Elliptic Curves and Applications

Emre Oytun

January 2025

Abstract

In this study, we present elliptic curves and their applications, with a focus on cryptographic implementations. Beginning with the mathematical foundations and properties of elliptic curves, we delve into their significance in finite fields and non-singularity conditions. We examine key algorithms such as point addition and multiplication, as well as their computational complexities. Additionally, we discuss prominent applications of elliptic curves, including Elliptic Curve Cryptography (ECC), Elliptic Curve Digital Signature Algorithm (ECDSA), and Elliptic Curve Diffie-Hellman (ECDH), emphasizing their advantages over traditional cryptographic systems like RSA. This research highlights the practical utility of elliptic curves in ensuring secure communication and data protection.

1 Introduction

This study explores the mathematical foundations and cryptographic applications of elliptic curves. By focusing on the structure and properties of these curves, we highlight their significance in providing secure and efficient solutions to modern cryptographic challenges.

The second section combines fundamental definitions, theorems, and algorithms related to elliptic curves. This includes concepts such as the group law, non-singularity, operations over finite fields, point addition, and point doubling. Additionally, the double-and-add algorithm, a widely used method in elliptic curve computations, is presented along with its computational aspects and complexities.

The final sections focus on the practical applications of elliptic curves in cryptography, including Elliptic Curve Cryptography (ECC), Elliptic Curve Digital Signature Algorithm (ECDSA), and Elliptic Curve Diffie-Hellman (ECDH). These sections demonstrate the advantages of elliptic curves over classical cryptographic systems, particularly in terms of efficiency and security.

2 The Introduction to Elliptic Curves

Definition: [1] An *elliptic curve* over a field K is the set of solutions $(x, y) \in K \times K$ to the equation:

$$y^2 = x^3 + ax + b, \tag{1}$$

where $a, b \in K$, along with a point at infinity, denoted ∞. The curve is *non-singular* if it satisfies the condition:

$$4a^3 + 27b^2 \neq 0. \tag{2}$$

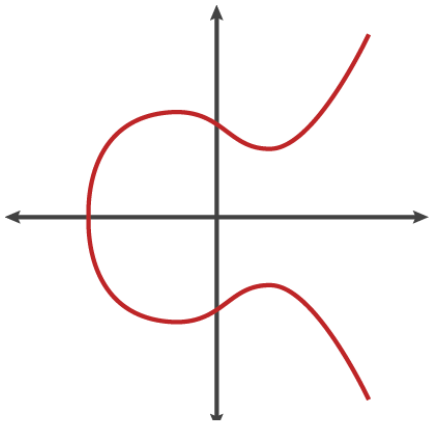

Figure 1: An example of a basic elliptic curve.

Remark: [1] The condition $4a^3 + 27b^2 \neq 0$ ensures that the curve has no cusps or self-intersections, which are crucial for defining a well-behaved group structure on the curve.

Definition: [3] The *point at infinity*, ∞, acts as the identity element for the group of points on the elliptic curve.

Remark: [2] The symmetry property of elliptic curves guarantees that if $P(x, y)$ is a point on the curve, then $-P(x, -y)$ is also on the curve. Formally, this can be expressed as:

$$P(x, y) = -P(x, -y). \tag{3}$$

Definition: [7] A group G is called an *Abelian group* (or commutative group) if it satisfies the following properties:

- **Closure:** For all $a, b \in G$, the operation $a \cdot b \in G$.
- **Associativity:** For all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity Element:** There exists an element $e \in G$ such that for all $a \in G$, $a \cdot e = e \cdot a = a$.

- **Inverse Element:** For every $a \in G$, there exists an element $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.
- **Commutativity:** For all $a, b \in G$, $a \cdot b = b \cdot a$.

Proposition: [2] Elliptic curves form an abelian group under the operation of point addition. The group operation satisfies the following properties:

- **(A1) Closure:** For $P_1, P_2 \in E(K)$, the sum $P_1 + P_2 \in E(K)$.
- **(A2) Associativity:** For $P_1, P_2, P_3 \in E(K)$, $P_1 + (P_2 + P_3) = (P_1 + P_2) + P_3$.
- **(A3) Identity:** The point at infinity ∞ acts as the additive identity, so $P + \infty = P$ for all $P \in E(K)$.
- **(A4) Inverses:** For each point $P \in E(K)$, there exists an inverse point $-P \in E(K)$ such that $P + (-P) = \infty$.

Remark: [2] For an elliptic curve E, any line intersects the curve in at most three points. This property is crucial for defining the group law and ensures that elliptic curves behave predictably under addition.

2.1 Geometric Addition

Definition: [2] The addition operation on an elliptic curve can be visualized geometrically. For points P and Q on the curve, the line passing through P and Q intersects the curve at a third point, $-R$. The point $R = P + Q$ is defined as the reflection of $-R$ across the x-axis.

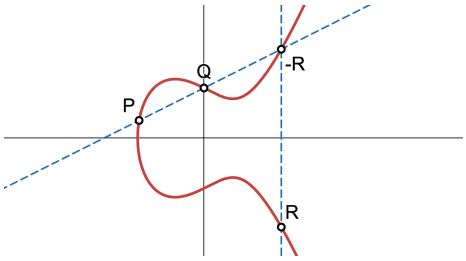

Figure 2: Illustration of $P + Q = R$ on an elliptic curve.

2.1.1 What if $Q = 0$?

Proposition: [2] Let P be a point on the elliptic curve E. The point at infinity 0 serves as the additive identity in the group of points on E. Therefore, for all $P \in E(K)$, we have: $\mathrm{P} + 0 = \mathrm{P}$ and $0 + \mathrm{P} = \mathrm{P}$.

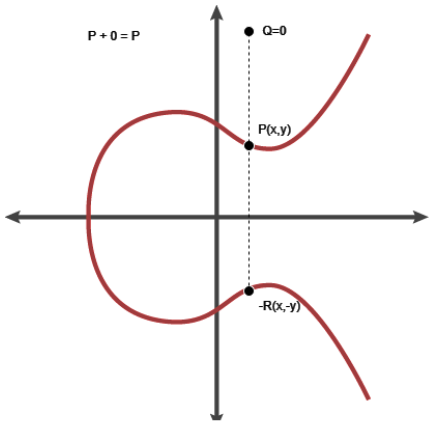

Figure 3: The identity point 0 on the elliptic curve, where $P + 0 = P$.

2.1.2 What if $Q = -P$?

Proposition: [2] Let P be a point on the elliptic curve E. For every point $P \in E(K)$, there exists an additive inverse $-P \in E(K)$ such that:

$$P + (-P) = 0 \quad \text{and} \quad (-P) + P = 0.$$

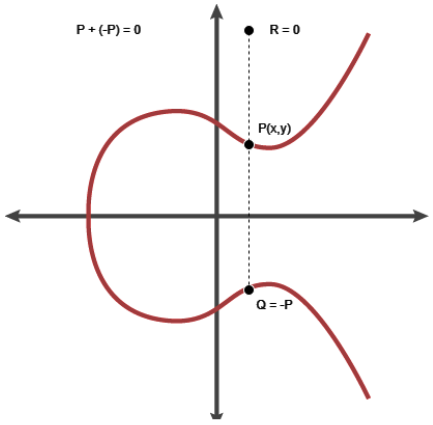

Figure 4: When $Q = -P$, the line through P and Q is vertical.

2.1.3 What if $Q = P$?

Proposition: [2] Let P be a point on the elliptic curve E. When $Q = P$, the sum $P + P$ is computed using the tangent line at P. Formally, for all $P \in E(K)$: P + P = R, where R is the reflection across the x-axis of the third point of intersection of the tangent line with the curve.

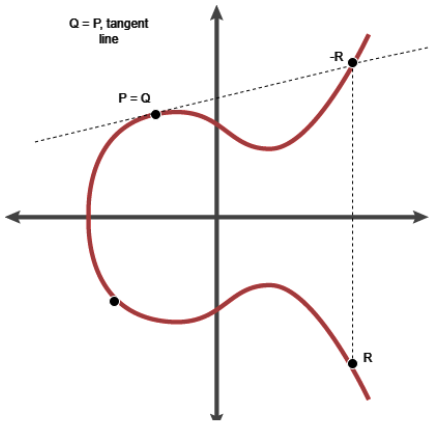

Figure 5: Illustration of the tangent line passing through P when $Q = P$.

Geometrically, this means that the addition of P and P results in a point where the tangent line intersects the curve at a third point. The point of addition is the reflection of that third point across the x-axis.

2.2 Algebraic Method for Point Addition

While the geometric method provides an intuitive understanding of point addition on elliptic curves, we often need to convert this method into an algebraic form for computational purposes. The algebraic method relies on the following formulas for point addition:

Definition: [2,3] Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on the elliptic curve. The sum of these points, $R = P + Q = (x_3, y_3)$, is given by the following steps:

- **Proposition: y_3 is formulated using algebra as below**

  1. First, we compute the slope m of the line joining the points P and Q:

  $$\boxed{m = \frac{y_2 - y_1}{x_2 - x_1}}$$

  2. Next, we calculate the slope of the line joining P and the reflection of Q (denoted as $-Q$) to find y_3. The slope m in this case is:

  $$m = \frac{-y_3 - y_1}{x_3 - x_1}$$

  Rearranging, we get:

  $$-y_3 - y_1 = m(x_3 - x_1)$$

Solving for y_3, we find:

$$\boxed{y_3 = m(x_1 - x_3) - y_1}$$

- **Proposition: x_3 is formulated using algebra as below**

1. Using the equation of the line $y = mx+B$, where B is the y-intercept:

$$y = mx + B$$

2. The equation of the elliptic curve is:

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

Substituting the equation of the line into the curve's equation:

$$(mx + B)^2 = x^3 + ax + b$$

Expanding:

$$x^3 - m^2x^2 + (a - 2mB)x + (b - B^2) = 0$$

3. Since a non-singular elliptic curve has 3 roots, this is a cubic equation. The roots are x_1, x_2, x_3, and we can factorize the cubic equation as:

$$(x - a)(x - b)(x - c) = 0$$

Expanding:

$$(x^2 - (b + a)x + ab)(x - c) = 0$$

This gives:

$$x^3 - (a + b + c)x^2 + (ab + ac + bc)x + abc = 0$$

4. Comparing the two expressions for the cubic equation, we find:

$$m^2 = a + b + c$$

and since $m^2 = x_1 + x_2 + x_3$, we have:

$$\boxed{x_3 = m^2 - x_1 - x_2}$$

Thus, given $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, we can compute the sum $P+Q = (x_3, y_3)$ using these algebraic formulas.

2.3 Point Doubling

Definition: [2,3] Point doubling is an operation equal to the same point addition:

$$2P(x, y) = P(x, y) + P(x, y)$$

Remember, when two points are the same in point addition, there are an infinite number of lines passing through the point P. In this case, we take the tangent line at P.

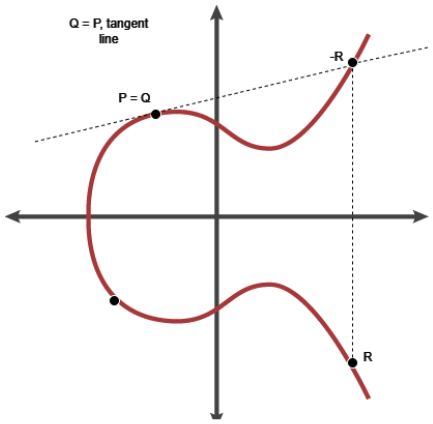

Figure 6: Point Doubling Operation $Q = P$

Proposition: Slope of the tangent line m is formulated as below in the case Q = P

Starting with the elliptic curve equation:

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

Differentiating both sides:

$$2y\,dy = (3x^2 + a)\,dx$$

Solving for $\frac{dy}{dx}$, we find:

$$\boxed{\frac{dy}{dx} = \frac{3x^2 + a}{2y} = m}$$

2.4 Double and Add Algorithm

In applications of elliptic curves, we need to calculate nP where $n \in \mathbb{N}^*$ and P is a point on the elliptic curve.

2.4.1 Brute-Force Double and Add Algorithm

We can calculate nP recursively as shown:

$$2P = P + P$$

$$3P = 2P + P$$

$$\vdots$$

$$nP = (n-1)P + P$$

Complexity:

$$T(n) = T(n-1) + 1$$

$$T(n-1) = T(n-2) + 1$$

$$\vdots$$

$$T(3) = T(2) + 1$$

Summing them all:

$$T(n) = T(2) + n - 2 \in \mathcal{O}(n)$$

Since $n = 2^x$ ($x \in \mathbb{N}^*$), $T(n) \in \mathcal{O}(2^x)$ (exponential complexity).

2.4.2 Greedy Double and Add Algorithm

1. Convert n to its binary representation.
2. Starting from the high-end bit, go to the low-end bit one by one:
   - If the bit is 0, double it.
   - If the bit is 1, double and add.

Example: $26P$

$$26P = 11010$$

Initial: $0P$

$$1 : 2 \cdot 0P + P = P$$

$$1 : 2 \cdot P + P = 3P$$

$$0 : 2 \cdot 3P = 6P$$

$$1 : 2 \cdot 6P + P = 13P$$

$$0 : 2 \cdot 13P = 26P$$

Example: $10P$

$$10P = 01010$$

$$\text{Initial: } 0P$$

$$0 : 2 \cdot 0P$$

$$1 : 2 \cdot 0P + P = P$$

$$0 : 2 \cdot P = 2P$$

$$1 : 2 \cdot 2P + P = 5P$$

$$0 : 2 \cdot 5P = 10P$$

Complexity: Total number of operations = Number of digits in the binary representation = $\log_2(n)$.

$$T(n) \in \mathcal{O}(\log_2 n)$$

Since $n = 2^x$ $(x \in \mathbb{N}^*)$, $T(n) \in \mathcal{O}(x)$ (linear complexity).

2.5 Singularity

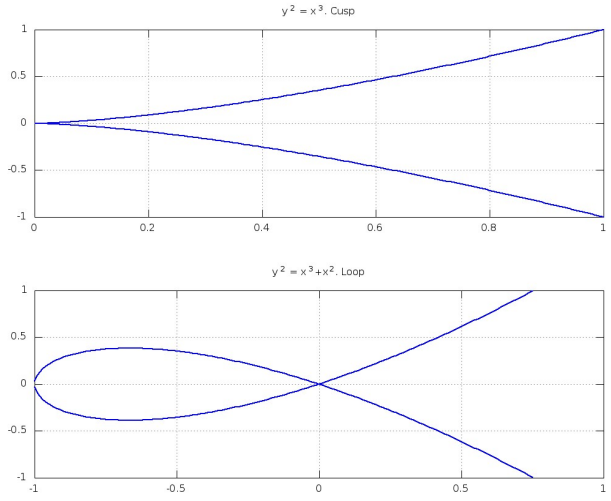

Figure 7: Non-singular curves

2.5.1 Weierstrass Equations

Definition: [2] Weierstrass equation is a two-variable equation $F(x, y) = 0$ forms a curve in the plane:

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

where $a_1, a_2, \ldots, a_i, x, y \in K$ and K is a field, such as $\mathbb{Q}$ (rational numbers), $\mathbb{C}$ (complex numbers), $\mathbb{F}_p$, or $\mathbb{F}_2$.

2.5.2 Singularity Check

Proposition: [2] For Weierstrass equations $F(x, y)$, if $\frac{dF(x,y)}{dx} = \frac{dF(x,y)}{dy} = 0$, then it is a singular curve equation.

Let's apply this singularity check for the elliptic curve E with the equation:

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

The derivatives are:

$$\frac{dE(x,y)}{dx} = 3x^2 + a = 0 \quad \frac{dE(x,y)}{dy} = 2y = 0$$

From $\frac{dE(x,y)}{dy} = 0$, we get:

$$y = 0$$

Substituting this into $\frac{dE(x,y)}{dx} = 0$:

$$x^2 = -\frac{a}{3} \quad x = \left(-\frac{a}{3}\right)^{1/2}$$

Substitute $x = \left(-\frac{a}{3}\right)^{1/2}$ into the original equation:

$$y^2 = x^3 + ax + b = 0$$

$$0 = \left(-\frac{a}{3}\right)^{3/2} + a\left(-\frac{a}{3}\right)^{1/2} + b$$

$$[-b = \left(-\frac{a}{3}\right)^{3/2} + a\left(-\frac{a}{3}\right)^{1/2}]^2$$

$$b^2 = -\frac{a^3}{27} - \frac{a^3}{3} + \frac{2a^3}{9}$$

$$b^2 = -\frac{4a^3}{27}$$

$$27b^2 = -4a^3$$

$$4a^3 - 27b^2 = 0 \quad \boxed{4a^3 - 27b^2 = 0}$$

Therefore, if $4a^3 - 27b^2 \neq 0$, then the elliptic curve is non-singular.

2.5.3 Why Non-Singularity is Required for Elliptic Curves?

1. **It breaks the group law at some points:**
   - In elliptic curve cryptography, we use the elliptic curve's ability to form a group under addition.
   - In this addition operation, we need to find the intersection of lines with the curve and compute new points.
   - Singular points break this group law because the slope of the tangent line at these points is undefined.
2. **Not secure as non-singular curves:**
   - Singular curves often have weaker mathematical structures.
   - They can allow some shortcuts in solving the elliptic curve discrete logarithm problem.

3 Elliptic Curves Over Finite Fields

Definition: [5] A *finite field*, also known as a Galois field, is a field that contains a finite number of elements.

Proposition: [1,2,3] In cryptography, elliptic curves are restricted to finite fields because this way has a stronger mathematical structure. We generally use a finite field as a set of integers modulo p, where p is a prime number. It is generally shown as $\mathbb{F}_p$.

An elliptic curve over a finite field is defined as:

$$E : \{(x, y) \in \mathbb{F}_p^2 \mid y^2 \equiv x^3 + ax + b \pmod{p},\ 4a^3 + 27b^2 \not\equiv 0 \pmod{p}\} \cup \{\infty\}$$

where ∞ is the point at infinity, and a and b are two integers in $\mathbb{F}_p$. This forms a group over $K = \mathbb{F}_p$, and the addition law is the same as we discussed before.

Example: Finding the Points on the Curve

Consider the elliptic curve:

$$y^2 \equiv x^3 + 2x + 3 \pmod{5}$$

We compute the solutions for x as follows:

$$\begin{aligned}
x = 0 &\implies y^2 \equiv 3 \pmod{5} \implies \text{no solution} \pmod{5} \\
x = 1 &\implies y^2 \equiv 6 \equiv 1 \pmod{5} \implies y = 1, 4 \pmod{5} \\
x = 2 &\implies y^2 \equiv 15 \equiv 0 \pmod{5} \implies y = 0 \pmod{5} \\
x = 3 &\implies y^2 \equiv 36 \equiv 1 \pmod{5} \implies y = 1, 4 \pmod{5} \\
x = 4 &\implies y^2 \equiv 75 \equiv 0 \pmod{5} \implies y = 0 \pmod{5}
\end{aligned}$$

The points on the elliptic curve are:

$$(1, 1), (1, 4), (2, 0), (3, 1), (3, 4), (4, 0), \text{and the point at infinity.}$$

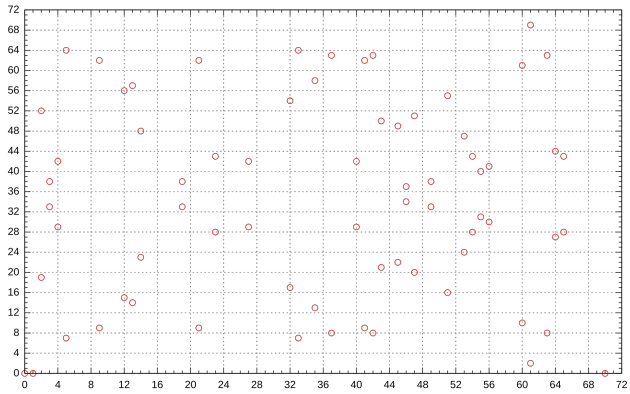

Figure 8: Elliptic curve $y^2 = x^3 - x$ over finite field $\mathbb{F}_{71}$.

3.1 Counting Points on Elliptic Curves over Finite Field

Remark: [2,4] Elliptic curves over finite fields have a finite number of points. We need to calculate these points to calculate the complexity of cryptography algorithms that use elliptic curves over finite fields.

3.1.1 Naive Approach (Brute-force Algorithm)

The easiest method is the naive approach, which tries all possible values for x from 0 to p. However, this is not a feasible way to count the points because it requires $O(p)$ steps, which is difficult since p is a large prime.

3.1.2 Hasse Theorem

The Hasse Theorem states that the order of the elliptic curve group over $\mathbb{F}_p$ must be bounded by the following inequality:

$$p + 1 - 2\sqrt{p} \leq \text{order} \leq p + 1 + 2\sqrt{p}$$

Subtracting the boundaries, we can see the complexity:

$$0 \leq \text{order} \leq (p + 1 + 2\sqrt{p}) - (p + 1 - 2\sqrt{p})$$

$$\text{order} \leq 4\sqrt{p}$$

Therefore, the complexity is:

$$T(n) = 4\sqrt{p} \in O(\sqrt{p})$$

It is still difficult to run that many operations.

3.1.3 Schoof's Algorithm

Schoof's Algorithm is better than the naive approach, but its mathematical background is much more complex. The algorithm combines the Hasse theorem, the Chinese remainder theorem, and polynomial division. The complexity of this algorithm is:

$$T(n) = O(\log_8 p)$$

3.2 Cyclic Subgroup

Definition: [6] A *cyclic subgroup* of a group G is a subgroup $H \subseteq G$ that consists of all powers (under the group operation) of a single element $g \in G$. Formally, if g is an element of G, the cyclic subgroup generated by g is defined as:

$$H = \langle g \rangle = \{g^n \mid n \in \mathbb{Z}\},$$

where g^n represents repeated application of the group operation. For example:

- If the group operation is addition, g^n corresponds to $n{\cdot}g$ (repeated addition of g).

Proposition: [6] A group G is called *cyclic* if it is generated by a single element, meaning $G = \langle g \rangle$ for some $g \in G$.

Proposition: [2] Multiplication of a point by an integer n on an elliptic curve is defined as repeated addition of the point. Specifically, for a point P, we write:

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ times}}$$

This operation has interesting properties in the finite field $\mathbb{F}_p$. Consider the curve:

$$y^2 \equiv x^3 + 2x + 3 \pmod{97}$$

and the point $P = (3, 6)$. Now, let's calculate the multiples of P:

$$\begin{aligned}
0P &= 0 \\
1P &= (3, 6) \\
2P &= (80, 10) \\
3P &= (80, 87) \\
4P &= (3, 91) \\
5P &= 0 \\
6P &= (3, 6) \\
7P &= (80, 10) \\
8P &= (80, 87) \\
9P &= (3, 91)
\end{aligned}$$

From these calculations, we observe two key things: 1. The multiples of P are just five distinct points: the other points on the elliptic curve never appear. 2. The multiples of P repeat cyclically.

Thus, we can write:

$$\begin{aligned} 5kP &= 0 \\ (5k+1)P &= P \\ (5k+2)P &= 2P \\ (5k+3)P &= 3P \\ (5k+4)P &= 4P \end{aligned}$$

Not only do the multiples of P repeat cyclically, but they are also closed under addition. That is, the sum of any two multiples of P is still a multiple of P. This can be shown as follows:

$$nP+mP = \underbrace{P+P+\cdots+P}_{n \text{ times}} + \underbrace{P+P+\cdots+P}_{m \text{ times}} = \underbrace{P+P+\cdots+P}_{n+m \text{ times}} = (n+m)P$$

This demonstrates that the set of multiples of P is closed under addition. Therefore, the set of multiples of P forms a cyclic subgroup of the group formed by the elliptic curve. The point P is referred to as the *generator* or *base point* of the cyclic subgroup.

4 Elliptic Curve Applications

4.1 Elliptic Curve Cryptography (ECC)

- Remark: ECC is used instead of RSA because there are several issues with RSA cryptosystems:
  - Factorization is the trapdoor function in RSA, but it has never been proven to be hard. This requires using huge prime numbers (at least 2048 bits), leading to large private keys.
  - Quantum computing will render RSA obsolete.

ECC	**RSA**
Smaller key sizes for the same security level (e.g., 256 bits ECC = 3072 bits RSA)	Larger key sizes, typically using big prime numbers
Key generation and signing are faster	Operations are slower
Harder to understand and implement securely	Easier to understand and implement
Elliptic Curve Discrete Logarithm Problem (ECDLP) is harder	Prime factorization is simpler

4.2 Elliptic Curve Discrete Logarithm Problem (ECDLP)

- An elliptic curve E is defined by $y^2 = x^3 + ax + b$, where $4a^3 + 27b^2 \neq 0$, over a finite field $\mathbb{F}_p$ of order n.
- Points $P(x, y)$ and $R(x, y)$ are on the elliptic curve.
- Trapdoor function: $xP = R$ (where $1 \leq x \leq n$).
- Given the curve E, prime number p, and points P and R, finding x is computationally hard.
- **Brute-force complexity**:

$$T(n) \in O(n) = O(2^k) \quad \text{(exponential complexity)}.$$

- **Best known algorithm (Pollard's rho or BSGS)**:

$$T(n) \in O(\sqrt{n}) = O(2^{k/2}).$$

- For a 256-bit elliptic curve group:

$$n = 2^{256}, \quad T(n) \in O(2^{128}) \quad \text{(128-bit security level)}.$$

4.3 Elliptic Curve Diffie-Hellman Algorithm (ECDH)

Definition: [2,3] ECDH is a variant of the Diffie-Hellman algorithm for elliptic curves. It is actually a key-agreement protocol. The problem it solves is the following: two parties, Alice and Bob, want to exchange information securely, so that a third party, the Man In the Middle, may intercept them, but may not decode them. Here's how it works:

- **Public values:**
  - Elliptic curve $y^2 = x^3 + ax + b$ with order n ($|E| = n$).
  - Generator point $R(x_r, y_r)$.
  - Public keys K_a and K_b.
- **Key generation:**
  - **Alice:**
    - Chooses $a \in [2, |E| - 1]$.
    - Computes $K_a = a \cdot R(x_r, y_r)$.
  - **Bob:**
    - Chooses $b \in [2, |E| - 1]$.
    - Computes $K_b = b \cdot R(x_r, y_r)$.
- **Key exchange:**

  - Alice sends K_a to Bob.
  - Bob sends K_b to Alice.

- **Private key calculation:**
  - **Alice:** Computes $PK = a \cdot K_b = a \cdot b \cdot R(x_r, y_r)$.
  - **Bob:** Computes $PK = b \cdot K_a = b \cdot a \cdot R(x_r, y_r)$.

- **Complexity:** The attacker needs to find a from K_a or b from K_b, which takes exponential time.

Code Implementation: For a detailed Python implementation of the ECDH algorithm using the Secp256k1 elliptic curve, you can refer to the following GitHub repository [**ecdh.py**]:
`https://github.com/emreOytun/ECDH-ECDSA-Implementations`

Code Explanation:

- The Python code begins by defining the elliptic curve parameters for the Secp256k1 curve, commonly used in cryptographic applications like Bitcoin. These parameters include:
  - a and b: Coefficients of the elliptic curve equation $y^2 = x^3 + ax + b$.
  - G: The base point (generator) of the elliptic curve, provided as a pair of coordinates.
  - p: The prime modulus defining the finite field over which the curve is defined.
  - n: The order of the elliptic curve group.

- The `add_points` function implements point addition, a core operation in elliptic curve cryptography. It handles both cases: adding distinct points and doubling a point.

- The `apply_double_and_add_method` function implements scalar multiplication of the generator G using the efficient double-and-add method. This is used for generating public keys.

- The `main` function demonstrates the ECDH key exchange:
  - Both Alice and Bob generate private keys (k_a and k_b).
  - Using their private keys and the generator point G, they compute their public keys (Q_a and Q_b).
  - They exchange public keys and compute the shared secret key by applying scalar multiplication with their private key and the received public key. The shared key is identical for both Alice and Bob, ensuring a secure exchange.

- The shared secret key is verified to be the same for both parties, demonstrating the success of the key exchange protocol.

For further exploration, the GitHub repository includes additional comments.

4.4 Elliptic Curve Digital Signature Algorithm (ECDSA)

Definition: [2,3] Elliptic Curve Digital Signature Algorithm is an elliptic curve based algorithm used for signing and verifying messages so that sender can sign messages and receiver can verify if a message was sent by a specific user. Here's how it works:

- **Public values:**
  - Elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ with prime p and order n ($|E| = n$).
  - Generator point $R(x, y)$.
  - Alice's public key K_a.
- **Signature creation by Alice:**
  1. Choose $a \in [1, n-1]$, compute $K_a = a \cdot R(x, y) \mod n$.
  2. Hash the message: $Z = \text{SHA(msg)}$.
  3. Choose random $k \in [1, n-1]$, compute $(x, y) = k \cdot R(x, y) \mod n$.
  4. Compute $r = x \mod n$, $s = k^{-1} \cdot (z + r \cdot a) \mod n$.
  5. Signature is (r, s).
- **Signature verification by Bob:**
  1. Verify K_a, r, and s are valid within $[1, n-1]$.
  2. Hash the message: $Z = \text{SHA(msg)}$.
  3. Compute $u_1 = z \cdot s^{-1} \mod n$, $u_2 = r \cdot s^{-1} \mod n$.
  4. Compute $(x, y) = u_1 \cdot R + u_2 \cdot K_a$.
  5. Signature is valid if $r \equiv x \mod n$.
- **Verification Proof:**
  - Recall that $s = k^{-1} \cdot (z + r \cdot a) \mod n$. Rearranging gives:
    $$k = s^{-1} \cdot (z + r \cdot a) \mod n$$
  - From Bob's computations, $u_1 = z \cdot s^{-1} \mod n$ and $u_2 = r \cdot s^{-1} \mod n$, so:
    $$k = u_1 + u_2 \cdot a \mod n$$

- Substituting $k = u_1 + u_2 \cdot a$ into the elliptic curve multiplication:

$$(x, y) = k \cdot R = (u_1 + u_2 \cdot a) \cdot R$$

- Expanding using the properties of elliptic curves:

$$(x, y) = u_1 \cdot R + u_2 \cdot (a \cdot R) = u_1 \cdot R + u_2 \cdot K_a$$

- Since (x, y) was computed as $(u_1 \cdot R + u_2 \cdot K_a)$, the x-coordinate satisfies $r \equiv x \mod n$, proving the signature is valid.

Example:

- Elliptic curve: $y^2 = x^3 - 2x + 15$, $p = 23$, $R = (4, 5)$.
- **Alice:**
  - $a = 3$, $K_a = 3 \cdot (4, 5) = (13, 22)$.
  - $Z = \text{SHA}(\text{msg}) = 10$.
  - $k = 19$, $K = 19 \cdot (4, 5) = (9, 17)$, $r = 9$.
  - Compute $k^{-1} = 17 \mod 23$ (since $17 \cdot 19 \equiv 1 \mod 23$).
  - $s = k^{-1} \cdot (z + r \cdot a) \mod 23 = 17 \cdot 14 \mod 23 = 8$.
  - Signature: $(r, s) = (9, 8)$.
- **Bob:**
  - $Z = 10$, $s^{-1} = 3 \mod 23$.
  - $u_1 = z \cdot s^{-1} \mod 23 = 30 \mod 23 = 7$.
  - $u_2 = r \cdot s^{-1} \mod 23 = 27 \mod 23 = 4$.
  - $(S_x, S_y) = 7 \cdot (4, 5) + 4 \cdot (13, 22) = (9, 17)$.
  - Verify $r = S_x \mod 23$. Signature is valid.

Code Implementation: For a detailed Python implementation of the ECDSA algorithm, you can refer to the following GitHub repository [**ecdsa.py**] :

`https://github.com/emreOytun/ECDH-ECDSA-Implementations`

Explanation:

- The code starts by defining the elliptic curve parameters for the Secp256k1 curve. This includes:
  - a and b: Coefficients of the curve equation $y^2 = x^3 + ax + b$, specific to the Secp256k1 curve.
  - G: The base point (generator), specified by its coordinates (x_0, y_0).
  - p: The prime modulus defining the finite field.

  - n: The order of the elliptic curve group.

- The `add_points` function handles point addition on the elliptic curve, a fundamental operation for signature generation and verification.
- The `apply_double_and_add_method` function performs scalar multiplication using the double-and-add method, an efficient algorithm for elliptic curve computations.
- The `main` function demonstrates the ECDSA process:
  - A private key is generated, and the corresponding public key is computed as $K_a = a \cdot G$.
  - A random key and point are generated for use in the signing process.
  - The message is hashed using the SHA-1 algorithm, and the resulting hash value is converted into an integer.
  - The signature (r, s) is created using the private key, hash value, random key, and the random point's x-coordinate.
  - During verification, the signature (r, s) is validated by reconstructing the random point from the signature and comparing it with the original r.
- The code verifies that the signature is valid if the reconstructed point matches the r coordinate in the signature.

For further exploration, the GitHub repository includes additional comments.

5 Conclusion

In this study, we provided an in-depth exploration of elliptic curves, starting with their mathematical foundations and progressing to their cryptographic applications. Fundamental concepts such as the group law, non-singularity, and operations over finite fields were presented to establish a robust theoretical basis for understanding elliptic curves.

The discussion of algorithms, including point addition, point doubling, and the double-and-add method, demonstrated the computational efficiency of elliptic curves. Moreover, the applications of elliptic curves in cryptographic protocols like ECC, ECDSA, and ECDH emphasized their practical utility and advantages over traditional systems such as RSA.

By combining mathematical theory with practical applications, this study highlights the critical role of elliptic curves in modern cryptography and underscores their potential for addressing emerging security challenges in the digital era.

References

1. [Elliptic Curve in Wikipedia]
   https://en.wikipedia.org/wiki/Elliptic_curve

2. [Youssef El Housni. Introduction to the Mathematical Foundations of Elliptic Curve Cryptography. 2018. ffhal-01914807f]
   https://hal.science/hal-01914807/document

3. [Elliptic Curve Cryptography in Department of Computer Science and Engineering, IIT Madras]
   https://cse.iitkgp.ac.in/~debdeep/pres/TI/ecc.pdf

4. [Counting Points on Elliptic Curves over Finite Field: Order of Elliptic Curve Group Sefik Ilk]
   https://sefiks.com/2018/02/27/counting-points-on-elliptic-curves-over-finite-field

5. [Finite Filed in Wikipedia]
   https://en.wikipedia.org/wiki/Finite_field

6. [Cyclic Group in Wikipedia]
   https://en.wikipedia.org/wiki/Finite_field

7. [MATH 323, Algebra I, Fall 2020 Course notes, Chapter 3, Abelian groups Laurence Barker, Bilkent University]
   http://www.fen.bilkent.edu.tr/~barker/algebranoteschap3