ELLIPTIC-CURVE CRYPTOGRAPHY

TRISHA SABADRA

Abstract. For decades, public-key cryptography has been vital for secure online communication. As computing power grows and more advanced algorithms are developed, the widely used Diffie-Hellman and RSA algorithms require larger keys for security. Elliptic curve cryptography (ECC) offers a more efficient alternative to RSA, achieving equivalent security with smaller keys and reduced computing power. This paper explores how elliptic curves are used in modern cryptography, current limitations, and potential future developments. Addressing the threat of quantum computers to public-key systems, we discuss a recent attack to supersingular elliptic curve isogenies, prompting further research into quantum-resistant cryptographic solutions.

1. Introduction to Cryptography

Before the 1970s, for two parties to securely communicate without a third party intercepting the message, they had to physically meet and agree on a shared secret. This kind of cryptography is known as *symmetric*, because a single key is used for both encryption and decryption. While effective in the early 20th century, the rise of computer network communication and frequent online transactions spurred the need for two parties to quickly and securely communicate without ever actually meeting.

1.1. **Diffie-Hellman Key Exchange.**

In 1976, Diffie, Hellman, and Merkle invented one of the first *asymmetric* or public key cryptosystems, allowing two parties to establish a shared secret key without prior communication. In public key encryption, the encryption and decryption are performed using a combination of public (known to everyone) and private (secret) keys. The mathematics behind the Diffie-Hellman Key Exchange is based on the discrete logarithm problem.

Definition 1. *We denote the* ***discrete logarithm problem*** *in the finite field of integers modulo prime p as follows: given $g, k \in \mathbb{Z}/p\mathbb{Z}$, find an integer a such that:*

$$g^a \equiv k \mod p$$

Let's do a quick run through of this: Let's say Alice and Bob want to communicate securely without Eve *eaves*dropping.

(1) Alice and Bob publicly agree on a prime p and an integer g
(2) Alice randomly chooses a private integer a and Bob chooses b, both between 1 and p.
(3) Alice calculates $g^a \mod p$, Bob calculates $g^b \mod p$, and they publicly exchange the result.
(4) Both sides have the shared key k such that

$$k = (g^b)^a \mod p = (g^a)^b \mod p = g^{ab} \mod p$$

The security of Diffie-Hellman Key Exchange rests on the assumption that while Alice and Bob can quickly compute the shared key, the third party Eve will be unable to get the shared key without solving the discrete logarithm problem - which is believed to be exponentially hard. Thus, the discrete logarithm problem is an example of a trapdoor function, where one direction is easy to compute, while the inverse is hard.

Example. Calculating $3^{29} \mod 17$, is easy, but finding a given $3^a \mod 17 \equiv 12$ is harder. Because a computer can quickly brute force this example to find $a = 29$, the recommended order of the prime modulus is 2048 bits - which could take thousands of years to run through all possibilities!

1.2. **RSA.**

In 1977, Ron Rivest, Adi Shamir, and Leonard Adleman proposed another public key encryption scheme, known as RSA. The security of RSA is based on a similar problem to Diffie-Hellman, the discrete factoring problem.

Definition 2. *We denote the* ***discrete factoring problem*** *as follows: given an integer N, find primes p, q such that $pq = N$.*

RSA was the first widely used public key algorithm and is still used in many modern protocols. However, the general number field sieve can solve the factoring problem in sub-exponential time. This algorithm, coupled with the rise in computational power, raises scalability concerns for RSA, necessitating the growth of key sizes to maintain security (like DH, the recommended key size is a minimum of 2048 bits). Given that these systems might be present on low-computing power devices like mobile phones or chips, many are concerned that longer keys can negatively impact the performance of application because of the processing power required for encryption and decryption.

2. Elliptic Curves

After the introduction of RSA and Diffie-Hellman, researchers explored other mathematics-based solutions for cryptography - looking for algorithms that would serve as good trapdoor functions and achieve an equivalent level of security with less computing power. In 1985, Neal Koblitz and Victor S. Miller independently suggested the use of elliptic curves in cryptography.

Definition 3. *An* ***elliptic curve*** *is the set of solutions (x, y) of*

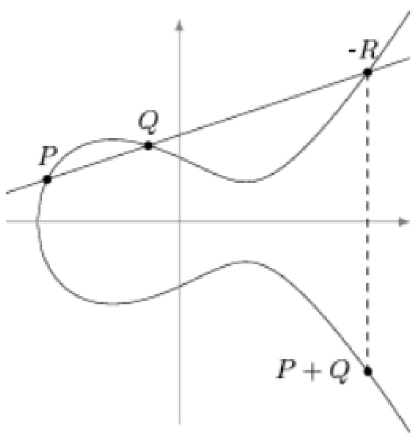
$$y^2 = x^3 + ax + b$$

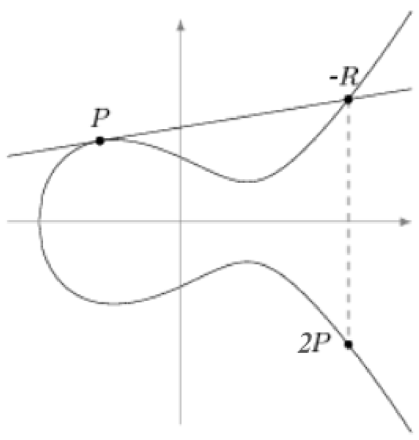
for some numbers a, b, together with the point at ∞. We require that the discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$, meaning that $x^3 + ax + b$ has no repeated roots.

2.1. **Group Law.** What makes elliptic curves so important to cryptography is that the rational points on the curve form an *abelian group*, which allows us to add together points on the curve.

To define the group structure, we make the identity of the group the "point at infinity." This sounds quite abstract - but we can think of it as a point that all vertical lines go through.

Note that for each point $P = (x, y)$ on the curve, the point $P' = (x, -y)$ must also be on the curve - so we can define P and P' as inverses in the group.





Now, what does it actually mean to add two points on the elliptic curve? We can look at this geometrically: take two points P and Q on an elliptic curve E and draw a line connecting them. This line intersects the curve at a third point, call it $-R$. Reflecting this point across the x-axis (i.e taking the inverse) we get the point $R = P + Q$.

Note that this construction is not possible if $Q = -P$ because the line through P and Q is vertical, so $-P + Q = \infty$. If $P = Q$, i.e we want to double a point, we take the tangent to the curve at P and use the second intersection with the curve as the point $-R$.

2.2. **Elliptic Curves over Finite Fields.** For elliptic curve cryptography, we are interested in elliptic curves over finite fields, which is very similar to the definition of elliptic curves over the reals.

Definition 4. *For prime p, let $F_p = \mathbb{Z}/p\mathbb{Z}$ be the finite field of integers from 0 to $p - 1$. The elliptic curve E over F_p is the set of solutions of*

$$y^2 \equiv x^3 + ax + b \mod p$$

for $x, y \in F_p$.

Like the previous definition of elliptic curves, the cubic polynomial has no repeated roots in F_p, together with the point at ∞. This means $-16(4a^3 + 27b^2) \mod p \neq 0$.

The graph of elliptic curves over a finite field doesn't looks like a curve in the traditional sense, as it is just a discrete set of points. You can think of it as the original curve "wrapped" around at the edges, but only the whole number coordinates have points on the graph. Like $E(\mathbb{Q})$, $E(F_p)$ forms a group, so we can carry over the group law for adding points.

Definition 5. *Using this operation, we can define* ***scalar multiplication*** *on the group by repeatedly adding a point to itself:*

$$nP = P + P + \cdots + P$$

n times

Definition 6. *The* ***order*** *of a point P is the smallest positive integer n such that $nP = \infty$, where ∞ is the identity of the group.*

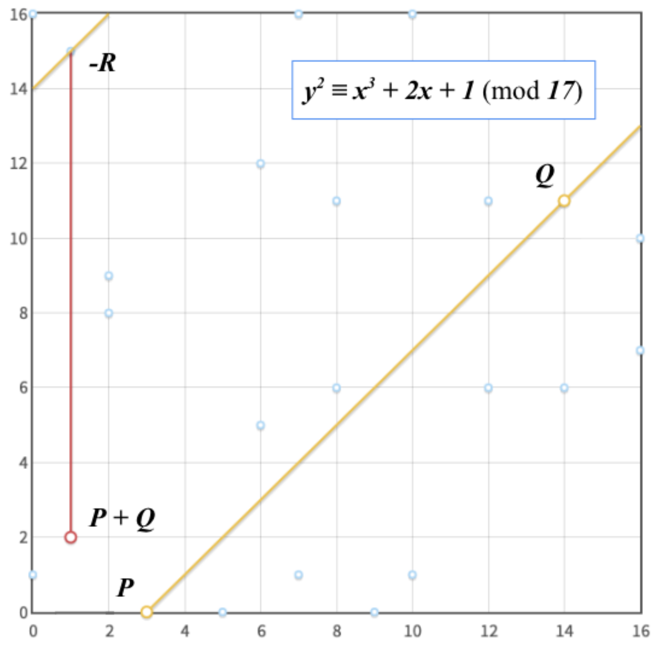


FIGURE 1. Given points $P = (3, 0)$ and $Q = (14, 11)$ on the elliptic curve above, we draw a line between P and Q and "wrap" it around the graph modulo 17. Taking the inverse, we get the point $R = P + Q = (1, 2)$

Example. Let E be the elliptic curve $y^2 = x^3 + 2x + 1$ in F_{17} (shown in the image above). Say we start with the point $P = (2, 8)$. Then we can find the points *generated* by P

$$2P = (5, 0)$$

$$3P = (2, 9)$$

$$4P = \infty$$

$$5P = (2, 8)$$

$$\dots$$

This generates the cyclic *subgroup*: $\{\infty, (2, 8), (5, 0), (2, 9)\}$. Since $4P = \infty$, the order of P is 4.

We can compute nP efficiently for large n using the *double-and-add* method.

```
(1) initialize the resulting point $Q = 0$
(2) for each bit in the binary representation of $n$:
    (a) double $Q$
    (b) if the current bit is 1, add $P$ to $Q$.
```

Since we are only iterating over $\log_2(n)$ bits, this method can compute nP in $O(\log(n))$ time, which is exponentially better than the the naive linear algorithm of simply adding P to itself n times.

3. Elliptic Curve Cryptography (ECC)

The biggest advantage of ECC over RSA and Diffie-Hellman is efficiency - ECC allows the use of smaller keys while maintaining the same level of security as RSA, resulting in faster encryption and less memory. Arjen K. Lenstra visualized this in a unique way, comparing how much water the energy that is needed to break a cryptographic algorithm could boil - a kind of cryptographic carbon footprint [LKT13]. By this measure, breaking a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water, while breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, you'd need a key with 2,380-bits.

The reason that ECC can use smaller keys is because it is computationally more challenging than breaking RSA - the fastest attacks on ECC run in exponential time, while RSA can be broken in sub-exponential time. Consequently, RSA is considered "easier" to break and is thus more susceptible to potential attacks.

After a slow start, elliptic curve cryptography is now used in a wide variety of applications: the U.S. government uses it to protect internal communications, it is the mechanism used to prove ownership of bitcoins, and it provides signatures in Apple's iMessage service. While RSA and Diffie-Hellman are still the norm, elliptic curve cryptography has become the leading alternative for privacy and security online.

3.1. **Elliptic Curve Diffie-Hellman (ECDH).**

Like the regular Diffie-Hellman, the ECDH is a key exchange protocol based on the discrete logarithm problem [Woh16]

Definition 7. *We denote the* ***elliptic curve discrete logarithm problem*** *(ECDLP) as follows: given the elliptic curve E on the finite field F_p and points P, Q on E, find the smallest integer n such that $nP = Q$.*

Lets run through the ECDH with Alice and Bob:

(1) They agree on the public prime p, elliptic curve E over the finite field F_p, random point P on E, and the order n of P (for the discrete logarithm problem to be difficult on elliptic curves, the order n needs to be large). This generates the cyclic subgroup:
$$\{\infty, P, 2P, 3P, \ldots, (n-1)P\}$$
(2) Alice and Bob each randomly choose a private key from 1 to $n-1$, a and b respectively.
(3) Alice calculates the public key $Q_A = aP$ and Bob calculates $Q_B = bP$ efficiently using the double and add algorithm.
(4) They exchange public keys and each compute the shared key $aQ_B = bQ_A = abP$. They can use the x or y coordinate of the point as the shared key.

Example. Let the elliptic curve E be $y^2 \equiv x^3 + 2x + 2 \mod 17$ and the generator point $P = (5, 1)$. Calculating the multiples of P, we get $19P = \infty$, so the order $n = 19$. Say Alice privately chooses $a = 3$ and Bob chooses $b = 9$. They calculate the points $Q_A = 3P = (10, 6)$ and $Q_B = 9P = (7, 6)$ and exchange them. Alice computes $3Q_B = (13, 7)$ and Bob computes $9Q_A = (13, 7)$. They get the same point because $3(9P) = 9(3P) = 27P = 8P = (13, 7)$.

Eve knows the starting point P and the ending points Q_A and Q_B, but the scalars a, b are private, so she will be unable to get the shared key without solving the discrete logarithm problem, for which no efficient algorithm is known. Thus, this is a trapdoor function because computing nP can be done efficiently, but the opposite direction, namely finding n given P and nP is hard.

3.2. **Elliptic Curve Integrated Encryption Scheme (ECIES).**

The shared key generated from ECDH can be used to set up a secured symmetric channel for communication. This is known as a *hybrid encryption scheme*, as it leverages the efficiency of symmetric-key encryption for encrypting the actual message and the convenience of public-key cryptography for securely agreeing on the shared key. Here is a high level description of how ECIES works (for more details see [Bro09])

(1) Alice and Bob follow the ECDH protocol to derive a shared secret S
(2) They compute a symmetric key k using a *key derivation function*: $k = \mathrm{KDF}(S)$
(3) Using an authenticated encryption scheme, Alice encrypts her message m with the key k to get the ciphertext $c = E(k; m)$
(4) Alice computes a *message authentication code* (MAC) d of her encrypted message
(5) Alice sends the c and d to Bob
(6) Bob uses the authentication tag d to ensure that the message is from Alice and hasn't been changed.
(7) Bob decrypts the message $m = E^{-1}(k; c)$

3.3. **Elliptic Curve Digital Signature Algorithm (ECDSA).**

Another advantage of elliptic curve cryptography is that Alice can "sign" a message, so that Bob can verify that the message *is* from Alice, and there was no third party interference. Here is an overview of the signature generation algorithm:

(1) Alice and Bob publicly agree on an elliptic curve E over F_p, and a point P on E.
(2) Alice takes a hash of her message m and truncates it so that it has bit-length n, where n is the order of the subgroup generated by P. Let z be the resulting truncated hash of m.
(3) Alice privately chooses an integer k between 1 and n and computes $Q = (x_1, y_1) = kP$.
(4) Let $r = x_1 \mod n$. If $r = 0$, Alice must go back to step 3 and choose another k.
(5) Alice calculates $s = k^{-1}(z + rd_A) \mod n$, where d_A is Alice's private key and k^{-1} is the multiplicative inverse of $k \mod n$. If $s = 0$, go back to step 3 and choose another k.
(6) The pair (r, s) forms the signature

Now to verify Alice's signature, Bob uses Alice's public key Q_A (the end point) and proceeds as follows:

(1) Bob computes $u_1 = zs^{-1} \mod z$
(2) Bob computes $u_2 = rs^{-1} \mod z$
(3) Bob computes the point $Q = (x_1, y_1) = u_1P + u_2Q_A$
(4) If $r \equiv x_1 \mod n$, then Bob knows the message came from Alice.

Proof. We start with Q as described above. $Q_A = aP$, where a is Alice's private key.

$$\begin{aligned} Q &= u_1 P + u_2 Q_A \\ &= u_1 P + u_2 a P \\ &= (u_1 + u_2 a) P \end{aligned}$$

Now let $s = k^{-1}(z + r d_A)$. Then,

$$\begin{aligned} Q &= (z s^{-1} + r s^{-1} a) P \\ &= s^{-1}(z + r a) P \\ &= kP \end{aligned}$$

So in both cases we found the point Q, which would have only been possible if the message was signed by Alice's private key. This completes the proof.

3.4. **Pollard's ρ Algorithm.**

Although there is currently no mathematical proof for the complexity of the discrete logarithm problem, the security of ECC relies on the claim that it is "hard" to solve. To get an idea of how "hard" it is, we'll look at Pollard's ρ Algorithm, currently the most efficient classical algorithm for computing discrete logarithms on elliptic curves [WIN].

Definition 8. *With Pollard's ρ, we first solve a slightly different problem: given P and Q, find integers a, b, A, B where $(a, b) \neq (A, B)$ such that*

$$aP + bQ = AP + BQ$$

If we just try all possible pairs (a, b) and (A, B) randomly, we have an $O(k^2)$ running time (or $O(2^{2m})$, where m is the bit length). However, there is a more efficient method: *Floyd's cycle finding algorithm*

(1) Choose a random sequence S of points and start the pointers $p_1 = (a, b), p_2 = (A, B)$ start at the beginning of S.
(2) In each iteration, p_1 moves one step, while p_2 moves two steps along the sequence.
(3) Stop when $aP + bQ = AP + BQ$, i.e p_1 and p_2 are at the same point within the cycle.

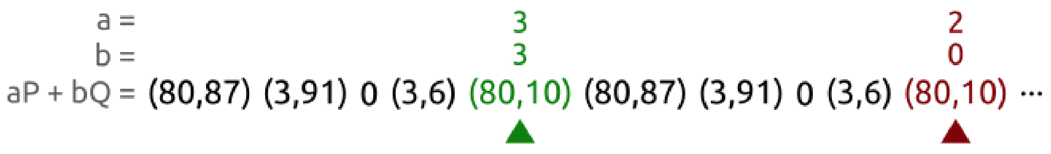


FIGURE 2. In this example, we have the curve $y^2 \equiv x^3 + 2x + 3 \mod 97$ and the points $P = (3, 6)$ and $Q = (80, 87)$, belonging to a cyclic subgroup of order 5. We walk the sequence of pairs at different speeds until we find the pairs $(3, 3)$ and $(2, 0)$ that produce the same point.

The reason why the algorithm must terminate is that the subgroup is finite and cyclic, so it will eventually find a pair that works.
Once a, b, A, B are found, we can substitute $xP = Q$ and solve for x :

$$aP + bQ = AP + BQ$$
$$aP + bxP = AP + BxP$$
$$(a - A)P = (B - b)xP$$

Now we can cancel out P, but since the subgroup is cyclic with order n, we solve it modulo n:

$$a - A \equiv (B - b)x \mod n$$
$$x = (a - A)(B - b)^{-1} \mod n$$

Thus, we found x such that $xP = Q$.

The hard part is analyzing the complexity of the algorithm, but a probabilistic proof can show a bound of roughly $O(\sqrt{k})$. The proof is based on the "birthday paradox," which is about the probability of two people having the same birthday, where here we are concerned about the probability of two (a, b) pairs yielding the same point. Although this time complexity is better than a brute force attack, it is still exponential time ($O(2^{m/2})$), so in practice, it takes months for Pollard's ρ to break a standard elliptic curve protocol. Thus, elliptic curves will likely remain at the center of public key cryptography for many years.

3.5. **Challenges.**

The choice of elliptic curve parameters and key sizes is crucial for ensuring the security of the digital signatures, but the domain parameters are expensive to generate, so the curves need to be pre-computed in advance. The US National Institute of Standards and Technology (NIST) provides guidelines for selecting secure elliptic curves, however, these standard curves are complicated to implement securely in practice. Yet, looming over these considerations is a far-reaching threat: quantum computing.

4. Post-Quantum ECC

A quantum computer works in a completely different way from a classical computer. Unlike classical bits, which are either 0 or 1, quantum bits or *qubits* represent a probability to be either 0 or 1. We say that a qubit can exist in a *superposition state* of both 0 and 1 simultaneously. The probabilities of measuring the qubit in states 0 or 1 are given by the squaring the amplitudes.

Example. Mathematically, we can represent a qubit's state as follows:

$$|x\rangle = 0.8|0\rangle + 0.6|1\rangle \longrightarrow |x\rangle = 64\%|0\rangle + 36\%|1\rangle$$

This means that a set of two qubits can be in a superposition of four states, which therefore require four numbers to uniquely identify the state. So the amount of information stored in n qubits is 2^n classical bits. Thus, quantum computers don't speed up individual operations, but the number of operations to get the result is exponentially smaller. However, the resulting state is limited, because although a qubit can exist in a superposition, when we measure the state of a qubit, we get a single random value.

4.1. **Shor's Algorithm.**

In 1994, Peter Shor showed that a quantum computer could factor large numbers and solve the discrete logarithm problem in polynomial time rather than exponential [Sho94]. The key to the polynomial running time of Shor's algorithm is a particular procedure on quantum computers call the Quantum Fourier Transform. We won't present the QFT in this paper, but essentially it allows us to find the period of a cyclic function. Here is a brief overview of Shor's Algorithm for the large prime factorization problem:

Suppose we want to find primes p, q such that $pq = n$ for some large n. We can first reduce the factoring problem to the problem of order-finding.

(1) Choose some $g < n$ such that g and n are coprime.
(2) Note that $f(x) = g^x \mod n$ is a cyclic function because the remainders cycle. Thus, to find the smallest r such that $g^r \equiv 1 \mod n$, we can apply the QFT to the function to find the period (since the period is the same value as r).
(3) Once we know r, we can rearrange:

$$g^r - 1 \equiv 0 \mod n$$

$$(g^{r/2} - 1)(g^{r/2} + 1) \equiv 0 \mod n$$

(4) If r is odd, go back to step 1 and choose another g. If r is even, then $a = g^{r/2} - 1$ and $b = g^{r/2} + 1$ are integers that share factors with n.
(5) Use Euclid's algorithm to find $\gcd(a, n)$, which will give p or q.

Shor's algorithm for the discrete logarithm problem is very similar. Suppose we want to find x such that $g^x \equiv k \mod p$. First, apply QFT to find the smallest integers r, s such that $g^r \equiv 1 \mod p$ and $k^s \equiv 1 \mod p$. Then, set them equal and solve:

$$g^r \equiv k^s \mod p$$

$$g^r \equiv g^{xs} \mod p$$

$$1 \equiv g^{xs-r} \mod p$$

$$xs \equiv r \equiv 0 \mod r$$

Finally, we can use Euclid's algorithm to find $x = \frac{r}{gcd(r,s)}$.

If and when large quantum computers become practical, Shor's algorithm poses a significant threat to all current widely used public-key cryptographic systems. As Shor's algorithm can efficiently break the discrete logarithm problem in polynomial time, it is crucial to explore quantum-resistant schemes. One class of quantum-resistant encryption methods that stands out is isogeny-based encryption, as it employs the shortest keys and the most sophisticated math. While current ECC methods perform computations on elliptic curves, isogeny methods are based on networks of functions between elliptic curves. Before we delve deeper into isogeny-based encryption, we provide a brief introduction to isogenies between elliptic curves.

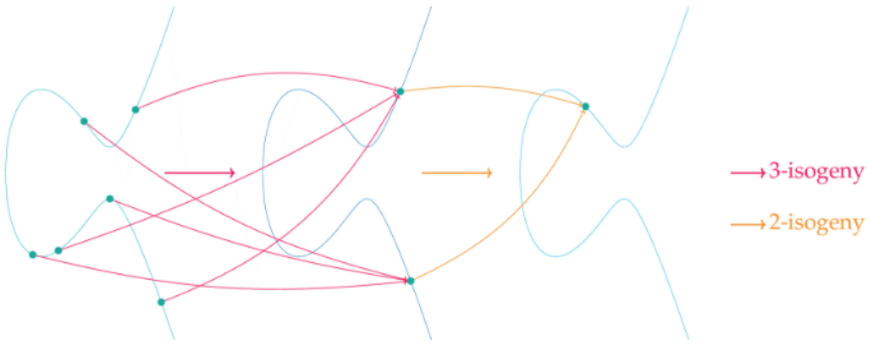
4.2. **Background.**

Definition 9. *Let E_1 and E_2 be elliptic curves over a finite field F_p, where p is a prime power. An* ***isogeny*** *$\phi : E_1 \longrightarrow E_2$ is a rational map between E_1 and E_2.*

It is surjective, so every point in E_1 maps to a point in E_2. ϕ is also a *group homomorphism,* i.e it preserves the group law for points P, Q on E_1:

$$\phi(P+Q) = \phi(P) + \phi(Q)$$

The *kernel* of ϕ, denoted as $\ker(\phi)$, is the set of points on E_1 that map to the identity in E_2. The *degree* for separable isogenies is equivalent to the number of points in the kernel. We won't define separability in this paper, but we will be mostly working with separable isogenies.



Example. Consider the two elliptic curves $E_1 : y^2 = x^3 + x + 1$ and $E_2 : w^2 = z^3 + 16z + 64$. We have the degree-1 isogeny $\phi : E_1 \to E_2$ sending a point $(x, y) \in E_1$ to $(4x, 8y) \in E_2$. To confirm this is true, we can make the substitution:

$$(8y)^2 = (4x)^3 + 16(4x) + 64$$

$$64y^2 = 64x^3 + 64x + 64$$

$$y^2 = x^3 + x + 1$$

Definition 10. *If $E_1 = E_2$ then ϕ is called an endomorphism. The set of endomorphisms under addition and composition from a ring, denoted by End(E).*

In SIDH, the *j-invariant* uniquely characterizes the supersingular elliptic curve isogenies involved in the key exchange process. It is crucial because the security of SIDH relies on the difficulty of computing isogenies between curves with specific j-invariants.

Definition 11. *We define the j-invariant for the elliptic curve $y^2 = x^3 + ax + b$ as:*

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

The two types of elliptic curves we have are ordinary and *supersingular*. The precise definition of supersingular is not important for cryptographic purposes, but what is important is that elliptic curves in the same isogeny class are either all supersingular or all ordinary. SIDH is based on isogenies between supersingular curves.

4.3. **Supersingular Isogeny Diffie-Hellman (SIDH).**

In 2011, De Feo, Jao, and Plût introduced SIDH, a promising quantum-resistant cryptographic protocol which includes entity authentication, key exchange, and public-key cryptography [DFJP11]. The Supersingular Isogeny Key Exchange (SIKE) is just one part of this protocol based on the difficulty of computing isogenies between supersingular elliptic curves. Note that many steps of the key exchange process involve complex isogeny calculations, so in this paper, we review SIKE at a high level:

(1) Alice and Bob decide on the following public parameters:
  (a) A prime p of the form $p = a^e b^d \pm 1$, where a, b are small primes and $d, e \in F_{p^2}$
  (b) A supersingular curve E over F_{p^2}
  (c) Fixed elliptic points P_A, Q_A, P_B, Q_B on E, such that P_A, Q_A, has order a^e and P_B, Q_B has order b^d

(2) Alice and Bob each choose two random integers $m_a, n_a < a^e$ and $m_b, n_b < b^d$ and generate kernels

$$K_A = m_a P_A + n_a Q_A$$
$$K_B = m_b P_B + n_b Q_B$$

(3) They use their respective points R_A, R_B to create isogeny mappings:

$$\phi_A : E \longrightarrow E_A$$
$$\phi_B : E \longrightarrow E_B$$

(4) Using their respective isogenies, Alice and Bob compute the points $\phi_A(P_A), \phi_A(Q_A)$ and $\phi_B(P_B), \phi_B(Q_B)$ on the curves E_A and E_B.

(5) They exchange their public keys. Alice sends Bob $E_A, \phi_A(P_A), \phi_A(Q_A)$, and Bob sends Alice $E_B, \phi_B(P_B), \phi_B(Q_B)$.

(6) With Bob's curve, Alice creates the secret isogeny $\psi_A : E_B \to E_{AB}$ and Bob creates $\psi_B : E_A \to E_{BA}$

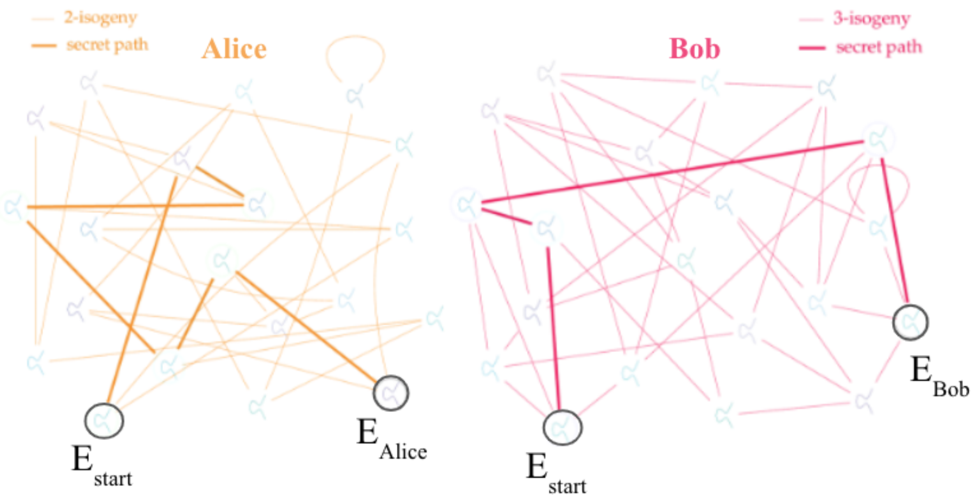


FIGURE 3. In this example, Alice computes a 2^6-isogeny, which corresponds to a walk of length six in her 2-isogeny graph. Similarly, Bob computes a 3^4-isogeny, corresponding to a walk of length four in his 3-isogeny graph.

(7) Each isogeny has kernel K'_A, K'_B respectively:

$$K'_A = m_a\phi_B(P_B) + n_a\phi_B(Q_B)$$
$$K'_B = m_b\phi_A(P_A) + n_b\phi_A(Q_A)$$

(8) Because $j(E_{AB}) = j(E_{BA})$, this helps form the shared secret k between Alice and Bob (a function of k is actually used as the shared secret). [SH19]

Finding an isogeny between two random elliptic curves is conjectured to be extremely hard, even for a quantum computer. However, the problem is that Alice and Bob aren't only exchanging their elliptic curves E_A, E_B; they are also each publicly sharing the auxiliary points $\phi_A(P_A), \phi_A(Q_A)$ and $\phi_B(P_B), \phi_B(Q_B)$. We need the points to run the protocol, but this extra information makes it vulnerable to attacks. On July 22nd, 2022, Wouter Castryck and Thomas Decru published a polynomial time algorithm capable of recovering the secret encryption keys - essentially breaking SIDH [CD23]. What surprised most people was that this was a classical attack, and does not require quantum computers at all. The paper shows how SIDH is vulnerable to a theorem developed by Ernst Kani in 1997 that characterizes reducibility.

A key aspect of the attack is that we target Bob's secret isogeny $\phi_B : E \longrightarrow E_B$, which can be viewed as a secret path in the 3-isogeny graph. In other words, there is a sequence of curves $E \to E_1 \to E_2 \to \cdots \to E_B$ connected by 3-isogenies. Essentially, the attack determines the intermediate curves E_i, and at step i the attack does a brute-force search of all possible $E_i \to E_{i+1}$, eventually determining the private key.

Since SIKE and SIDH were proved as insecure, they will not be standardized for post-quantum public key exchange. However, since the attack assumes that the extra point information that Alice and Bob exchange and the fixed degrees of their respective isogenies are public, many other isogeny-based cryptographic protocols – such as CSIDH, SQISign, and M(D)-SIDH – do not provide all this information and are hence still considered secure! In addition, in December 2016, NIST initiated the process of post-quantum cryptography standardization by issuing a public call for submissions. In August of 2023, following several rounds of selection and evaluation, NIST published the draft documents for the three selected algorithms. Two of these algorithms are lattice-based cryptography, and the third is hash-based [Ber09]. Therefore, despite the vulnerability of SIDH, the resilience of alternative methods and ongoing standardization efforts signal a commitment to advancing secure post-quantum cryptographic solutions.

References

[Ber09] Daniel J. Bernstein. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Bro09] Daniel RL Brown. Standards for efficient cryptography 1 (sec-1). *Standards for Efficient Cryptography*, 2009.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 423–447. Springer, 2023.

[DFJP11] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2011.

[LKT13] Arjen K Lenstra, Thorsten Kleinjung, and Emmanuel Thomé. Universal security: From bits and mips to pools, lakes–and beyond. In *Number Theory and Cryptography: Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*, pages 121–124. Springer, 2013.

[SH19] Vladimir Soukharev and Basil Hess. Pqdh: a quantum-safe replacement for diffie-hellman based on sidh. *Cryptology ePrint Archive*, 2019.

[Sho94] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1994.

[WIN] NOLAN WINKLER. The discrete log problem and elliptic curve cryptography. pages 8–9.

[Woh16] Jeremy Wohlwend. Elliptic curve cryptography: Pre and post quantum. *Recuperado de http://math. mit. edu/ apost/courses/18.204-2016/18.204 Jeremy Wohlwend final paper. pdf*, 2016.

Euler Circle, Mountain View, CA 94040

Email address: `trisha.sabadra@gmail.com`