunchtime attack, describes an attack model in which the adversary can decrypt arbitrary ciphertexts by querying the oracle during the first phase. Once they received the challenge ciphertext and the second phase begins, they are not allowed to send further queries to the oracle.

- **Adaptive Chosen Ciphertext Attack (CCA2)** describes an even stronger adversary than in the CCA1 scenario. They are now also allowed to query the oracle for arbitrary ciphertexts in the second phase, after receiving the challenge ciphertext. The only restriction that applies is that they can not ask the oracle to decrypt the challenge ciphertext itself.

If an encryption scheme is indistinguishable under the CCA2 attack model, it is said to be IND-CCA2 or just IND-CCA (IND-CCA1 and IND-CPA for the CCA1 and CPA attack models, respectively). IND-CCA2 security is currently the highest level of security that an asymmetric encryption scheme can achieve and is used as the gold standard for new encryption schemes.

In fact, it is possible to create an IND-CCA2-secure scheme from any other IND-CPA secure scheme by employing an Encrypt-then-MAC scheme. The term MAC stands for message authentication code and describes a cryptographic scheme that implements both the integrity and authenticity properties discussed earlier. A MAC provides a signing function that takes a message and a secret parameter as input and creates a so-called tag that is only valid for the supplied message. Later, the MAC’s verify function can be used with the same key on the tag and the message in order to confirm that the message has neither been modified nor been replaced with a message by someone else.
Namprempre and Bellare showed in [1] that including a tag of the encrypted message in the ciphertext during encryption and verifying said tag before decrypting the message results in an IND-CCA2-secure encryption scheme provided that encryption and decryption on its own are IND-CPA and that the MAC is unforgeable. Unfortunately, Encrypt-then-MAC schemes oftentimes have poor performance, which limits their usefulness in resource-restricted environments.

2.2 Groups, fields and morphisms

Groups and fields are the basic algebraic structures that many asymmetric cryptographic schemes are based on. Although we are not going to go into much detail of them in this section, for the sake of completeness, we are going to quickly review some important definitions before going into the more complicated matter.

Definition 2.1 (Groups). Let $M$ be a set of elements together with a binary map $\oplus : M \times M \rightarrow M$, $(\sigma, \tau) \mapsto \sigma \oplus \tau$ that maps two elements from $M$ to another element in $M$. $G = (M, \oplus)$ is called a group if $\oplus$ has the following properties:

\[
\forall \sigma, \tau, \rho \in M : \quad (\sigma \oplus \tau) \oplus \rho = \sigma \oplus (\tau \oplus \rho),
\]

\[
\exists e \in M, \forall \sigma \in M : \quad e \oplus \sigma = \sigma \quad \text{and} \quad \sigma \oplus e = \sigma,
\]

\[
\forall \sigma \in M, \exists \hat{\sigma} \in M : \quad \sigma \oplus \hat{\sigma} = \hat{\sigma} \oplus \sigma = e.
\]

Additionally, $G$ is called an abelian group if

\[
\forall \sigma, \tau \in M : \quad \sigma \oplus \tau = \tau \oplus \sigma.
\]

$e$ is called the neutral element or the identity element of the group and $\hat{\sigma}$ is called the inverse element of $\sigma$ and is often also written as $\sigma^{-1}$.

If the set $M$, the group $G$ is defined over, has finitely many elements, we call $G$ a finite group. In a finite group, the number of times an element $\sigma$ has to be mapped by $\oplus$ repeatedly until arriving at the neutral element $e$ is called the order $\text{ord}(\sigma)$ of $\sigma$. Since a group only has one operation $\oplus$, we often omit the operator sign and write $\sigma \tau$ for $\sigma \oplus \tau$. One example for a (non-finite) group is the group of integers $\mathbb{Z}$.

A subgroup $H \subseteq G$ of a group $G = (M, \oplus)$ is any subset $\hat{M} \subseteq M$ of the set $M$ that the group $G$ is defined over such that $H = (\hat{M}, \oplus)$ still fulfills all three group axioms with the same operation $\oplus$ of $G$. The smallest subset $g \subseteq M$ that can be used to construct all other elements in the group $G$ using linear combinations with $\oplus$ is called the generator of the group $G$. We also write $\langle g \rangle = G$ to indicate that $g$ generates $G$. If $g$ only consists of a single element, then the generated group $G$ is called cyclic. All cyclic groups are abelian groups by its definition.

Instead of only looking at binary maps that map from a group $G$ to itself, we can also look at maps from one group $G$ to another group $G'$. If such a map is structure-preserving, it is called a morphism.

Definition 2.2 (Morphisms). Let $G$ and $G'$ be groups.

- A homomorphism is a map $\phi : G \rightarrow G'$ with $\phi(\sigma \tau) = \phi(\sigma)\phi(\tau)$ for all $\sigma, \tau \in G$.
- The kernel $\ker(\phi) = \{ \sigma \in G : \phi(\sigma) = \iota_{G} \}$ is the set of all elements in $G$ that get mapped to the neutral element $\iota_{G}$ in $G'$.
- An isomorphism is a bijective homomorphism, i.e., $|G| = |G'|$ and $\ker(\phi) = \{ \iota \}$.
- An endomorphism is a homomorphism $\phi : G \rightarrow G$ that maps from the group $G$ to itself, i.e., $G = G'$.

If an isomorphism between two groups $G$ and $G'$ exists, the groups are called isomorphic. The set of all isomorphic groups is called an isomorphism class.

Based on groups, we can define fields.

Definition 2.3 (Fields). Let $G$ be a set with two binary maps $\oplus : G \times G \rightarrow G$ and $\otimes : G \times G \rightarrow G$. $(G, \oplus, \otimes)$ is called a field if

- $(G, \oplus)$ forms an abelian group with neutral element 0,
- $(G \setminus \{0\}, \otimes)$ forms an abelian group with neutral element 1, and
- $\forall \sigma, \tau, \rho \in G$:

\[
\sigma \otimes (\tau \oplus \rho) = \sigma \otimes \tau \oplus \sigma \otimes \rho \quad \text{and} \quad (\sigma \oplus \tau) \otimes \rho = \sigma \otimes \rho \oplus \tau \otimes \rho.
\]

$(G, \oplus)$ and $(G \setminus \{0\}, \otimes)$ are called the additive group and the multiplicative group of the field, respectively.

Similarly to groups, a field with finitely many elements is called a finite field. For a finite field with $q$ elements, where $q$ is a prime power, we write $\mathbb{F}_q$. Every finite field with $q$ elements is isomorphic to every other field with $q$ elements. We can thus uniquely identify fields with prime-power order by their number of elements. One example for a (non-finite) field is the field $(\mathbb{R}, +, \cdot)$ of rational numbers with addition and multiplication.
2.3 Elliptic curves

Elliptic curves are a staple technology in today’s world. They are used in many areas of public-key cryptography and are valued for their small size and computational efficiency. An elliptic curve $E$ is defined by a specific equation, and a point $P = (x, y)$ with the coordinates $x$ and $y$ is said to lie on the curve $E$ if the values for $x$ and $y$ satisfy the equation.

There are several different ways to define elliptic curves, with the following being the most popular one.

**Definition 2.4 (Short Weierstrass Equation).** An elliptic curve $E$ defined over a field $K$ (with $\text{char}(K) \notin \{2, 3\}$) is given by the short Weierstrass Equation,

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in K$.

Additionally, $E$ has to be smooth (non-singular), i.e., every point on the curve needs to have a unique tangent.

Figure 1 shows two different curves in Weierstrass form, both of which are defined over the field of the rational numbers $\mathbb{R}$. Additionally, the curve on the left is smooth and hence qualifies as an elliptic curve. The one on the right, however, has a cusp at $(0, 0)$, and it is easy to see that at this point, infinitely many tangents exist. Hence, the curve on the right is singular (not smooth) and thus not an elliptic curve.

When used in cryptography, the field $K$ is usually a finite field with a prime number of elements (denoted by $\mathbb{F}_p$ for $p$ prime) or an extension thereof. A $K$-rational point is a point $P = (x, y)$ with $x, y \in K$ that lies on the curve if it satisfies the equation for $E$.

What makes elliptic curves so interesting is the fact that with the correct definitions, we can use two random points $P, Q$ on $E$ to do some math. Indeed, with the following definitions, we obtain a group structure over the points of an elliptic curve.

**Definition 2.5.** Let $E : y^2 = x^3 + ax + b$ be an elliptic curve. Then an elliptic curve group $E(K)$ is formed by the union of $K$-rational points on $E$ and the neutral element $O$. Let $P = (x, y), Q = (x', y')$ be in $E(K)$. Let $O$ denote the neutral element. We define the group law by the following rules:

- $P \oplus O = O \oplus P = P$,
- $P \oplus -P = -P \oplus P = O$ for $-P = (x, -y)$,
- $P \oplus Q = (\alpha, \beta)$ with
  $$\alpha = \lambda^2 - x - x', \quad \beta = -\lambda x - y + \lambda x$$

where

$$\lambda = \begin{cases} 
\frac{y - y'}{x' - x} & \text{if } P \neq Q \text{ and } \\
\frac{3x^2 + a}{2y} & \text{if } P = Q.
\end{cases}$$

It is clear that when $K$ is a finite field, there are only finitely many points that can lie on $E$. Finding the exact number $\#E$ of points is not easy, however, with Hasse’s theorem, we have an upper bound of $|\#E(K) - q - 1| \leq 2\sqrt{q}$ for a field $K$ with $q$ elements.

Figure 2 intuitively shows the group operation in its geometric representation for the elliptic curve $E : y^2 = x^3 - x + 1$ defined over $\mathbb{R}$. Figure 2(a) displays the addition of two distinct points $P$ and $Q$. A line is drawn through the...
two points and the third intersection with the elliptic curve, here labeled $R$, is mirrored at the $x$ axis, finally giving us the point $P \oplus Q$. Figure 2(b) shows what happens when we add a point $P$ to itself: we draw the tangent line through point $P$ and mirror the resulting intersection with $E$ at the $x$ axis, resulting in point $[2]P$. This demonstrates why we have required $E$ to be smooth (have a unique tangent at every point) — otherwise we would have no way to add a point $P$ to itself. Finally, Fig. 2(c) demonstrates what happens if we add a point to its inverse: here, $Q = -P$. We draw a line through both points and receive the neutral element of our group, the point at infinity $\mathcal{O}$. This also shows why we mirror the point of intersection after drawing a line through two points: if we “draw” a line through $\mathcal{O}$ and $P$, we intersect $E$ at $Q = -P$. If we now mirror $Q$ at the $x$ axis, we return to point $P$, which is exactly what we have required with $P \oplus \mathcal{O} = P$.

Note that due to the fact that elliptic curves in Weierstrass form are equations of degree 3, it is not possible for a random line in the plane to intersect more than three points of the elliptic curve at once.

Until now, we have not specified the neutral element $\mathcal{O}$. Since we are defining a group, $\mathcal{O}$ has to be a valid point on the curve $E$, the same way all other points are. This is achieved by adding a third coordinate $z$ and defining the point at infinity as $\mathcal{O} = (0, 1, 0)$. For all other points $P = (x, y, z) \neq \mathcal{O}$ we set $z = 1$. To accommodate our changes, we modify the equation in the following way:

$$y^2z = x^3 + axz^2 + bz^3.$$}

With this, the point $\mathcal{O} = (0, 1, 0)$ is the only point on the line $Z = 0$. Since we have restricted all other points to have a $z$ coordinate equal to zero, we can set $x = \hat{x}/z$ and $y = \hat{y}/z$ in order to end up with the short Weierstrass equation again.

It is possible to define subgroups of $E$ that are identified by special properties. One such example that will later be useful are the so-called torsion subgroups of an elliptic curve.

---

Fig. 2. A demonstration of the group law on the elliptic curve $E : y^2 = x^3 - x + 1$ defined over $\mathbb{R}$. 

---

(a) Addition of two points $P$ and $Q$  
(b) Addition of a point $P$ with itself  
(c) Point at infinity $\mathcal{O}$
Definition 2.6 (Torsion subgroup). Let $E$ be an elliptic curve and let $m$ be a positive integer. Let $K$ be a field and $\bar{K}$ its algebraic closure. An $m$-torsion point is a point $P \in E(\bar{K})$ of order $m$, i.e., $P$ satisfies $[m]P = \mathcal{O}$. Let $E(K)[m]$ denote the subgroup consisting of $m$-torsion points in $E(K)$. We write $E[m]$ for $E(\bar{K})[m]$. $E[m]$ is called the $m$-torsion subgroup of $E(K)$.

The last thing to note is that it is possible to easily identify an elliptic curve up to isomorphism with the help of the so-called $j$-invariant.

Definition 2.7 ($j$-invariant). Let $E(K)$ be an elliptic curve given by a Weierstrass equation $y^2 = x^3 + ax + b$ defined over a field $K$ with $\text{char}(K) \notin \{2, 3\}$. The $j$-invariant of $E$ is defined as

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

Lemma 2.8. Let $E(K)$ and $E'(K)$ be two elliptic curves defined over the same field $K$. If $E(K)$ and $E'(K)$ are isomorphic over $K$, then $j(E) = j(E')$. The converse is also true if $K = \bar{K}$, where $\bar{K}$ is the algebraic closure of $K$.

For the rest of the paper we will denote an elliptic curve group by $E$ instead of $E(K)$ if the underlying field $K$ is clear.

2.4 Isogenies

The term isogeny, the mathematical construct that all protocols described in this paper are based on, simply refers to a homomorphism between two elliptic curves: an isogeny maps all points from one elliptic curve $E_1$ to another elliptic curve $E_2$.

Definition 2.9 (Isogeny). Let $E_1$ and $E_2$ be elliptic curves defined over a finite field $\mathbb{F}_p$. An isogeny $\phi : E_1 \to E_2$ defined over $\mathbb{F}_p$ is a non-constant rational map defined over $\mathbb{F}_p$ which is also a group homomorphism from $E_1(\mathbb{F}_p)$ to $E_2(\mathbb{F}_p)$.

One example of an isogeny is the identity map $\phi : E \to E$, $\phi(P) = P$. Another easy to understand example is the multiply-by-$n$ $\psi : E \to E$, $\psi(P) = [n]P$, which maps any point $P$ to its multiple. Incidentally, both of these examples are endomorphisms, since they map from $E$ to itself.

Lemma 2.10 (Endomorphism ring). Let $E$ be an elliptic curve defined over some field $K$. Let $\mathcal{I}$ denote the set of all endomorphisms, i.e., all isogenies $\phi : E \to E$ that map from $E$ to itself. The set $\mathcal{I}$ then forms a ring with pointwise addition and composition such that

$$(\phi \oplus \psi)(P) = \phi(P) \oplus \psi(P),$$

$$(\phi \odot \psi)(P) = \phi(\psi(P)).$$

for all $\phi, \psi \in \mathcal{I}$. The neutral elements are the multiply-by-0 and the identity map respectively. The resulting ring is called the endomorphism ring $\text{End}(E)$ of $E$.

Two elliptic curves $E_1$ and $E_2$ are called isogenous if an isogeny $\phi : E_1 \to E_2$ exists. If $E_1$ and $E_2$ defined over the same field $\mathbb{F}_p$ are isogenous, the two curves have the same number of points: $\#E_1(\mathbb{F}_p) = \#E_2(\mathbb{F}_p)$. The degree deg$(\phi)$ of an isogeny $\phi$ is its degree as an algebraic map. If $\phi$ is a separable isogeny [10] then its degree is equal to the size of its kernel ker$(\phi)$. An isogeny of degree $\ell$ is also called an $\ell$-isogeny. For every $\ell$-isogeny $\phi : E_1 \to E_2$ from $E_1$ to $E_2$, a so-called dual isogeny $\hat{\phi} : E_2 \to E_1$ of the same degree $\ell$ exists.

Figure 3 shows a non-trivial example of a 2-isogeny: between the two elliptic curves $y^2 = x^3 + 8x$, displayed on the left, and $y^2 = x^3 - 2x$, displayed on the right, both defined over $\mathbb{R}$, an isogeny of degree 2 exists. The point $R = (1, 3)$ on the left curve gets projected to the point $\phi(R) = (9/4, -21/8)$ on the right curve*. The kernel of the isogeny is ker$(\phi) = \{\mathcal{O}, P\}$, that is, the point at infinity and the point with the coordinates $(0, 0)$, the locus of the curve. The dual isogeny $\hat{\phi}$ has the kernel ker$(\hat{\phi}) = \{\mathcal{O}, Q\}$.

Note that since isogenies are non-constant maps, the size of their kernel is always guaranteed to be finite. In fact, we can uniquely identify an isogeny based on its kernel alone. Not only that, but thanks to a paper by French mathematician Jean Vélu released in 1971 [34], we can use Vélu’s formulas to evaluate an isogeny only by knowing its kernel. Storing isogenies is usually a memory-bound task that makes the application of isogenies in algorithms infeasible — Vélu’s formulas offer us a way around this by only having to store the comparatively smaller kernel of an isogeny.

Definition 2.11 (Vélu’s formulas). Let $E : y^2 = x^3 + ax + b$ be an elliptic curve in Weierstrass form defined over a finite field $K$. Let $G \subset E(\bar{K})$ be a subgroup of $E$ finite in size. For a point $P$, let $x(P)$ and $y(P)$ denote the $x$-coordinate and the $y$-coordinate of $P$, respectively. For any point $P \in E$, the isogeny $\phi$ with the kernel $G$, written as $\phi : E \to E/G$, can be evaluated as follows:

*The calculation is done by using SageMath [31]
The last thing left to explain is the meaning of supersingular. When we use the term supersingular isogeny, what we really mean is an isogeny between supersingular elliptic curves.

**Definition 2.12 (Supersingular elliptic curve).** Let $E$ be an elliptic curve defined over the field $\mathbb{F}_q$, where $q = p^2$ for some prime $p$. $E$ is called supersingular if its endomorphism ring $\text{End}(E)$ over the algebraic closure $\overline{\mathbb{F}_q}$ is non-commutative, otherwise it is called ordinary.

In particular, if $E$ is supersingular, then its endomorphism ring $\text{End}(E)$ is isomorphic to a maximal order in the quaternion algebra ramified at $p$ and $\infty$. If $E$ is ordinary, $\text{End}(E)$ is either isomorphic to a quadratic imaginary field or to $\mathbb{Z}$ iff $p = 0$, in both of which cases $\text{End}(E)$ ends up being commutative. Since no other cases exist, it is sufficient for us to define supersingular elliptic curves by the commutativity of their endomorphism rings.

There are many other equivalent ways of defining supersingular elliptic curves, some of which are easier to work with in certain contexts than others. For the purposes of this survey, it should suffice to understand that elliptic curves can be divided into the supersingular and the ordinary case without having to worry about the specifics.

Note that, due to the different structure of their endomorphism rings, a supersingular elliptic curve can only be isogenous to another supersingular elliptic curve.

### 2.5 Expander graphs

While most popular cryptographic schemes are not known for their usage of graph theory, the latter plays a very crucial role in the security of supersingular isogeny-based cryptosystems. As such, we shall begin with a short review of some graph theoretical jargon.

A graph $G = (V, E)$ consists of a set of nodes or vertices, usually denoted by $V$, and a set of edges, usually denoted by $E$. An edge is a tuple $(v, w) \in V \times V$ that consists of two nodes $v, w \in V$. In this case, the graph is called a directed graph or digraph and an edge $e = (v, w)$ is not the same as an edge $(w, v)$, that is, the edge $e$ connects $v$ to $w$ but not $w$ to $v$. In an undirected graph, on the other hand, an edge is a set $\{v, w\}$ connecting two nodes $v, w \in V$. If $e = \{v, w\} \in E$, then $e$ connects both $v$ to $w$ and $w$ to $v$. In the following, only undirected graphs are of interest to us.
The nodes \( v \) and \( w \) are called neighbors if an edge \([v, w] \in E\) exists. The degree of a node \( v \) is the number of neighbors it has, that is, \([|\{(v, w) \in E : w \in V\}|]\. A path \( v_1 \to \cdots \to v_n \) is a sequence of \( n \) nodes in \( V \) that is connected by edges in \( E \). Two nodes \( v, w \) are called connected if a path \( v \to \cdots \to w \) between them exists. The length of the shortest such path is called the distance \( \text{dist}(v, w) \) between \( v \) and \( w \). A graph \( G \) is called connected if for every pair of nodes \( v, w \in G \), a path \( v \to \cdots \to w \) exists, otherwise it is called disconnected. A set of connected nodes is called a connected component of \( G \). If \( G \) is a connected graph, it only has one connected component. The diameter of a connected component is the longest distance of all distances between any node \( v, w \in V \).

A connected graph \( G \) is called highly connected if, in order to separate it into at least two connected components, it is necessary to remove a large number of edges from \( E \). Conversely, if \( G \) can be separated by only removing a small number of edges from \( E \), then it is called weakly connected.

If \( G \) only has very few edges, it is called a sparse graph. If, on the other hand, its number of edges is close to the maximal number of possible edges (close to being complete graph described below), \( G \) is called a dense graph.

A graph \( G \) is called \( k \)-regular if every node \( v \in V \) has degree \( k \). \( G \) is called complete if for every two nodes \( v, w \in V \), an edge \([v, w] \in E\) exists, that is, if every node is connected to every other node.

A graph \( G \) can be represented using its adjacency matrix, a square matrix of size \( n \times n \), where \( n = |V| \). Assuming that the nodes in \( V \) are labeled by ascending numbers \( 1, \ldots, n \), the entry in the \( v \)-th row and \( w \)-th column of the adjacency matrix is 1 if and only if an edge \([v, w] \) exists in \( E \), otherwise it is 0. If the graph \( G \) is undirected, its adjacency matrix is symmetric and has \( n \) real eigenvalues \( \lambda_1, \ldots, \lambda_n \).

By examining the eigenvalues of the adjacency matrix of an undirected, \( k \)-regular graph, we can make a statement about both its denseness and connectedness.

**Definition 2.13 (Expander graph).** Let \( G = (V, E) \) be an undirected, \( k \)-regular graph, for \( k > 0 \). Let \( \lambda_1 \geq \cdots \geq \lambda_n \) be the eigenvalues of its adjacency matrix, sorted in ascending order. Let \( \varepsilon > 0 \). \( G \) is called a one-sided \( \varepsilon \)-expander graph if

\[
\lambda_2 \leq (1 - \varepsilon)k
\]

and a two-sided \( \varepsilon \)-expander if additionally

\[
\lambda_n \geq - (1 - \varepsilon)k.
\]

The formal definition of expander graphs is not very intuitive and does little to help us understand the practical applications of a graph with expander property. Specifically, the requirement for \( \lambda_2 \) instead of \( \lambda_1 \) in the first part of definition 2.13 is no mistake — in fact, \( \lambda_1 = k \). Intuitively, an expander graph is basically a graph that is both relatively sparse while still proving to be relatively strongly connected. In other words, an expander graph has only a small number of edges that are distributed among the nodes in such a way that it is difficult to divide the entire graph into two separated connected components by only removing a small number of edges.

Consider Fig. 4, which shows three different graphs. The circular graph on the left is sparse and weakly connected — it is possible to separate it by only removing two random edges. The complete graph on the right side, however, has the maximum amount of edges possible. It is very dense and highly connected — to separate it, one has to remove between 10 and 37 edges. The graph in the center is a so-called Peterson graph and fulfills the requirements to be considered an expander graph. It is very sparse while still being relatively highly connected, requiring between three and seven edges to be removed in order to separate it into two connected components.

**Definition 2.14.** Let \( G = (V, E) \) be an undirected, \( k \)-regular graph, for \( k > 0 \). Let \( \lambda_1 \geq \cdots \geq \lambda_n \) be the eigenvalues of its adjacency matrix, sorted in ascending order. Let \( \varepsilon > 0 \). \( G \) is called Ramanujan if it satisfies

\[
|\lambda_i| \leq 2\sqrt{k - 1},
\]

for all \( \lambda_i \) except \( \lambda_1 \) and \( \lambda_n \).

---

Fig. 4. A sparse, weakly connected circular graph (left), a sparse, highly-connected Peterson graph (center) and a dense, highly-connected complete graph (right).
Ramanujan graphs are a special case of expander graphs in that they are considered optimal expanders. Their fantastic stability properties make them very attractive for use in cryptographic contexts. The Peterson graph in Fig. 4 (center), for example, is not only an expander graph, it is also Ramanujan.

What makes Ramanujan graphs of interest for us is that they have very good mixing properties. Starting on a fixed node \( v \in V \), it only takes a relatively short random walk on the graph before the probability of stopping on a random node \( w \in V \) is the same for all nodes in \( V \), that is, we quickly approach a uniform distribution. This makes expander graphs in general and Ramanujan graphs in particular a good source of pseudo-randomness, a property that is very important in almost all cryptographic schemes.

### 2.6 Isogeny graphs

One interesting thing we can do with isogenies is viewing them as edges in a graph. The nodes of the graph then take the form of the isomorphism classes of the elliptic curves the isogenies map between, in other words, every node can be labeled with a corresponding \( j \)-invariant. The resulting graph is then called an isogeny graph.

The shape of an isogeny graph depends on two things: the field \( K \) that the elliptic curves are defined over and the degree \( \ell \) of the isogenies between them. An isogeny graph with isogenies of degree \( \ell \) representing the edges is also called an \( \ell \)-isogeny graph.

In general, an isogeny graph is not connected — instead, it consists of many small connected components, each consisting of isomorphism classes of isogenous elliptic curves. These components can be further divided: since supersingular elliptic curves can only be isogenous to other supersingular elliptic curves, it follows that no component in the isogeny graph can contain isomorphism classes of both ordinary and supersingular elliptic curves at the same time. Thus, we can divide the graph into ordinary and supersingular components — we obtain the ordinary isogeny graph and the supersingular isogeny graph.

In Fig. 5 we can see three components of the ordinary 3-isogeny graph defined over the finite field \( \mathbb{F}_{109^2} \). Since for every isogeny \( \phi : E_1 \to E_2 \) a corresponding dual isogeny \( \phi^* : E_2 \to E_1 \) exists, isogeny graphs are usually modelled as undirected graphs. It is easy to see that the components of the ordinary part are sparse and weakly connected. Due to their form, these components are also called isogeny volcanoes [13] — most of them start with a central “crater” from which all other nodes branch out until they reach the “foot” of the volcano. Note that volcano is not a term unique to isogeny graphs but a technical term that is used in other fields of graph theory as well.

Much more interesting to us are the components containing supersingular elliptic curves. As it turns out, they have a very different structure compared to their ordinary counterparts. Not only that, it is also possible for us to uniquely identify the supersingular component: every isogeny graph contains exactly one such component. Due to its uniqueness, we simply refer to it as the supersingular isogeny graph.

In Fig. 6, we can see the supersingular 2-isogeny graph (left) and the supersingular 3-isogeny graph (right) defined over the same field \( \mathbb{F}_{109^2} \). Compared to the ordinary case, it is immediately obvious that in the supersingular case, the resulting graph is much more highly connected. In particular, the 2-isogeny graph on the left is 3-regular, while the 3-isogeny graph on the right is 4-regular. As it turns out, every supersingular \( \ell \)-isogeny graph is (almost) \( \ell + 1 \)-regular.

In fact, supersingular \( \ell \)-isogeny graphs are not only (almost) \( \ell + 1 \)-regular, they are (almost) Ramanujan, giving them optimal expansion properties and making them a fantastic source of pseudo-randomness. Indeed, it is this structure that supersingular isogeny-based cryptography is based upon.

Note that both graphs have the same amount of nodes. This is due to the fact that, if defined over a field \( \mathbb{F}_p^\ast \) with \( p \) prime, approximately \( p/12 \) supersingular isomorphism classes exist.

Another thing one might note is the existence of edges from a node to itself, so-called loops, as well as two distinct edges between the same two nodes in the 3-isogeny graph on the right. These are not restricted to the supersingular case and might occur in some components of the ordinary isogeny graph, too. Indeed, isogeny graphs are multigraphs.

---

Fig. 5. Three different components of the ordinary 3-isogeny graph defined over the field \( \mathbb{F}_{109^2} \).
2.7 Random walks

As we have seen in the last section, we can use supersingular isogenies of degree \( \ell \) in order to create \( \ell + 1 \)-regular multi-graphs with optimal expansion properties. These graphs have fantastic mixing properties, meaning that any two nodes \( v \) and \( w \) in the graph are connected by a relatively short path and that a random walk starting at a random node \( v \) quickly approaches uniform distribution in regards to the last node \( w \) in the path. Indeed, all supersingular isogeny-based cryptographic schemes are based on the idea of random walks in supersingular isogeny graphs.

Consider Fig. 7, which shows random walks of length 5 in the supersingular isogeny graphs of degree 2 on the left side and degree 3 on the right side, both defined over the field \( \mathbb{F}_{1092} \). Every edge taken in the graph represents an isogeny of degree 2 in the left graph (respectively 3 in the right graph). Since we are dealing with separable isogenies only, the size of the kernel \( \ker(\phi) \) of each isogeny \( \phi \) is the same its degree, here, 2 and 3. A walk of length two in the left graph can be seen as chaining two isogenies \( \phi \) and \( \psi \), both of degree 2: \( \phi \circ \psi : E \to E', P \mapsto \psi(\phi(P)) \). The resulting isogeny is one of degree \( 2^2 = 4 \).

Indeed, we can continue this for walks of arbitrary length \( e \) in order to end up with an isogeny of degree \( 2^e \). In our example, the walk of length 5 in the supersingular 2-isogeny graph on the left is equal to a walk of length one in the supersingular isogeny graph of degree \( 2^5 = 32 \) over the same field \( \mathbb{F}_{1092} \). Similarly, the walk of length 5 in the supersingular 3-isogeny graph on the right is equal to a walk of length one in the supersingular isogeny graph of degree \( 3^5 = 243 \) over the same field.

Since the resulting graph would be a 32-regular (respectively 244-regular) multi-graph with only nine nodes, we will refrain from trying to create a plot.

Even though, in our example, the resulting isogeny is of degree 32 (respectively 243), to construct such isogeny by chaining smaller isogenies of degree 2 (respectively 3), it suffices to know the kernel of each small-degree isogeny in the path one would take inside the supersingular isogeny graph. In our example, with a random walk of length 5, that would be 2 (respectively 3) points for each edge, so \( 2 \times 5 = 10 \) in total (respectively \( 3 \times 5 = 15 \)). Since we know that each kernel must include the neutral element, the point at infinity \( \mathcal{O} \), we can reduce this number even further: the knowledge of \( 5 \times (2 - 1) = 5 \) points is enough for us to construct an isogeny with kernel size 32 in the left graph (respectively \( 5 \times (3 - 1) = 10 \) for an isogeny of degree 243 in the right graph).

Chaining of small-degree isogenies gives us the ability to easily compose large-degree isogenies in a space-efficient manner. Explicitly constructing isogenies of large-degree is actually a memory-bound task and plays an important role in the security of the following cryptographic schemes.
3. Supersingular Isogeny Diffie–Hellman

While there were previous attempts to design quantum-secure cryptosystems based on isogenies due to Rostovtsev and Stolbunov [27], the first practical scheme was a key exchange protocol based on supersingular isogenies due to Jao and De Feo [22]. Not only was it several orders of magnitudes faster, taking less than a second for a key exchange compared to several minutes in the previous approach; being based on isogenies between supersingular elliptic curves as opposed to ordinary curves like was the case in the Rostovtsev–Stolbunov protocol, it also improved on the fact that cryptosystems based on ordinary isogenies have been found to be vulnerable to sub-exponential quantum attacks [6], therefore ruling them out as potential candidates for secure communication in a post-quantum world.

The Jao–De Feo protocol is what we will be discussing in this section of our survey. While introducing several new ideas that have not appeared in any previous cryptosystems before, its basic structure is reminiscent of a well-established cryptographic protocol used almost everywhere: the Diffie–Hellman key exchange protocol. This is also where the Jao–De Feo scheme takes its other name from: the SIDH.

In Sect. 3.1, we will quickly go over the original Diffie–Hellman protocol. In Sect. 3.2, we will describe the SIDH due to De Feo and Jao, and explain novel concepts which the protocol introduced in order to work around the quantum-vulnerability of the original Diffie–Hellman. In Sect. 3.3, we will introduce the required security assumptions that are needed in order for the protocol to be considered secure. Finally, in Sect. 3.4, we will show potential attack vectors that need to be taken into consideration when discussing SIDH as a realistic candidate for post-quantum key exchange.

3.1 Ordinary Diffie–Hellman

The Diffie–Hellman key exchange protocol is named after its inventors Whitfield Diffie and Martin Hellman who published their idea for the first time in 1976. It was one of the first practical public-key cryptosystems and is still of great importance even today.

Diffie–Hellman solves the problem of two parties, hereby called Alice and Bob, trying to establish a secure connection over an insecure communication channel. If Alice and Bob both had knowledge of a common passphrase, they could encrypt their communication using a symmetric cipher. Since this is not the case, the only thing they can do is to use the insecure channel to negotiate a shared secret in such a way that an eavesdropper, hereby called Eve, can not derive the secret value on her own just by observing the messages sent by Alice and Bob. In other words, Diffie–Hellman is a key exchange protocol.

Let Alice and Bob agree on a finite cyclic group $G$ of prime order $p$ with generator $g$, i.e., $G = \langle g \rangle$. This group is public knowledge and visible to everyone. Alice and Bob then both choose secret elements $1 \leq a, b < p$ that they will not share with anyone else. Now,

- Alice computes $g^a$ and sends it to Bob,
- Bob computes $g^b$ and sends it to Alice,
- Both derive the shared secret $s = (g^a)^b = (g^b)^a$.

Our eavesdropper, Eve, only has knowledge of $G, g, (g^a)$ and $(g^b)$ in this scenario. She knows neither $a$ nor $b$. The general idea behind the key exchange is shown in Fig. 8.

The security of the Diffie–Hellman protocol strongly depends on the representation of the group $G$. The original implementation defined $G$ to be the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^\times$ of integers modulo $p$, where $p$ is a prime. Eve then would have to solve the following problem.

Problem 3.1 (Diffie–Hellman problem). Given a cyclic group $G$ and its generator $g$, as well as the values $g^a$ and $g^b$, derive the value of $g^{ab}$.

This problem is the so-called DHP. In the case of the group $(\mathbb{Z}/p\mathbb{Z})^\times$, this problem is considered infeasible to solve on traditional computers. However, as discussed in Sect. 1, this is no longer the case for quantum computers—the DHP can be broken if an easy solution for the so-called discrete logarithm problem (DLP) is known. Shor’s algorithm [28] provides exactly that in the context of a quantum computer: given an integer $N$, it finds all its prime factors. Thus,
Alice's random walk does not backtrack. In other words, Alice choosing a non-backtracking random walk of length $e_A/C^{30}$:

Supersingular Isogeny-based Cryptography: A Survey

In her supersingular exponentiation by graph depends on two things: the field both take random walks in two different supersingular isogeny graphs defined over the same set of nodes, i.e., the same algorithm, it also means that achieving the commutative structure that is required for asymmetric protocols to work is a algorithm.

Since we are dealing with separable isogenies, the sizes of the kernels of their isogenies $\ker$ isogenies of large degree $\text{an}$ however, let Alice and Bob use isogenies of different degrees $\text{an}$ $\ker$ is chosen. $E(F_p)$ and $G$ make up the public parameters of the scheme. The secret parameters that Alice and Bob choose are positive integers $a, b$ that are smaller than the order of $G$. Instead of exponentiation by $n$, Alice and Bob use the multiply-by-$n$ map to arrive at the secret value $S = [a][b]G = [b][a]G$.

The ECDH protocol has its own version of the DHP, called the elliptic curve Diffie–Hellman problem. Unfortunately, due to the cyclic nature of the groups defined by elliptic curves, ECDH is just as vulnerable to Shor’s algorithm as the normal Diffie–Hellman.

3.2 Protocol

This section explains the SIDH key exchange protocol due to Jao and De Feo. This scheme was first proposed in [22]. Later, with [11], an extended version of the paper due to De Feo, Jao and Plüts was released. The latter paper due to the cyclic nature of the groups defined by elliptic curves, ECDH is just as vulnerable to Shor’s algorithm as the normal Diffie–Hellman.

As it turns out, there is no group action at all on the structure of supersingular isogeny graphs. While this means that the scheme’s security is not based on a version of the discrete logarithm problem and thus not vulnerable to Shor’s algorithm, it also means that achieving the commutative structure that is required for asymmetric protocols to work is a bit tricky.

The SIDH protocol solves this problem by taking a novel approach: our participants in the protocol, Alice and Bob, both take random walks in two different supersingular isogeny graphs defined over the same set of nodes, i.e., the same isomorphism classes of supersingular elliptic curves. As we have seen before, the shape of a supersingular isogeny graph depends on two things: the field $F_q$ it is defined over and the degree $\ell$ of the isogenies that are used to represent the edges in the graph.

Since we want Alice’s and Bob’s graph to consist of the same nodes, we cannot change the field $F_q$. We can, however, let Alice and Bob use isogenies of different degrees $\ell_A$ and $\ell_B$. That way, Alice’s random walk takes place in an $\ell_A + 1$-regular graph, while Bob’s random walk takes place in an $\ell_B + 1$-regular graph. Since we can efficiently create large isogenies by chaining small isogenies, it suffices to select small numbers for both $\ell_A$ and $\ell_B$ — indeed, in practice, the protocol defines $\ell_A$ to be 2 and $\ell_B$ to be 3.

After taking their random walks of length $e_A$ and $e_B$ through their graphs, Alice and Bob essentially end up with isogenies of large degree $\ell_A$ and $\ell_B$ defined over $F_q$. We call these the secret isogenies $\phi$ and $\psi$ of Alice and Bob. Since we are dealing with separable isogenies, the sizes of the kernels of their isogenies $\ker(\phi)$ and $\ker(\psi)$ will be the same as their degrees, $\ell_A^\phi$ and $\ell_B^\psi$.

In fact, as is always the case for kernels, the points in $\ker(\phi)$ and $\ker(\psi)$ define a group structure on their own: if $\phi : E \to E_A$, then $\ker(\phi) \subset E$, i.e., the kernel of $\phi$ is a subset of its domain curve $E$. This kernel is cyclic if and only if Alice’s random walk does not backtrack. In other words, Alice choosing a non-backtracking random walk of length $e_A$ in her supersingular $\ell_A$-isogeny graph is the same as her choosing a subgroup $\langle A \rangle \subset E$ of $E$, or more specifically, a

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Public parameters</strong></td>
<td></td>
</tr>
<tr>
<td>A finite field $\mathbb{F}_p$</td>
<td></td>
</tr>
<tr>
<td>An elliptic curve $E(\mathbb{F}_p)$</td>
<td></td>
</tr>
<tr>
<td>A generator $G$ of $E(\mathbb{F}_p)$</td>
<td></td>
</tr>
<tr>
<td><strong>Secret parameters</strong></td>
<td></td>
</tr>
<tr>
<td>$1 \leq a &lt; \text{ord}(G)$</td>
<td>$1 \leq b &lt; \text{ord}(G)$</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Exchange</strong></td>
<td></td>
</tr>
<tr>
<td>$A \rightarrow$</td>
<td>$\leftarrow B$</td>
</tr>
<tr>
<td><strong>Shared secret</strong></td>
<td></td>
</tr>
<tr>
<td>$S = [a]B$</td>
<td>$S = [b]A$</td>
</tr>
</tbody>
</table>

Fig. 9. A sketch of the Elliptic Curve Diffie–Hellman key exchange protocol.
Alice needs access to one. The starting elliptic curve $E \in \mathcal{E}_A$. See the reference [10, Sect. 14.2] for the details. The same is of course true for Bob and his secret isogeny $\psi$: by choosing a random walk of length $e_B$ in the supersingular $\ell_B$-isogeny graph, he essentially chooses a secret cyclic subgroup $(B) \subset \mathcal{E}_B$ of the $\ell_B$-torsion subgroup in $E$.

Because $(A)$ and $(B)$ are well-defined subgroups with generating sets $A$ and $B$, we can construct a new subgroup $(A,B)$ of $E$ with generating set $(A,B)$. Since we know that $(A)$ and $(B)$ are cyclic and that $\ell_A \neq \ell_B$, the resulting subgroup is also cyclic of order $\ell_A \ell_B$. If we use this subgroup together with Vélu’s formulas, we receive a new separable isogeny $\rho : E \to E_{AB}$ of degree $\ell_A \ell_B$. As one might suspect, it is possible to construct the $\ell_A \ell_B$-isogeny $\rho$ by composing smaller degree isogenies. Until now, we have only used isogenies of the same degree to compose larger isogenies of degree $\ell'$ (e.g., by taking random walks through an isogeny graph). However, we do not have to restrict ourselves like that: it is easily possible to compose isogenies of different degrees $\ell$ and $\ell'$ in order to receive a larger isogeny of degree $\ell \ell'$.

This is the entire idea behind the commutativity aspect of the SIDH protocol.

- Alice and Bob both decide on random walks starting from a preset node $j(E)$ in their supersingular isogeny graphs in order to receive their isogenies $\phi : E \to E_A$ and $\psi : E \to E_B$, respectively.
- Alice sends $E_A$ to Bob and Bob sends $E_B$ to Alice.
- Alice and Bob repeat their random walks, this time starting from nodes $j(E_A)$ and $j(E_B)$ respectively, in order to receive the isogenies $\phi' : E_B \to E_{AB}$ and $\psi' : E_A \to E_{AB}$.

Alice and Bob then choose $j(E_{AB})$ as their shared secret.

A sketch of the scheme can be seen in Fig. 10. Here, $E$ is the elliptic curve that corresponds to the starting node in the supersingular isogeny graphs. We recall that every node in the supersingular isogeny graphs is labeled with a corresponding $j$-invariant. $\phi$ and $\psi$ denote the secret isogenies of degree $\ell_A$ and $\ell_B$ that Alice and Bob receive after completing their random walks in their corresponding graphs. While not directly used in the scheme, $\rho$ denotes that a direct isogeny with kernel $(A,B)$ from $E$ to $E_{AB}$ exists.

One obvious problem that remains is that even though the kernels $(A)$ and $(B)$ of the secret isogenies $\phi$ and $\psi$ consist of points on the elliptic curve $E$, they might not be valid points on the curves $E_B$ or $E_A$, let alone define a valid subgroup that can be used as a kernel. Since isogenies are homomorphisms and as such structure-preserving, this problem can be solved by projecting the points of the generating sets $A$ and $B$ into the domain curves $E_B$ and $E_A$.

This, however, results in another problem. In order to project the generating set $B$ of his isogeny $\psi$ into the domain $E_A$, Bob needs access to Alice’s secret isogeny $\phi$, otherwise he cannot compute $\phi(B)$ (compare Fig. 10). Conversely, Alice needs access to $\psi$ in order to compute $\psi(A)$. Revealing their secret isogenies to the other party so they can do the computation themselves is obviously no option, since this would defeat the entire point of the protocol. Conversely, revealing their secret subgroup and asking the other party to do the computation is also out of the question, since this is akin to revealing the secret isogeny itself.

SIDH solves this problem by fixing parts of the generating sets $A$ and $B$ as part of the public parameters, next to $E$. In fact, the elliptic curve $E$ that is used as the starting point of the protocol is constructed in such a way as to make the selection of $A$ and $B$ easy: as we have seen before, the degrees of Alice’s and Bob’s isogeny graphs are set to small primes $\ell_A$ and $\ell_B$, in most cases 2 and 3. Since the security of the protocol depends on the degree of their final secret isogenies $\phi$ and $\psi$, the length of their random walks $e_A$ and $e_B$ are chosen so that $\ell_A^e_A \approx \ell_B^e_B$. Based on these two numbers, a prime $p = \ell_A^e_A \ell_B^e_B f + 1$ for some small $f$ is selected. In practice, many such primes exist and it is easy to find one. The starting elliptic curve $E$ is then selected such that it is supersingular over the field $\mathbb{F}_p$ and has order $(p + 1)^2$.

The resulting elliptic curve $E$ then has cyclic $\ell_A^e_A$- and $\ell_B^e_B$-torsion subgroups with generating sets $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ respectively, called the bases of the torsion groups. With this, we have identified all the parts necessary to make up the public parameters

$$(p, E(\mathbb{F}_p), P_A, Q_A, P_B, Q_B).$$

![Fig. 10. Commutativity diagram of the SIDH key exchange.](image-url)
Instead of letting Alice and Bob choose \( \langle A \rangle \subset E[\ell_A^\alpha] \) and \( \langle B \rangle \subset E[\ell_B^\alpha] \) directly, we let each of them choose two secret integers: \( 0 < m_A, n_A < \#E[\ell_A^\alpha] \) for Alice and \( 0 < m_B, n_B < \#E[\ell_B^\alpha] \) for Bob. Using the bases fixed in the public parameters, they can now compute their secret subgroups like this:

\[
\langle A \rangle = \langle [m_A] P_A + [n_A] Q_A \rangle \subset E[\ell_A^\alpha],
\]

\[
\langle B \rangle = \langle [m_B] P_B + [n_B] Q_B \rangle \subset E[\ell_B^\alpha].
\]

The problem of Alice and Bob having to reveal their secret isogenies is now solved due to the fact that isogenies are only isogenous if and only if they have the same \( j \)-invariant, which is how Alice and Bob establish their shared secret in the final step.

A sketch of the entire scheme (without computation of \( \phi \) and \( \psi \) using random walks) can be seen in Fig. 11. It shows the public parameters that may be fixed before the execution of the protocol, selection of the secret parameters for both parties and the preparation, exchange and final steps of the protocol. The shared secret that both Alice and Bob use for their further communication is the \( j \)-invariant of the final curve \( E/(A,B) \in \mathbb{F}_{p^2} \), which lies in the algebraic closure of the field \( \mathbb{F}_p \) the starting curve \( E \) is defined over.

One thing that might be worth noting is that \( \phi \) and \( \psi \) are merely structure-preserving, so while Alice and Bob end up in the same isomorphism class, the elliptic curve \( E/(A,B) \) that Alice obtains might not be the same curve that Bob obtains. However, since both curves are guaranteed to be isomorphic, both of them will end up having the same \( j \)-invariant, which is how Alice and Bob establish their shared secret in the final step.

### 3.3 Security assumptions

While understanding the idea behind the SIDH protocol and the structure of it is possible having only partial knowledge of the structure of supersingular elliptic curves, assessing its security is not possible without having a solid grasp on all of the details that the scheme entails. As such, we will restrict ourselves to stating the required assumptions for the scheme to be considered secure against passive attacks. For formal security proofs based on these assumptions we refer the reader to [11].

For a compact summary of computational problems in the case of supersingular isogenies, we refer to [18]. The paper summarizes several problems that are presumed to be hard to solve both in the traditional and the quantum case. It also touches upon known algorithms and approaches to solve said problems. This section is partially based on this paper.

Let us begin with the most basic problem that all other ones are based upon.

**Problem 3.2** (General isogeny problem). Given the \( j \)-invariants \( j, j' \in \mathbb{F}_q \), find an isogeny \( \phi : E \to E' \), so that \( j(E) = j \) and \( j(E') = j' \).

Deciding if such an isogeny exists, i.e., deciding if \( E \) and \( E' \) are isogenous, is an easy task, since elliptic curves are only isogenous if and only if they have the same number of points. The difficulty with finding the isogeny itself is that storing and representing isogenies is a memory-bound task: the kernel size of \( \phi \) grows exponentially with the size of the degree \( \ell \) of \( \phi \). This problem can also be adapted to one of finding a path from \( j(E) \) to \( j(E') \) in one of the corresponding isogeny graphs. However, creating such a graph, is, again, a memory-bound problem.

**Problem 3.3** (\( \ell \)-isogeny problem). Given the \( j \)-invariants \( j, j' \in \mathbb{F}_q \) and a positive integer \( \ell \), find an isogeny \( \phi : E \to E' \) of degree \( \ell \), so that \( j(E) = j \) and \( j(E') = j' \).
This more specific version of the general isogeny problem requires the solution $\phi$ to be an isogeny of a specific degree $\ell$. It is unclear whether this version makes the problem harder or easier to solve than the original: on one hand, it could make the problem easier since it drastically reduces the search space by assuring that an isogeny of degree $\ell$ exists. On the other hand, it could make the problem harder since it restricts the possible solutions to only a few (in most cases one) isogenies.

**Problem 3.4 (SIDH isogeny problem).** Let $E$ be a supersingular elliptic curve and let $P_A, Q_A, P_B, Q_B$ be the auxiliary points as defined in the SIDH protocol. Let $E_A$ be another supersingular elliptic curve such that an isogeny $\phi: E \to E_A$ of degree $\ell_A^A$ exists. Additionally, let $P_B = \phi(P_B)$ and $Q_B = \phi(Q_B)$.

Given $(E, P_A, Q_A, P_B, Q_B, E_A, P_B', Q_B')$, find an isogeny $\phi': E \to E_A$ of degree $\ell_A'$ such that $\phi'(P_B) = P_B'$ and $\phi'(Q_B) = Q_B'$.

This very specific problem could also be formulated as follows: after observing an SIDH handshake between Alice and Bob (as defined in the previous section), try to reconstruct Alice’s secret isogeny $/C30$.

It is unclear how this will influence passive attacks on the scheme and, similarly to the $\ell$-isogeny problem, if the restrictions an adversary would have to work with will make this problem harder or easier to solve.

Similarly to how the Decisional Diffie–Hellman problem follows from the normal Diffie–Hellman problem, it is possible to formulate different variants of the SIDH isogeny problem for different contexts.

**Problem 3.5 (Decisional SIDH isogeny problem).** Let $(E, P_A, Q_A, P_B, Q_B)$ be as in the SIDH isogeny problem and let $E_A$ be an elliptic curve. Let $P_B', Q_B' \in E_A[n]$ and let $n$ be a positive integer less than $\ell_A$.

Given $(E, P_A, Q_A, P_B, Q_B, E_A, P_B', Q_B', n)$, determine whether an isogeny $\phi: E \to E_A$ of degree $\ell_A^A$ exists, such that $P_B = \phi(P_B')$ and $Q_B = \phi(Q_B)$.

If an efficient solution for the Decisional SIDH isogeny problem exists, then the normal SIDH problem can also be solved efficiently [18, Sect. 6.2].

One last problem that is worth mentioning is the following.

**Problem 3.6** Let $E$ and $E'$ be two isogenous supersingular elliptic curves. Given $E, E'$ and the endomorphism ring $\text{End}(E)$, compute $\text{End}(E')$.

It is widely assumed that computing the endomorphism rings of two supersingular elliptic curves $E$ and $E'$ is equivalent to computing an isogeny $\phi: E \to E'$. Since the elliptic curve $E$ is a part of the SIDH public parameters is fixed, we can assume its endomorphism ring $\text{End}(E)$ is already known. Hence, it would suffice for an adversary to be able to compute $\text{End}(E')$ of an isogenous curve $E'$ in order to break the SIDH protocol.

In order to understand why these two problems are equivalent, it is necessary to understand the significance of quaternion algebras and their maximal orders, something which we have not explained in the preliminaries, and as such is out of the scope of this paper. We instead refer the interested reader to Sect. 4 of [16], which gives an explicit algorithm for the computation of isogenies between two supersingular elliptic curves $E$ and $E'$ under the assumption that the endomorphism rings $\text{End}(E)$ and $\text{End}(E')$ are known.

Figure 12 gives an overview of the relations between the mentioned problems in this section.

### 3.4 Known attacks

Due to the unusual structure of the SIDH protocol that involves revealing auxiliary points, as well as its approach to efficiently evaluating large-degree isogenies by composing small-degree isogenies in what is essentially a random walk...

---

**Fig. 12.** Relationship between the different problems [18]; $A \to B$ means that solving the problem $A$ implies solving $B$. $AB$ means that solving the problem $A$ implies solving $B$ under some restriction.
on a supersingular isogeny graph, many novel attacks have been proposed, one of which even manages to recover the entire secret key of a party in the case of static secrets. In this section we are going to quickly list the most relevant attacks. Since most other supersingular isogeny-based cryptosystems can be seen as variations of the SIDH scheme, most of these attacks apply to other supersingular isogeny schemes, too.

We are starting with the most significant attack by Galbraith, Petit, Shani and Bo Ti [16], in which they assume the position of a dishonest Bob who is manipulating the key exchange in such a way that he can recover Alice’s entire static secret isogeny. Static in this context means that Alice does not change her secret parameters after every (successful or failed) key exchange. In the following we are explaining the first step of the attack.

Let \((E, P_A, Q_A, P_B, Q_B)\) be the public parameters of an SIDH key exchange. Let \((m_A, n_A)\) be Alice’s static secret key, such that \([m_A]P_A + [n_A]Q_A\) defines the kernel of her secret isogeny \(\phi\). Bob creates his own ephemeral secret \((m_B, n_B)\) and obtains Alice’s public key \((E_A, \phi(P_B), \phi(Q_B))\) before continuing as follows.

- Bob completes his side of the protocol to obtain the shared secret \(j(E_{AB})\).
- Instead of \((E_B, \psi(P_A), \psi(Q_A))\), he sends \((E_B, \psi(P_A), \psi(Q_A) + [e_A^{\ell_A-1}]\psi(P_A))\) to Alice.
- If the protocol succeeds and Alice obtains the same value \(j(E_{AB})\), Bob knows that \(n_A\) is even, otherwise it is odd.

The last step follows from the fact that if both arrive at the correct value for \(j(E_{AB})\), then

\[
([m_A]\psi(P_A) + [n_A]\psi(Q_A)) = ([m_A]\psi(P_A) + [n_A]\psi(Q_A) + [e_A^{\ell_A-1}]\psi(P_A)).
\]

Since \(P_A\) and \(Q_A\) are points in the \(\ell_A^n\)-torsion subgroup \(E[\ell_A^n]\), they have order \(\ell_A^n\) by definition. If \(n_A\) is even, then \(n_A\ell_A^n\) is a multiple of \(e_A^n\), hence \([n_A][e_A^n]\psi(Q_A) = \emptyset\) and the equation follows.

Conversely, if both groups are equal, then some factor \(\lambda\) exists, such that

\[
\lambda([m_A]\psi(P_A) + [n_A]\psi(Q_A) + [e_A^{\ell_A-1}]\psi(P_A))) = [m_A]\psi(P_A) + [n_A]\psi(Q_A).
\]

Because \(P_A\) and \(Q_A\) form the generator of a torsion group, they are independent and so \(\lambda n_A = n_A\), which implies \(\lambda = 1\), and so we are back in the case where \(n_A\ell_A^n\) is a multiple of \(\ell_A^n\), which implies that \(n_A\) is even. Hence, the last bit of Alice’s private key is 0.

The follow-up steps work in a similar fashion and retrieve one bit of Alice’s private key at a time, requiring less than \(\log_2 \lambda\) (successful or otherwise) key exchanges. For the full attack, we refer to the paper itself [16].

What the attack by Galbraith et al. demonstrates is that it is possible to exploit the unique structure of SIDH that involves computing auxiliary points, without breaking the security assumptions made by the protocol about the difficulty of computing isogenies between two given supersingular elliptic curves. Note that the attack by Galbraith et al. only breaks SIDH in the case of static keys, i.e., when Alice reuses her secret parameters instead of generating a new pair for every handshake. In the latter case of ephemeral keys, the protocol is still unbroken.

The paper suggests using a countermeasure that has been proposed by Kirkwood et al. [24], which involves Bob not choosing the secret values by himself, but instead using a key derivation function that is part of Alice’s static public key. This allows Alice to later confirm that Bob honestly completed his part of the protocol by repeating his computations on her side — however, this drastically impacts the performance of the scheme, one of the supposed main advantages of supersingular isogeny-based cryptography.

Other attacks against supersingular isogeny-based systems make use of the fact that computing the secret isogenies of large degree is done by composing many small-degree isogenies in what is essentially a random walk on a small-degree supersingular isogeny graph. This has spawned interest in fault attacks, a type of side-channel attack against static key cryptosystems that relies on disrupting the computation of one parties’ side of the protocol in order to trick them into revealing partial information of their secrets. We will quickly describe two of these attacks here.

The first such attack is due to Bo Ti [32]. His attack model assumes the adversary’s ability to induce a fault in Alice’s computation of \(\phi(P_B)\), tricking her into evaluating her secret isogeny \(\phi\) on a random point \(X\) instead, thus publishing \((E_A, \phi(X), \phi(Q_B))\) instead of \((E_A, \phi(P_A), \phi(Q_A))\) after completing her part of the protocol. In particular, the paper shows that it is possible to recover Alice’s secret isogeny \(\phi\) if we have knowledge of the image \(\phi(X)\) for at least one point \(X \in E[\ell_A^n]\) in the \(\ell_A^n\)-torsion group of \(E\). A proposed countermeasure is to check the order of the points \(\phi(P_B)\) and \(\phi(Q_B)\) before sending them to Bob.

Another fault attack on supersingular isogeny cryptosystems was proposed by Gélin and Wesolowski in [19]. Their attack is based on the notion of loop-aborts, which, in the case of SIDH, means to trick Alice into prematurely stopping her secret walk, resulting in computing her secret isogeny only partially. Using this approach, they reconstruct Alice’s secret isogeny by tricking her into iteratively revealing the low-degree isogenies she uses during her walk in the supersingular isogeny graph.

The last paper we are going to mention is by Christophe Petit [26]. While all other attacks up to this point involved an active adversary, Petit provides new algorithms for unbalanced variants of SIDH. This shows that the hard problems SIDH and other supersingular isogeny protocols build upon might be easy to solve in certain situations, when the public parameters slightly deviate from what is required by the protocol.
4. Supersingular Isogeny-based Public Key Encryption

In this section we are going to describe the public-key encryption (PKE) scheme that was introduced by De Feo, Jao and Plütt in [11], the extended version of the paper that initially introduced the SIDH protocol [22]. Its structure is based on the SIDH protocol and is reminiscent of how the ElGamal encryption scheme follows from the ordinary Diffie–Hellman protocol.

As a short reminder, ElGamal, like Diffie–Hellman, is defined over a cyclic group $G$ of prime order $p$ with a generator $g$, such that $(g) = G$. Alice chooses a random integer $x < p$ and computes $g^x$. She also chooses a random hash function $h \in \mathcal{H}$. Her public key is $(G, g, (g^y), h)$, and her private key is $x$. To encrypt a plaintext $m$, Bob chooses a random $y < p$, computes the shared secret $(g^y)^x$ and creates the ciphertext $((g^y)^x, c)$, where $c = m \oplus h((g^y)^x)$. Alice then decrypts $c$ by computing $m = c \oplus h((g^y)^x)$. Note that the binary operator $\oplus$ means the bitwise exclusive OR in this section and the next section.

Since a large part of the protocol in this section is identical to the steps in the SIDH protocol, we will omit any details that have already been explained in Sect. 3.2 and instead only give a sketch of the PKE in Sect. 4.1. In Sect. 4.2 we are going to analyze its security under the security assumptions that have been made in Sect. 3.3 and show that the protocol is IND-CPA but not IND-CCA, much like the original ElGamal. Finally, in Sect. 4.3, we are going to discuss why it is difficult to obtain an efficient CCA-secure PKE scheme by analyzing the difficulties that arise when trying to translate a non-quantum-secure IND-CCA scheme into the supersingular isogeny case based on the example of the Cramer–Shoup cryptosystem [9], which was based on the IND-CPA ElGamal cryptosystem.

4.1 Protocol

This section explains the public-key encryption scheme based on supersingular isogenies due to De Feo, Jao and Plütt [11]. It basically follows from the SIDH in the same way that the ElGamal public-key encryption scheme follows from the ordinary Diffie–Hellman. Its structure is essentially similar to that of the SIDH protocol.

Let $E$ be a supersingular elliptic curve defined over the field $\mathbb{F}_p$, for a prime $p = e^{\ell_p} \ell_p f + 1$. Let $(P_A, Q_A)$ be a basis of the $\ell_p$-torsion group $E[\ell_p]$ and $(P_B, Q_B)$ a basis of the $\ell_p$-torsion group $E[\ell_p]$. This setup is identical to the setup of the SIDH protocol. Additionally, let $\mathcal{H}$ be a hash function family that maps elements from $\mathbb{F}_{p^f}$ to the message space $[0, 1]^w$, i.e., to bit strings of length $w$. We note about the treatment of hash functions in analyses of protocols. On the security of protocols dealt in this paper, the random oracle model [2] is employed as the security model of hash functions. In this model, hash functions are considered to output truly random strings, i.e., ideally random functions. Many cryptographic protocols use the random oracle model to focus on algorithms by eliminating the effects of applied hash functions.

**Key Generation.** Alice chooses two random integers $0 < m_A, n_A < \ell_A^{\ell_p}$, such that $m_A$ and $n_A$ are not divisible by $\ell_A$. Alice completes her side of the SIDH protocol to obtain $E_A, \phi(P_B)$ and $\phi(Q_B)$. Additionally, she chooses a random hash function $h \in \mathcal{H}$. Alice’s private key is $(m_A, n_A, h)$, and her public key is $(E_A, \phi(P_B), \phi(Q_B), h)$.

**Encryption.** Bob retrieves Alice’s public key and chooses two random integers $0 < m_B, n_B < \ell_B^{\ell_p}$, such that $m_B$ and $n_B$ are not divisible by $\ell_B$. He completes his part of the SIDH protocol to obtain the shared secret $j(E_{AB})$. For plaintext $m \in [0, 1]^w$, Bob computes $c = m \oplus h(j(E_{AB}))$, that is, he uses the hash function $h$ and the shared secret $j(E_{AB})$ to create a one-time pad.

Bob’s ciphertext is $(E_B, \psi(P_A), \psi(Q_A), c)$.

**Decryption.** Alice obtains Bob’s ciphertext, completes the final part of the SIDH protocol to obtain $j(E_{AB})$ and decrypts the obtains the original plaintext by computing $m = c \oplus h(j(E_{AB}))$.

4.2 Security

The way that the public-key encryption scheme described in this section derived from SIDH is essentially equivalent to how the ElGamal public-key encryption scheme is derived from the ordinary Diffie–Hellman protocol. ElGamal encryption is well-known to provide IND-CPA security.

**Theorem 4.1** ([21]). Under the assumption that the SIDH isogeny problem is hard, the supersingular isogeny public-key encryption scheme due to De Feo, Jao and Plütt provides IND-CPA security in the random oracle model.

While ElGamal proves to be IND-CPA-secure, it is unconditionally malleable, which makes it easily vulnerable to adaptive chosen ciphertext attacks. Unfortunately, the same is true for the public-key encryption scheme due to De Feo, Jao and Plütt.

Let $O$ be the oracle in the IND-CCA attack model as described in Sect. 2.1. Eve can ask $O$ for decryptions of arbitrary ciphertexts. At one point, Eve asks $O$ to encrypt one of two plaintexts $m_0, m_1$ to receive the ciphertext $c$, after which she can continue asking for decryptions of arbitrary ciphertexts not equal to $c$. Eve then has to decide if $c$ is an encryption of $m_0$ or $m_1$.

- Eve skips the first phase of the game and generates two random plaintexts $m_0, m_1 \in \{0, 1\}^w$. 


- Eve initiates the second phase of the game by handing \( m_0 \) and \( m_1 \) to \( O \).
- \( O \) randomly chooses \( b \in [0, 1] \), computes the encryption \( c \) of \( m_b \) and hands the ciphertext \((E_B, \psi(P_A), \psi(Q_A), c)\) to Eve.
- Eve asks \( O \) to decrypt the ciphertext \((E_B, \psi(P_A), \psi(Q_A), \tau)\), where \( \tau \) denotes the complement of \( c \), in order to receive the plaintext \( m' \).
- Eve answers

\[
b' = \begin{cases} 0 & \text{if } m' = \overline{m_0} \text{ and } \\ 1 & \text{otherwise.} \end{cases}
\]

In this game, Eve will always guess right. Assuming that \( j(E_{AB}) \) is the shared secret in the encryption scheme, an encryption of \( m_0 \) results in the ciphertext \((E_B, \psi(P_A), \psi(Q_A), c)\), where \( c = m_0 \oplus h(j(E_{AB})) \). The decryption of \((E_B, \psi(P_A), \psi(Q_A), \tau)\) results in the plaintext

\[
m' = \tau \oplus h(j(E_{AB})) = \overline{m_0} \oplus h(j(E_{AB})) \oplus h(j(E_{AB})) \oplus h(j(E_{AB})) = \overline{m_0}.
\]

### 4.3 Adapting other PKE schemes

Many attempts have been made to build upon the original ElGamal scheme in order to improve its security. In the past, the adaptive chosen ciphertext attack model has been seen as overly permissive and thus unrealistic. The fact that the early cryptosystems that provably provided IND-CCA security were either computationally expensive or only provided partial proofs added to the fact that IND-CCA security was seen as sufficient. Later, when the first practical real-world CCA-attacks showed up, this assumption proved to be wrong.

The first cryptosystem that was both proven to be secure against adaptive chosen ciphertexts under standard intractability assumptions and highly efficient was the Cramer–Shoup cryptosystem introduced in [9]. At its core, the Cramer–Shoup is a modified version of ElGamal, which adds the property of non-malleability to the protocol. It has been shown by Bellare and Namprempt that non-malleability and security against adaptive chosen ciphertext attacks are equivalent [1].

Let \( G \) be a cyclic group of prime order \( q \) and let \( \mathcal{H} \) be a family of one-way hash functions that map from bit strings to \( \mathbb{Z}_q \). A quick description of the Cramer–Shoup cryptosystem is as follows:

**Key Generation.** Alice chooses random elements \( g_1, g_2 \in G \) and \( x_1, x_2, y_1, y_2, z \in \mathbb{Z}_q \) as well as a random hash function \( H \in \mathcal{H} \).

Alice’s private key is \((x_1, x_2, y_1, y_2, z)\), and her public key is \((g_1, g_2, c, d, h, H)\).

**Encryption.** For plaintext \( m \in G \), Bob chooses a random \( r \in \mathbb{Z}_q \) and computes \( u_1 = g_1^r \), \( u_2 = g_2^r \), \( e = h^r m \) and \( v = c d^m \), where \( \alpha = H(u_1, u_2, e) \).

Bob’s ciphertext is \((u_1, u_2, e, v)\).

**Decryption.** Alice computes \( \alpha = H(u_1, u_2, e) \) and confirms that \( u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha} = v \), before recovering the plaintext \( m = e/\overline{u_1^\alpha} \).

In this scheme, the actual encryption and decryption of the plaintext \( m \) is completely analogous to the (non-hashed variant) of ElGamal. Its resilience against adaptive chosen ciphertext attacks is due to the fact that it is not possible to modify \( c \) without having to create a new tag \( v \). Changing \( v \), however, requires knowledge of either Alice’s or Bob’s secret parameters. The use of the hash value \( \alpha \) ensures that the tag \( v \) loses its otherwise homomorphic property. The scheme’s efficiency comes from the fact that the creation of \( v \) only depends on the hash function \( H \) and operations within the group \( G \), both of which can be implemented efficiently. For a formal proof of the scheme’s IND-CCA properties we refer to [9].

Seeing how it was easily possible to translate the ElGamal scheme into the supersingular isogeny case, one might assume that adapting the Cramer–Shoup scheme to work with supersingular isogenies is within the realm of possibilities. However, as we have seen in Sects. 3.3 and 3.4, the SIDH protocol quickly becomes insecure if only a partial leak of information occurs. This greatly reduces our freedom to extend the public-key cryptosystem due to De Feo, Jao and Plût.

A direct translation of the scheme would involve Alice choosing a random point \( R \in E \) as well as a one-way hash function \( H \in \mathcal{H} \), both of which she would add to her public key. Bob would then use \( H \) on the ciphertext \( C \) in order to receive the hash \( \alpha = H(C) \). Both parties could then combine \( \alpha \) with their secret isogenies, e.g., by computing \( \phi([\alpha] \psi(R)) \) and \( \psi'(1[\alpha] \phi(R)) \) (compare Fig. 10) to obtain a point \( R' \in E_{AB} \), the \( j \)-invariant of which would correspond to \( v \) in the original Cramer–Shoup scheme. A sketch of this can be seen in Fig. 13.

This approach fails for several reasons: unlike ElGamal and Cramer–Shoup, supersingular isogeny-based cryptosystems are not based on any group action. We recall that isogenies are simply a special case of homomorphisms between elliptic curves. As such, any isogeny computations with known kernels can be easily manipulated, since
5. Supersingular Isogeny Key Encapsulation

In this section we will introduce the SIKE specification that was submitted to NIST’s competition [25] to find a set of post-quantum cryptographic schemes, which would then be standardized.

SIKE consists of two parts: SIKE-PKE, an IND-CPA-secure public-key encryption scheme, and SIKE-KEM, an IND-CCA-secure key encapsulation mechanism protocol. SIKE-PKE is actually the ElGamal-like cryptosystem due to De Feo, Jao and Plût that we have introduced in Sect. 4. Hence, in this Sect. 5.1, we will only explain the key encapsulation mechanism part of the specification, SIKE-KEM, before looking at the resulting changes in regards to security and performance in Sect. 5.2.

5.1 Protocol

This section describes the key encapsulation mechanism SIKE-KEM. At its core, SIKE-KEM is an extended version of the IND-CPA-secure public-key encryption scheme due to De Feo, Jao and Plût that employs additional safeguards in order to achieve resilience against adaptive chosen ciphertext attacks and address some of the other attack vectors that have been discovered in the context of static public keys. In particular, it applies a transformation by Hofheinz, H"ovelmanns and Kiltz [20] to the PKE described in Sect. 4. This transformation is similar to the one proposed by Kirkwood et al. [24] that was suggested as a safeguard against the full-key recovery attack by Galbraith, Petit, Shani and Bo Ti. Both transformations are based on the model by Fujisaki–Okamoto [14].

Since we have already discussed the PKE that SIKE-KEM is based on in Sect. 4, we will only explain the differences introduced by the addition of the transformation by Hofheinz et al. For the sake of understanding, we will use the same parameter names and sizes that we have used throughout Sects. 3 and 4, even though the formal specification [21] uses slightly adjusted parameters in regards to key space and so on.

Let $E$ be a supersingular elliptic curve defined over the field $\mathbb{F}_p$ for a prime $p = 2^{256} - 3^{128} - 1$. Let $\{P_A, Q_A\}$ be a basis of the $3^s$-torsion group $E[3^s]$ and $\{P_B, Q_B\}$ be a basis of the $2^{25}t$-torsion group $E[2^{25}t]$. Let $F$, $G$ and $H$ be three hash functions.

**Key Generation.** Alice sets $m_A = 1$ and chooses a random integer $0 < n_A < 3^s$, not divisible by 3. Using $(m_A, n_A)$, she completes her side of the SIDH protocol to obtain $E_A, \phi(P_B)$ and $\phi(Q_B)$. Alice’s private key is $n_A$, and her public key is $p_{k_A} = (E_A, \phi(P_B), \phi(Q_B))$.

**Encapsulation.** Bob retrieves Alice’s public key and sets $m_B = 1$. He chooses a random $m \in \{0, 1\}^w$ and uses the hash function $G$ to compute $n_B = G(m || p_{k_A})$. Using $(m_B, n_B)$, he completes his part of the SIDH protocol to obtain $p_{k_B} = (E_B, \psi(P_A), \psi(Q_A))$ and the shared secret $j(E_{AB})$.

\[
\psi'([\alpha]\phi(R)) = [\alpha]\psi'(\phi(R)).
\]

For a modified ciphertext $C'$, it is thus easily possible to create a valid tag by simply computing \[H(C')[[\alpha^{-1}] \psi'([\alpha]\phi(R))].\]

Another, far more dangerous problem this approach runs into is the fact that, as we have seen in Sect. 3.4 and has been demonstrated in [32], the evaluation and the publication of random points $X$ on $E$ using the secret isogenies $\phi$ and $\psi$ are dangerous, as it allows an adversary an easy way to reconstruct the entire secret isogeny.

Another approach, which avoids the problem of the homomorphic properties of isogenies, would be to apply the hash $\alpha$ to the kernels of their secret isogenies. This would, however, break the protocol in the way that it would require more communications between Alice and Bob, since they would essentially be performing another SIDH handshake with slightly modified kernels, based on the ciphertext created during the first handshake.

The takeaway is that due to the unique properties of supersingular isogeny-based cryptosystems, which first-and-foremost includes the lack of a proper group structure, it is not easily possible to translate some existing cryptographic schemes into the supersingular isogeny case. As of writing, no truly efficient IND-CCA-secure public-key encryption scheme based purely on supersingular isogenies has been proposed. De Feo has also noted in [10] that generalizing efficient digital signatures like the elliptic curve-based ECDSA into the supersingular isogeny case also poses a problem due to the lack of group law on the public data.

\[
\phi(R) \rightarrow \phi'([\alpha]\psi(R))
\]

\[
\psi(R) \rightarrow \psi'([\alpha]\phi(R))
\]

Fig. 13. A direct translation of of the verification step into the supersingular isogeny case.
Bob uses the hash function $F$ to compute the encrypted text $c = m \oplus F(j(E_{AB}))$. He then uses the hash function $H$ to create the tag $K = H(m \parallel \|pk_B\|c)$. Bob’s ciphertext is $(E_B, \psi(P_A), \psi(Q_A), c)$.

**Decapsulation.** Alice obtains Bob’s ciphertext and completes the final part of the SIDH protocol to obtain the shared secret $j(E_{AB})$. She decrypts $c$ by computing $m' = c \oplus F(j(E_{AB}))$. Using the hash function $G$, Alice computes $n'_B = G(m' \parallel \|pk_A\|)$, sets $m''_B = 1$. She uses $(m''_B, n''_B)$ to recompute Bob’s part of the protocol. She obtains $pk'_B$. Alice confirms that $pk'_B = pk_B$. If true, she uses the hash function $H$ to compute $K = H(m' \parallel \|pk_B\|c)$, if not, she aborts the protocol without further comment as to why.

After completing the protocol, Alice and Bob can use the value of $K$ to secure any further communication, e.g., by using it as a key for a symmetric block cipher.

Notice that in SIKE-KEM, while Alice’s public key can be static, Bob has no choice when it comes to the selection of his own secret isogeny. The security of the protocol depends on the fact that Bob’s computations are completely transparent to Alice if Bob has completed his part of the protocol in an honest manner. The result is that SIKE-KEM only works in the context of ephemeral–ephemeral or static–ephemeral keys, but not in the context of static–static keys.

We highly stress that while our description of the protocol correctly portrays the idea behind SIKE-KEM, for the sake of ease of understanding, it heavily simplifies the official specification [21] that has been submitted to NIST. Among others, for reasons of efficiency, the official specification uses supersingular elliptic curves in Montgomery form instead of short Weierstrass form. This allows the evaluation of isogenies only storing their $x$ coordinate. Also for performance reasons, the points $P_A, Q_A$ (as well as $P_B, Q_B$) are encoded using three $x$-coordinates, $x_{P_A}, x_{Q_A}$ and $x_{R_A}$, where $x_{R_A}$ is the $x$-coordinate of the point $R_A = P_A - Q_A$. It goes without saying that SIKE-KEM does not compute direct isogenies from $E$ to $E_A$ and $E_B$ but instead uses an optimized strategy that includes the chaining of small-degree isogenies as we have seen in Sect. 2.

For anyone interested in the full specification, we refer to the document submitted to NIST [21].

### 5.2 Security and performance

SIKE-KEM is based on the public-key encryption scheme due to De Feo, Jao and Plüt, which is also referred to as SIKE-PKE in the SIKE specification. Since the latter provably provides IND-CPA-security, in order to provide the promised resilience against adaptive chosen ciphertext attacks, it is sufficient for the introduced transformations to make the ciphertext non-malleable. As Bellare and Namprempre [1] have shown, non-malleability and IND-CPA security are equivalent to IND-CCA security.

**Theorem 5.1 ([21]).** Assuming that the SIDH isogeny problem is hard, SIKE-KEM is IND-CCA secure in the random oracle model.

Even though SIKE-KEM manages to provide resilience against adaptive chosen ciphertext attacks, the overhead introduced by applying the transformations due to Hofheinz et al. [20] is significant. The result is that, while still being reasonable fast enough to be used in practical applications, SIKE-KEM ends up being slower than other popular post-quantum algorithms. However, supersingular isogeny-based systems still produce the smallest public keys out of all quantum-secure public-key schemes, making the choice of SIKE-KEM a tradeoff between high performance with large key sizes and acceptable performance with small key sizes.

### 6. Conclusion

In this survey we have a look at the most important post-quantum cryptographic schemes that have emerged within the young field of supersingular isogeny-based cryptography. After explaining the minimal amount of mathematical background knowledge required to get an intuitive understanding of supersingular isogeny-based schemes, we introduce the first and most defining protocol due to Jao and De Feo, the SIDH [22]. We follow up by describing an IND-CPA-secure public key encryption scheme due to De Feo, Jao and Plüt [11], which is based on the previously discussed SIDH protocol. We explain that even though the construction of the resulting PKE is essentially identical to how the well-known ElGamal cryptosystem follows from the original Diffie–Hellman, translating an IND-CCA-secure scheme like the ElGamal-based Cramer–Shoup cryptosystem into the supersingular isogeny case proves to be a difficult task that severely limits the applicability of supersingular isogenies in certain areas. In the end, we introduce the SIKE protocol that extends the previously-introduced public-key encryption scheme in order to address its security-concerns in the case of static keys and to add IND-CCA security in the random oracle model, while at the same time losing its advantage as one of the faster protocols for post-quantum cryptography.

Supersingular isogeny-based cryptosystems build on top of the already well-established area of elliptic curve cryptography, making it a fairly easy to understand topic for security researchers already invested in this branch of cryptography. At the same time, the protocols derived from supersingular isogenies introduce many edge cases and, due to their unique approach to the commutativity aspect required of asymmetric protocols, introduce even more complexity on top of the already complex topic of regular elliptic curve cryptography. Its structure, which is uniquely
defined by the lack of any group operations on the commutative structure of isogeny graphs, additionally makes it difficult to adapt many already existing protocols into the supersingular isogeny case. Right now, it is still an open question if it is possible at all to create efficient IND-CCA secure public-key encryption based on supersingular isogenies. Moreover, proving the security of supersingular isogeny-based cryptosystems in the quantum random oracle model [4] is also an important open question since supersingular isogeny-based cryptosystems are expected as quantum-resistant cryptosystems. While SIKE is a highly promising and very practical key encapsulation mechanism, it is apparent that supersingular isogenies are no silver bullet against the threat of quantum computers. Nonetheless, supersingular isogenies are a valuable tool, and there has been a lot of progress in both increasing the efficiency and decreasing the key sizes of supersingular isogeny cryptosystems, which indicates that schemes like SIDH have yet to reach their full potential. One can only hope that research interest in supersingular isogeny-based cryptography will continue to increase.

Acknowledgments

We would like to thank two anonymous reviewers for their valuable comments and suggestions which help us to improve the manuscript. We are also grateful to Kristin E. Lauter for her fruitful suggestions on the history of the supersingular isogeny-based cryptography.

REFERENCES

Supersingular Isogeny-based Cryptography: A Survey


