[CS309] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Kunj Thakkar (202251142)

Autumn 2024-2025
Lecture (Week 9&10)

1 Elliptic curve cryptography

- **Introduction**: RSA has been valued for its simplicity and dependability in cryptographic applications, primarily due to the Square and Multiply Algorithm. However, Elliptic Curve Cryptography (ECC) offers a more sophisticated approach to secure communication, characterized by its use of mathematical concepts involving elliptic curves.
- **Elliptic Curve Computations**: Unlike conventional cryptographic techniques that rely on arithmetic with large integers, ECC operates on points defined over elliptic curves. This shift in methodology has facilitated the development of modern cryptographic tools like the Elliptic Curve Diffie-Hellman (ECDH) key exchange and the Elliptic Curve Digital Signature Algorithm (ECDSA). These tools exploit the unique properties of elliptic curves to strengthen security and improve performance.
- **Key Exchange Protocols**: ECC is particularly significant in secure key exchange practices. The Elliptic Curve Diffie-Hellman (ECDH) protocol enables two parties to securely generate a shared secret over an unprotected channel. This method achieves robust security using much smaller key sizes compared to RSA, resulting in quicker calculations and reduced use of computational resources while maintaining strong security.
- **Digital Signature Generation**: Another vital application of ECC is in digital signatures, specifically through the Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA ensures data integrity and verifies authenticity by enabling users to create secure digital signatures. The use of elliptic curves in this context allows for effective signature generation and verification with lower computational demands, without compromising security.
- **Enhanced Security Features**: One of ECC's main strengths is its ability to deliver high levels of security using significantly smaller key sizes than those required by traditional methods like RSA. For instance, a 256-bit ECC key provides a comparable level of security to a 3072-bit RSA key. This efficiency makes ECC an attractive option for applications that require robust security but have constraints on processing power, bandwidth, or storage.
- **The Role of Discrete Structures**: The effectiveness of ECC is rooted in discrete mathematics, particularly through its use of operations over finite fields. Understanding the properties of real numbers can be an essential foundation for grasping the discrete mathematical concepts that underpin ECC. This transition from continuous to discrete structures underscores why elliptic curve cryptography is such a powerful and distinctive tool in modern cryptographic protocols.

Consider two real numbers a and b that satisfy:

$$a, b \in \mathbb{R} \text{ with the condition } 4a^3 + 27b^2 \neq 0$$

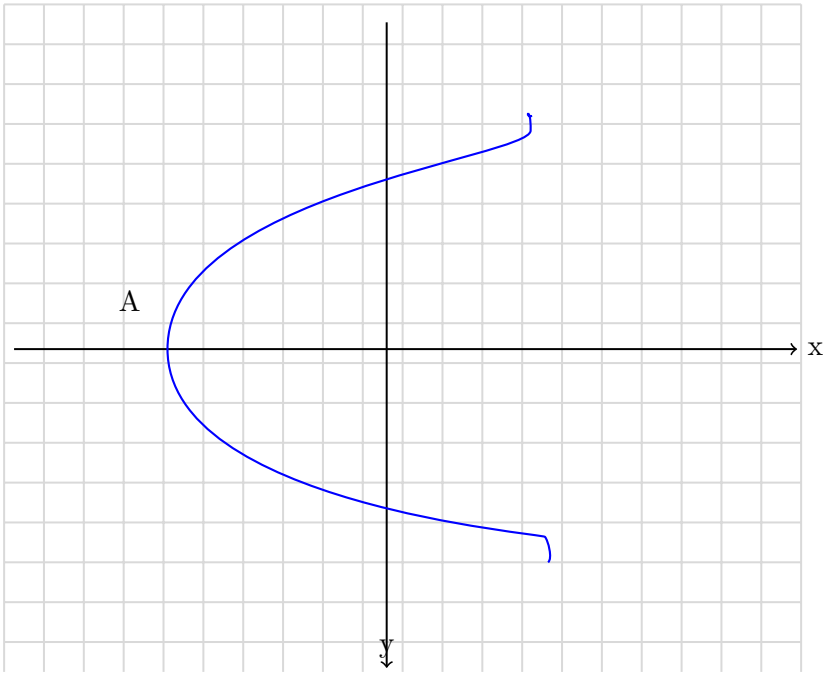
These numbers define an Elliptic Curve with the equation:

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

where (x, y) are points in $\mathbb{R}^2$.

1.1 Visual Representation

The curve can manifest in two distinct forms. Here's the first type:

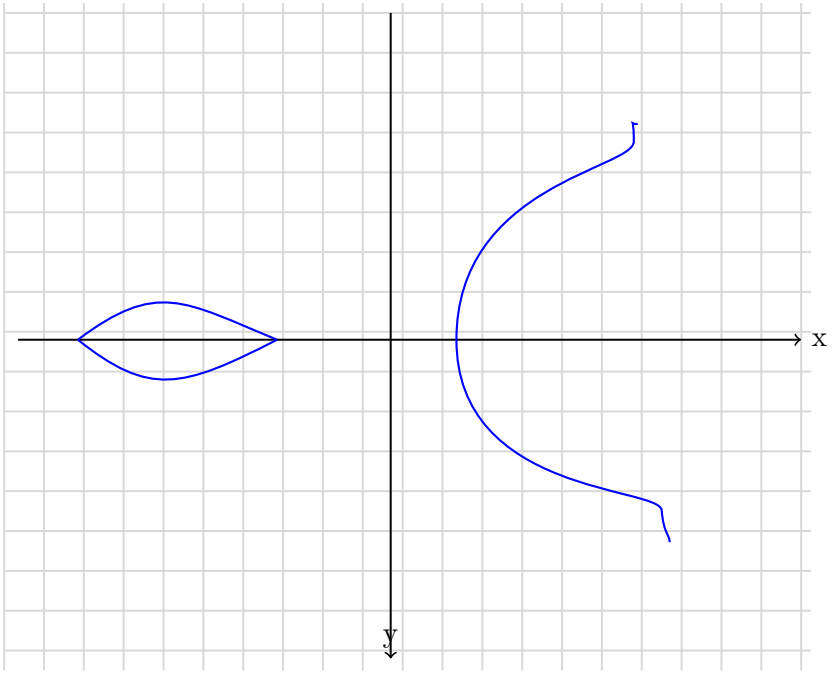


At point A, we have $y = 0$, which gives us the equation $x^3 + ax + b = 0$ (Equation 1). This cubic equation has either:

- Three distinct real roots, or
- One real root and two complex conjugate roots

The discriminant condition $4a^3 + 27b^2 \neq 0$ ensures distinct roots. The curve shown above represents the case with one real and two complex roots.

When Equation 1 has three real roots, the curve appears as:



1.2 Group Operations

The curve exhibits several important properties when we define a binary operator $\boxed{+}$:

1. For points P and Q, $P\boxed{+}Q = R$ is defined geometrically: draw a line through P and Q, find its third intersection with the curve, then reflect that point across the x-axis.
2. We introduce Θ (the point at infinity). When we connect P and -P with a vertical line, we say this line intersects the curve at Θ.
3. Key Properties:
   - $P\boxed{+}(-P) = \Theta$ (inverse elements)
   - $P\boxed{+}\Theta = P$ (identity element)
   - $(P\boxed{+}Q)\boxed{+}R = P\boxed{+}(Q\boxed{+}R)$ (associativity)
   - $P\boxed{+}Q = Q\boxed{+}P$ (commutativity)

For point doubling (when $P\boxed{+}P$), we use the tangent line at P. Where this tangent intersects the curve again, we reflect across the x-axis to get the result: $P\boxed{+}P = 2P = R$.

1.3 Mathematical Formulation

The curve equation:

$$y^2 = x^3 + ax + b$$
$$4a^3 + 27b^2 \neq 0$$

For points $P(x_1, y_1)$ and $Q(x_2, y_2)$, we consider three cases:

1.3.1 Case 1: Point Addition ($x_1 \neq x_2$)

When adding two different points:

$$\begin{aligned} y &= mx + c \text{ (line equation)} \\ m &= \frac{y_2 - y_1}{x_2 - x_1} \text{ (slope)} \\ c &= y_1 - mx_1 = y_2 - mx_2 \text{ (y-intercept)} \end{aligned}$$

Substituting into the curve equation:

$$\begin{aligned} (mx + c)^2 &= x^3 + ax + b \\ x^3 - m^2x^2 + (a - 2mc)x + (b - c^2) &= 0 \end{aligned}$$

The x-coordinate of the result is:

$$\begin{aligned} x_3 &= m^2 - x_1 - x_2 \\ y_3 &= y_1 + m(x_3 - x_1) \end{aligned}$$

Case 2: Point and Its Negative ($x_1 = x_2, y_1 = -y_2$)

When adding a point and its negative:

$$P \boxed{+} Q = \Theta$$

1.3.2 Case 3: Point Doubling ($x_1 = x_2, y_1 = y_2$)

When doubling a point:

$$\begin{aligned} \frac{dy}{dx} &= \frac{3x^2 + a}{2y} \text{ (tangent slope)} \\ m &= \frac{3x_1^2 + a}{2y_1} \\ c &= y_1 - mx_1 \end{aligned}$$

The result coordinates are:

$$\begin{aligned} x_3 &= m^2 - 2x_1 \\ y_3 &= y_1 + m(x_3 - x_1) \\ R &= (x_3, -y_3) \end{aligned}$$

1.4 Modular Arithmetic Version

We can define the same curve over a finite field $\mathbb{Z}_P$ where P is prime:

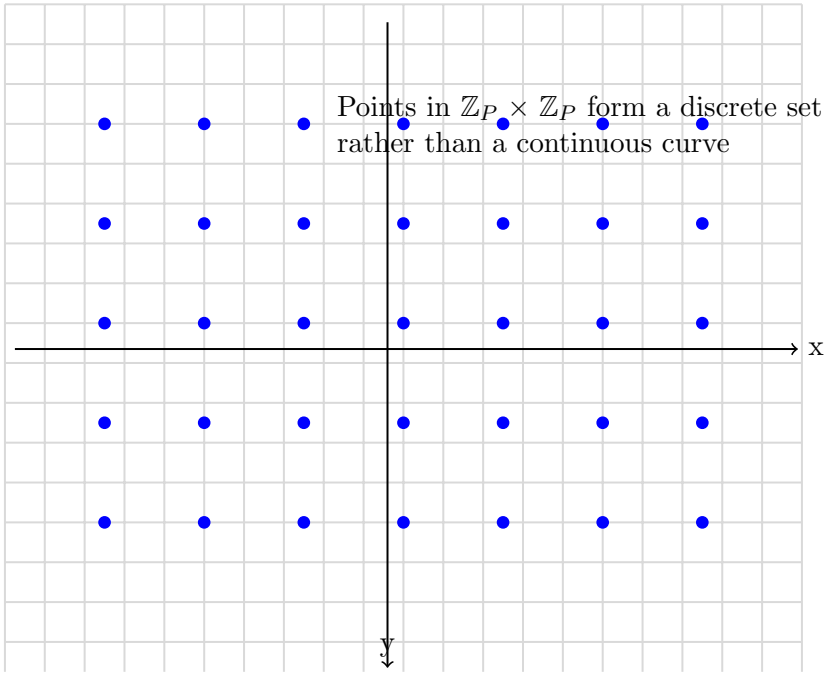
$$\begin{gathered} y^2 = x^3 + ax + b \\ \text{where } (x, y) \in \mathbb{Z}_P \times \mathbb{Z}_P \text{ and } a, b \in \mathbb{Z}_P \\ 4a^3 + 27b^2 \neq 0 \mod P \end{gathered}$$

1.4.1 Point Addition in $\mathbb{Z}_P$

The formulas remain similar, but division becomes multiplication by modular multiplicative inverse:

$$\begin{aligned} m &= (y_2 - y_1)(x_2 - x_1)^{-1} \mod P \\ x_3 &= m^2 - x_1 - x_2 \mod P \\ y_3 &= y_1 + m(x_3 - x_1) \mod P \end{aligned}$$

Note: $(x_2 - x_1)^{-1}$ exists because P is prime, making $\mathbb{Z}_P$ a field.



1.5 Conclusion

The elliptic curve, whether over real numbers or a finite field, forms a commutative group with the defined addition operation. This structure, particularly over finite fields, has important applications in cryptography and other areas of mathematics.

2 The Discrete Logarithm Problem (DLP)

The Discrete Logarithm Problem (DLP) involves finding an integer x such that $0 \leq x \leq (n-1)$, satisfying $\alpha^x = \beta$, given a cyclic group G of order n, a generator α, and an element $\beta \in G$.
To determine x when given α and $\beta = \alpha^x$, one straightforward approach is an exhaustive search. This involves computing α^i for $i = 1$ to n and checking if $\alpha^i = \beta$. However, this method has a time complexity of $O(n)$, which becomes impractical for large n. A more efficient method is the Baby-Step Giant-Step algorithm, which reduces the complexity to approximately $O(\sqrt{n})$.

2.1 Baby-Step Giant-Step Algorithm

First, compute $m = \lceil \sqrt{n} \rceil$. Since $\alpha^n = 1$ holds in a group of order n, we can express x using the Division Algorithm:

$$x = i \cdot m + j, \text{ where } 0 \leq i < m \text{ and } 0 \leq j < m.$$

This implies:

$$\alpha^x = \alpha^{i \cdot m} \cdot \alpha^j = \beta.$$

Rearranging gives:

$$\alpha^j = \beta \cdot (\alpha^{-m})^i.$$

The task now becomes finding i and j within $[0, m)$ such that this equation holds, minimizing the computational complexity.
The algorithm steps are as follows:

1. Compute $m = \lceil \sqrt{n} \rceil$.
2. Construct a table T containing pairs $(j, \alpha^j \mod p)$ for $0 \leq j < m$, sorted by the values of α^j.
3. Compute $\alpha^{-m} \mod p$ and initialize $\gamma \leftarrow \beta$.
4. For $i = 0$ to $m - 1$:
   - Check if γ matches any second component in T.
   - If $\gamma = \alpha^j$, compute $x = i \cdot m + j$.
   - Update γ by multiplying it with α^{-m}.

The space complexity is $O(\sqrt{n})$, and the algorithm's runtime involves $O(\sqrt{n})$ multiplications. Sorting T takes $O(\sqrt{n} \log(\sqrt{n}))$, which simplifies to $O(\sqrt{n} \log n)$.

Example

Consider the example where:

$$G = \mathbb{Z}_{113} \quad (p = 113)$$
$$\alpha = 3$$
$$|G| = 112$$
$$\beta = 57$$

To find x such that $3^x = 57$ and $0 \leq x < \sqrt{112}$:

Step 1: Compute $m = \lceil \sqrt{112} \rceil = 11$.

Step 2: Construct a table for $(j, 3^j \mod 113)$ for $0 \leq j < 11$:

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \mod 113$	1	3	9	27	81	17	51	40	7	21	63

Step 3: Sort the table based on $3^j \mod 113$ values:

j	0	1	8	2	5	9	3	7	6	10	4
$3^j \mod 113$	1	3	7	9	17	21	27	40	51	63	81

Step 4: Compute $3^{-11} = (3^{-1})^{11} \equiv 58 \mod 113$.

Step 5: Compute $r = \beta \cdot (58)^i \mod 113$ for $i = 0$ to 10:

i	0	1	2	3	4	5	6	7	8	9	10
r	57	29	100	37	112	55	26	39	2	3	no-need

Since $r = 3$ matches α^1 in the table for $i = 9$ and $j = 1$, we have:

$$x = 9 \cdot 11 + 1 = 100.$$

Thus, $x = 100$ is the solution.

3 ElGamal Public Key Cryptosystem

The ElGamal Public Key Cryptosystem operates on principles similar to RSA but leverages the Discrete Logarithm Problem for its security foundation. Below, we detail the encryption and decryption processes step-by-step:

1. Choose a prime number p.
2. Define the group $(\mathbb{Z}_p^*, *_p)$.

$$\mathbb{Z}_p^* = \{1, 2, 3, \ldots, (p-1)\}$$
$$x *_p y = x \cdot y \pmod{p}$$

   For $x \in \mathbb{Z}_p^*$, $\gcd(x, p) = 1$, implying the existence of a multiplicative inverse of x modulo p.
3. Choose a primitive element $\alpha \in \mathbb{Z}_p^*$, known as the generator of $\mathbb{Z}_p^*$.

$$\mathbb{Z}_p^* \text{ forms a cyclic group}$$
$$\mathbb{Z}_p^* = \langle \alpha \rangle$$

4. The plaintext space is $\mathbb{Z}_p^*$, and the key space is defined as $\{(p, \alpha, a, \beta), \beta = \alpha^a \pmod{p}\}$.
5. The public key consists of $\{p, \alpha, \beta\}$, while the secret key is $\{a\}$.
6. Select a random number $x \in \mathbb{Z}_{p-1}$, which should remain confidential.
7. **Encryption:** The encryption algorithm produces a ciphertext pair:

$$e_K(m, x) = (y_1, y_2)$$
$$y_1 = \alpha^x \pmod{p}$$
$$y_2 = m \cdot \beta^x \pmod{p}$$

8. **Decryption:** The decryption procedure is as follows:

$$d_K(y_1, y_2) = y_2 \cdot (y_1^a)^{-1} \pmod{p} = m$$
$$y_1^a = (\alpha^x)^a \pmod{p} = \beta^x \pmod{p}$$
$$y_2 \cdot (y_1^a)^{-1} = m \cdot \beta^x \cdot (\beta^x)^{-1} \pmod{p} = m \pmod{p}$$

   The randomness in the ciphertext comes from the random selection of x.

The public key is $\{\beta, \alpha, p\}$, and the security challenge lies in finding a from β and α, which corresponds to solving the Discrete Logarithm Problem. If an adversary can compute g^{ab} from g^a and g^b, the ElGamal scheme is compromised, which reflects the hardness of the Diffie-Hellman Problem without solving the Discrete Logarithm Problem itself.

Example

Consider the following example to illustrate the ElGamal cryptosystem.
Public Parameters:

$$p = 7, \quad \alpha = 4, \quad \beta = 4^3 \equiv 1 \pmod 7$$

Encryption (Alice's Side):

$$\begin{aligned} \text{Message } M &= 5 \\ x &= 2 \\ y_1 &= \alpha^x = 4^2 \equiv 2 \pmod 7 \\ y_2 &= M \cdot \beta^x = 5 \cdot 1 \equiv 5 \pmod 7 \end{aligned}$$

Alice sends the ciphertext $(y_1, y_2) = (2, 5)$ to Bob.

Decryption (Bob's Side):

$$\begin{aligned} \text{Bob's secret key } a &= 3 \\ \text{Received ciphertext } (y_1, y_2) &= (2, 5) \\ y_1^a &= 2^3 \equiv 1 \pmod 7 \\ y_2 \cdot (y_1^a)^{-1} &= 5 \cdot 1^{-1} \equiv 5 \pmod 7 \\ &= M \quad \text{(original message)} \end{aligned}$$

Note: All computations are performed modulo p (in this case, 7).

4 Kerberos (Version 4)

Kerberos is a network security protocol designed to authenticate service requests between trusted hosts over potentially insecure networks, such as the internet. It uses secret-key cryptography and a trusted third-party system to authenticate client-server communications and user identities. Central to this protocol is Symmetric Key Cryptography.

The Kerberos protocol involves three primary components:

- Ticket-Granting Server (TGS)
- Authentication Server (AS)
- Verifier (V)

The authentication process involves communication between a client C and the various servers: the Authentication Server (AS), the Ticket-Granting Server (TGS), and the Verifier (V). The communication flow is as follows:

1. The client initiates the authentication process by logging in and sending the following details to the Authentication Server:

$$\begin{aligned} C \to AS &: ID_c || ID_{TGS} || TS_1 \\ ID_c &\to \text{Identity of Client} \\ ID_{TGS} &\to \text{Identity of TGS} \\ TS_1 &\to \text{Timestamp of the request} \end{aligned}$$

2. Upon receiving the request, the AS encrypts a message and sends it back to the client. The message is encrypted using the shared key SK_c between the AS and the client:

$$AS \rightarrow C : E(SK_c, [SK_{c,TGS} \mid\mid ID_{TGS} \mid\mid TS_2 \mid\mid Lifetime_2 \mid\mid Ticket_{TGS}])$$

Where:

- $SK_{c,TGS}$ is the key used for communication between the client and TGS
- $Lifetime_2$ specifies the ticket's validity period

The ticket $Ticket_{TGS}$ is generated by the AS and includes the following information:

$$Ticket_{TGS} = E(SK_{TGS}, [SK_{c,TGS} \mid\mid ID_c \mid\mid AD_c \mid\mid ID_{TGS} \mid\mid Lifetime_2])$$

Where:

- AD_c is the client's address

The ticket is encrypted using the SK_{TGS} key, known only to AS and TGS. The client can decrypt the message using SK_c, but not the ticket itself since SK_{TGS} is not known to the client.

3. The client uses the session key $SK_{c,TGS}$ to initiate communication with the TGS:

$$C \rightarrow TGS : ID_v||Ticket_{TGS}||Authenticator_c$$
$$Authenticator_c = E(SK_{c,TGS}, [ID_c||AD_c||TS_3])$$

4. The TGS decrypts the ticket and authenticator to verify the client's identity and responds with encrypted information:

$$TGS \rightarrow C : E(SK_{c,TGS}, [SK_{c,v}||ID_v||TS_4||Ticket_v])$$

The $Ticket_v$ is encrypted using the key SK_v, known only to the TGS and verifier.

5. The client sends the ticket and a new authenticator to the verifier:

$$C \rightarrow V : Ticket_v||Authenticator_c$$

6. The verifier decrypts the message to authenticate the client and sends a confirmation:

$$V \rightarrow C : E(SK_{c,v}, [TS_5 + 1])$$