[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Priyansh Vaishnav (202151120) — Lecture 16& 17 (Week 10)

LECTURE 16 Date:- (9th April 2024)

. . . . . .

Generalized Solution (System of Modular Equations):-

We have a system of linear equations:

$$\begin{aligned} a \cdot x &\equiv b \mod m \\ x &\equiv a_1 \mod m_1 \\ x &\equiv a_2 \mod m_2 \end{aligned}$$

From Bezout's Identity, we know:

$$\begin{aligned} a \cdot x_0 + m \cdot y_0 &= \gcd(a, m) \\ x &= x_0 + \frac{m}{\gcd(a,m)} \cdot n \\ y &= y_0 + \frac{a}{\gcd(a,m)} \cdot n \\ y &= y_0 + m_2 \cdot n \end{aligned}$$

We substitute $x = a_1 + m_1 \cdot y$ into the second equation:

$$m_1 \cdot y \equiv (a_2 - a_1) \mod m_2$$

Since $\gcd(m_1, m_2) = 1$:

$$y = y_0 + m_2 \cdot n$$

From $x = a_1 + m_1 \cdot y$:

$$x = (a_1 + m_1 \cdot y_0) + n \cdot m_1 \cdot m_2$$

So, $x \equiv x_0 \mod m_1 \cdot m_2$.

Chinese Remainder Theorem

Consider the system of equations:

$$\begin{aligned} x &\equiv a_1 \mod m_1 \\ x &\equiv a_2 \mod m_2 \\ &\vdots \\ x &\equiv a_r \mod m_r \end{aligned}$$

where $m_1, m_2, \ldots, m_r$ are pairwise coprime.

Define δ_j as:

$$\delta_j = \begin{cases} 1, & \text{if} \quad \text{mod } m_j \\ 0, & \text{if} \quad \text{mod } m_i, \text{ for } i \neq j \end{cases}$$

Then, $x = \sum_{j=1}^{r} a_j \cdot \delta_j$ satisfies the equations.
To find δ_j:

1. Calculate $M = m_1 \cdot m_2 \cdot \cdots \cdot m_r$
2. Calculate b_j, the multiplicative inverse of $\frac{M}{m_j}$ modulo m_j
3. Set $\delta_j = \frac{M}{m_j} \cdot b_j$

The solution is $x = \sum_{j=1}^{r} a_j \cdot \delta_j$ modulo M.
To prove uniqueness:

- Assume x' is another solution.
- Since $x \equiv a_i \mod m_i$ and $x' \equiv a_i \mod m_i$ for $1 \leq i \leq r$, we have $x' \equiv x \mod (m_1 \cdot m_2 \cdot \cdots \cdot m_r)$.
- Thus, the solution is unique modulo $(m_1 \cdot m_2 \cdot \cdots \cdot m_r)$.

Uniqueness Condition: If x' is another solution satisfying the same equations, then $x' \equiv x \mod (m_1 \cdot m_2 \cdot \cdots \cdot m_r)$.

Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) utilizes elliptic curves over finite fields to streamline cryptographic operations. Offering enhanced security with smaller key sizes compared to traditional methods like RSA, ECC is ideal for resource-constrained environments. Algorithms such as Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA) enable secure key exchange and authentication. ECC's reliance on discrete structures aligns with cryptographic principles.

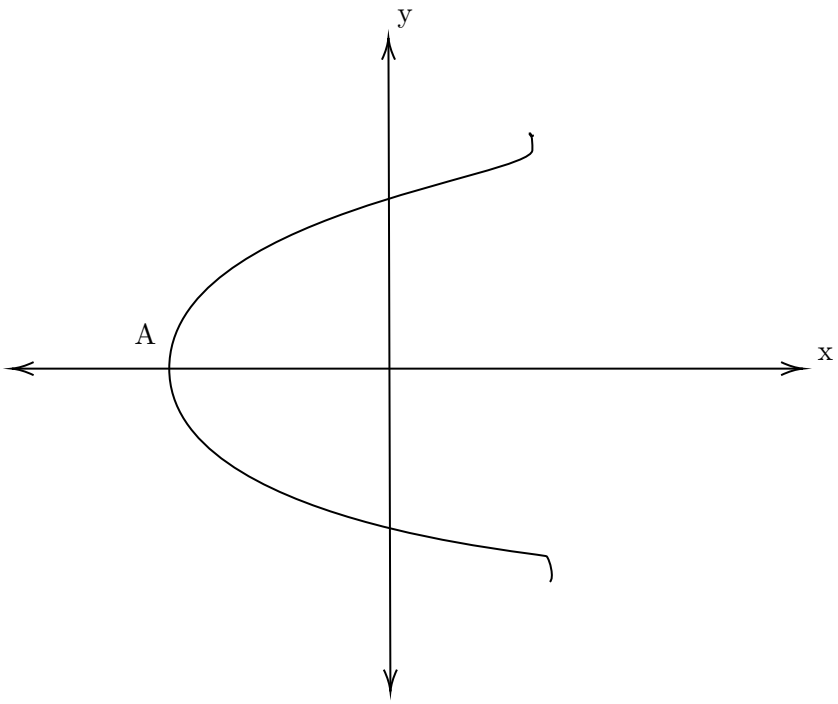
Let's define two number a and b such that,

$$a, b \in \mathbb{R} \; and \; 4a^3 + 27b^2 \neq 0$$

we are goint to define a curve,

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

where $(x, y) \in \mathbb{R}_2$. This curve is called as the Elliptic Curve. If we draw the curve, there will be two structures, one is shown below.

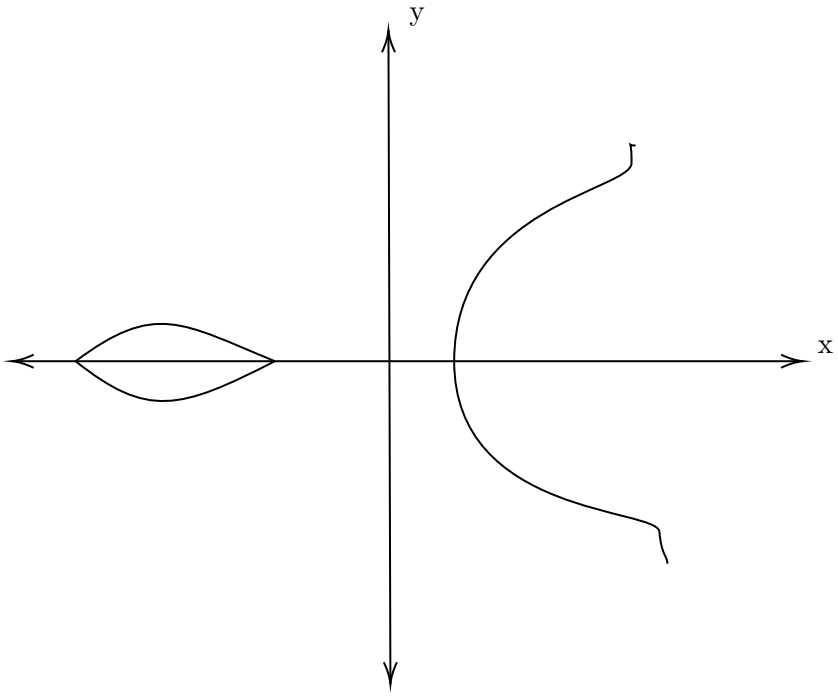


At point A, $y = 0x^3 + ax + b = 0$ (Eq.1). This equation will have three roots and the roots will be either:

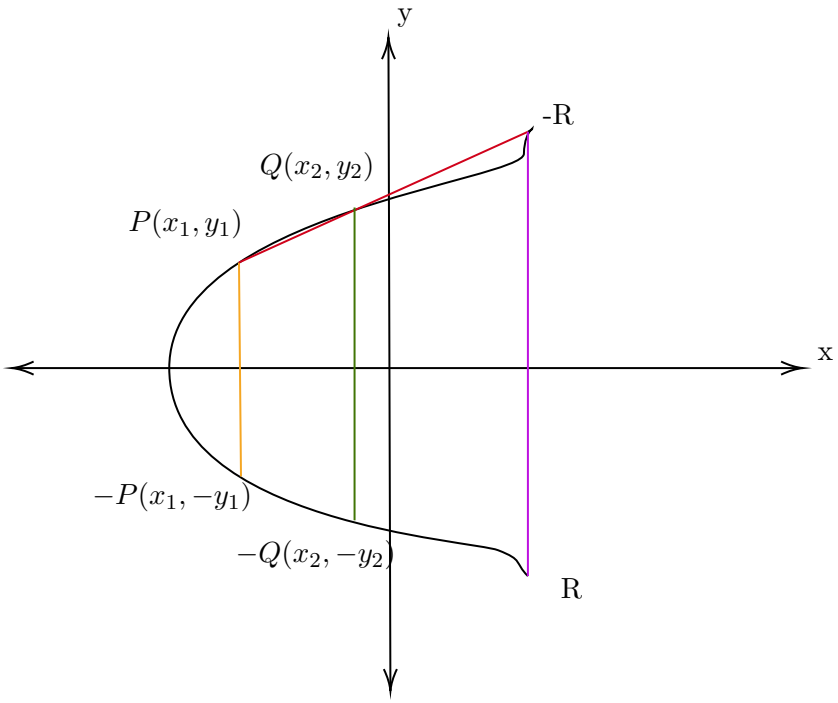
- Three real roots
- One real root, Two complex roots

Eq.1 will have three distinct root iff $4a^3 + 27b^2 \neq 0$ (can be real or complex). If we consider the above curve and put y = 0, we can see that it will have only one real root and two complex roots.

If we consider three real roots of Eq.1, then the curve will look as shown below.



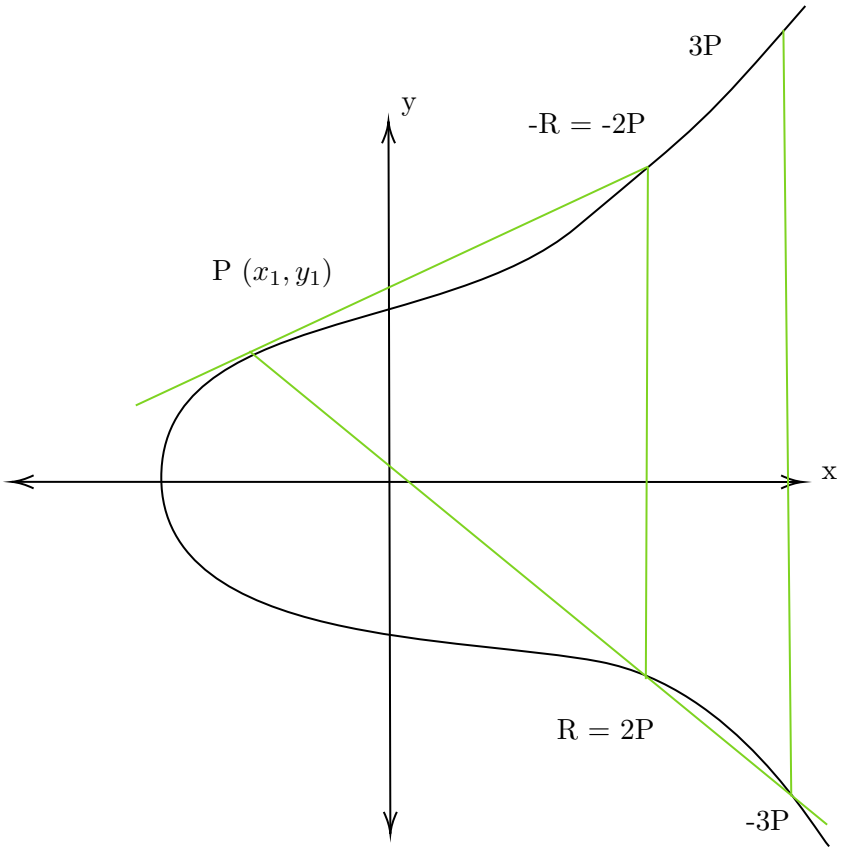
Let us define some properties on the curve we defined before.



Let P and Q be points on the curve. When connected by a straight line, they intersect the curve at $-R$, the mirror image of R with respect to the x-axis. Key properties of the operation $\boxed{+}$ are:

1. $P \boxed{+} Q = R$
2. $P \boxed{+} (-P) = \Theta$
3. $P \boxed{+} \Theta = P$
4. $(P \boxed{+} Q) \boxed{+} R = P \boxed{+} (Q \boxed{+} R)$
5. $P \boxed{+} Q = Q \boxed{+} P$

The operation $\boxed{+}$ forms a commutative group with Θ as the identity element and $-P$ as the inverse of P. To compute $P \boxed{+} P$, draw the tangent at P and find its intersection with the curve. This yields R, where $2P = R$



In the above figure, P and P co-incide and we draw the tangent and then find its image. If we have to find 3P, then 3P $= 2P \boxed{+} P$ as shown in figure. So, for NP, NP $= (N-1)P \boxed{+} P$.

Mathematical Aspects

Elliptic Curve:

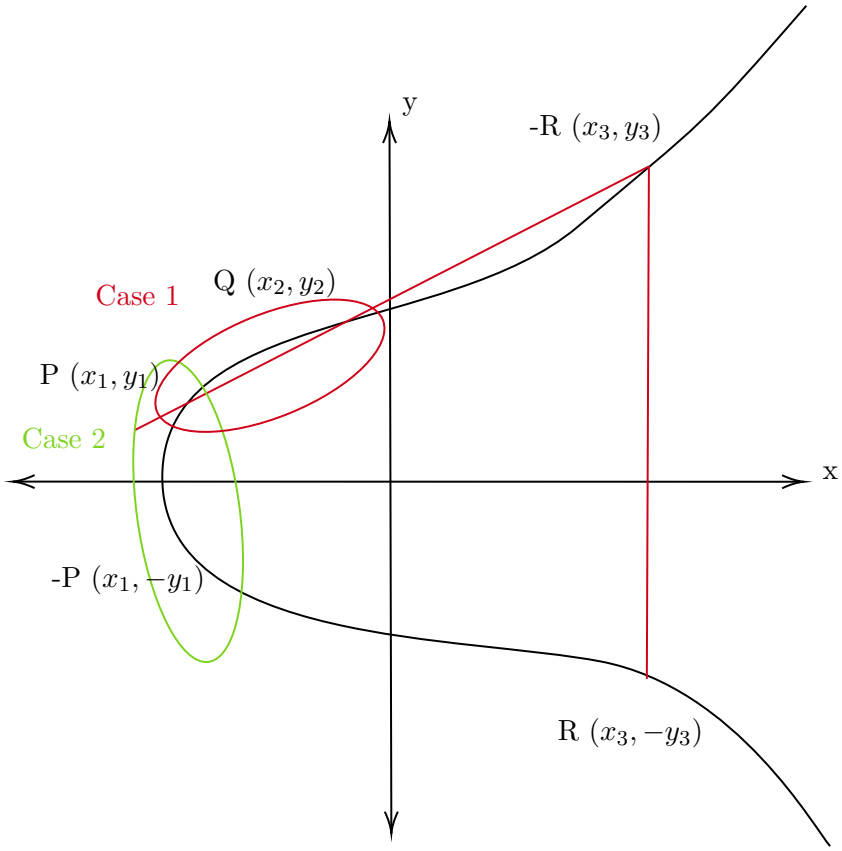
$$\begin{aligned} y^2 &= x^3 + ax + b \\ 4a^3 + 27b^2 &\neq 0 \end{aligned}$$

Let us consider two points P(x_1, y_1) and Q(x_2, y_2). We have three cases,

1. $x_1 \neq x_2$, $y_1 \neq y_2$
2. $x_1 = x_2$, $y_1 = -y_2$
3. $x_1 = x_2$, $y_1 = y_2$

LECTURE 17 Date:- (12th April 2024)

. . . . . .



Case-1:

Given two points P and Q on a curve, we can find the third point R that lies on the same line as P and Q.

$$x^3 - m^2x^2 + (a - 2mc)x + (b - c^2) = 0$$

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = y_1 + m(m^2 - 2x_1 - x_2)$$

$$P \boxed{+} Q \;=\; R$$

Case-2:

$$\mathrm{P} = (x_1, y_1)$$
$$\mathrm{Q} = (x_2, y_2)$$
where $x_1 = x_2, y_1 = -y_2$
In this case $P \boxed{+} Q \;=\; \theta$

Case-3:

$$\mathrm{P} = (x_1, y_1)$$
$$\mathrm{Q} = (x_2, y_2)$$
$$\text{where } x_1 = x_2, y_1 = y_2$$

$$y = mx + c$$
$$y_2 = x_3 + ax + b$$
$$\implies 2y\frac{dy}{dx} = 3x^2 + a$$
$$\implies \frac{dy}{dx} = \frac{3x^2+a}{2y}$$
$$(\frac{dy}{dx})_{(x_1,y_1)} = \frac{3{x_1}^2+a}{2y_1} = \mathrm{m}$$
$$\mathrm{c} = y_1 - mx_1$$

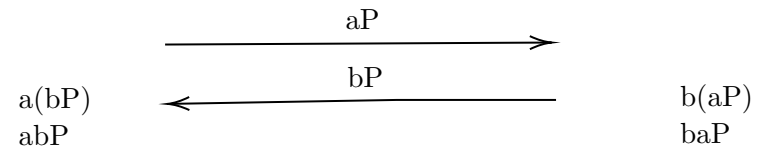
Let us substitute in curve

$$y_2 = x_3 + ax + b$$
$$\implies (mx + c)^2 = x_3 + ax + b$$
$$x_1 + x_2 + x_3 = m^2$$
$$\implies x_3 = m^2 - x_1 - x_2$$
$$m = \frac{y_3 - y_1}{x_3 - x_1}$$
$$\implies y_3 = y_1 + m(x_3 - x_1)$$
$$\mathrm{R}\rightarrow (x_3, -y_3)$$

0.0.1 Elliptic Curve Diffie-Hellman(ECDH)

Let's consider Alice and Bob want to exchange the messages. They have a curve E and a point P and (E, P) which is public.

Alice	Bob
secret: a public: aP	secret: b public: bP



abP=baP

Shared secret key = abP

The security of ECDH relies on the fact that finding x from xP and P is computationally difficult. This problem is known as the Discrete Logarithm Problem on elliptic curves (ECDLP).

RSA Signature

RSA encryption and decryption functions can be defined as follows:

Encryption Function:

$$\text{Enc}(m, e, n) = c \equiv m^e \pmod{n}$$

Decryption Function:

$$\text{Dec}(c, d, n) = m \equiv c^d \pmod{n}$$

RSA signature and verification functions can be defined as follows:

Signature Function:

$$\text{Sign}(m, d, n) = s \equiv m^d \pmod{n}$$

Verification Function:

$$\text{Verify}(m, s, e, n) = \begin{cases} \text{True}, & \text{if } m \equiv s^e \pmod{n} \\ \text{False}, & \text{otherwise} \end{cases}$$

Signing a Message by Alice

1. Alice chooses an elliptic curve E and a base point G on the curve.
2. Alice generates a large prime number n such that $n \cdot G = \mathcal{O}$, where $\mathcal{O}$ represents the point at infinity.
3. Alice selects her secret key d_A.
4. Alice computes her public key $Q_A = d_A \cdot G$.
5. Alice computes the hash of the message: $e = \text{hash}(m)$.
6. Alice selects a random integer k from 1 to $n - 1$.
7. Alice computes $(x_1, y_1) = k \cdot G$.
8. If $x_1 = 0$, Alice goes back to step 6.
9. Alice computes $r = x_1 \mod n$.
10. Alice computes z as the leftmost bits of e such that $\text{len}(z) =$ bit length of n.
11. Alice computes $\sigma = x_1 \mod n$.
12. Alice computes $s = k^{-1}(z + \sigma \cdot d_A) \mod n$.
13. If $s = 0$, Alice goes back to step 6.
14. Alice outputs the signature (r, s).

Verifying a Signature by Bob

1. Bob receives the message m and the signature (r, s) from Alice.
2. Bob computes the hash of the message: $e = \text{hash}(m)$.
3. Bob computes z as the leftmost bits of e such that $\text{len}(z) =$ bit length of n.
4. Bob computes $w = s^{-1} \mod n$.
5. Bob computes $u_1 = z \cdot w \mod n$ and $u_2 = r \cdot w \mod n$.
6. Bob computes the point $(x_1, y_1) = u_1 \cdot G + u_2 \cdot Q_A$.
7. If $(x_1, y_1) = \mathcal{O}$ (the point at infinity) or $r \neq x_1 \mod n$, the signature is invalid.
8. Otherwise, the signature is valid.

Verifying a Signature by Bob

1. Bob receives the message m and the signature (r, s) from Alice.
2. Bob computes the hash of the message: $e = \text{hash}(m)$.
3. Bob computes z as the leftmost bits of e such that $\text{len}(z) =$ bit length of n.
4. Bob computes $w = s^{-1} \mod n$.
5. Bob computes $u_1 = z \cdot w \mod n$ and $u_2 = r \cdot w \mod n$.
6. Bob computes the point $(x_1, y_1) = u_1 \cdot G + u_2 \cdot Q_A$.
7. If $Q_A = \mathcal{O}$ (the point at infinity) or if Q_A does not lie on the elliptic curve E, the signature is invalid.
8. If $(x_1, y_1) = \mathcal{O}$ or $r \neq x_1 \mod n$, the signature is invalid.
9. Otherwise, the signature is valid.