[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Archit Agrawal (202051213)

Winter 2022-2023
Lecture 20 and 21(Week 11)

1 Digital Signature Algorithms

Digital Signature Algorithms (DSA) are used to sign documents and messages digitally. If you're using a DSA, then you will be the only one to sign a document. However, everyone will be able to verify your signature. The formal definition of a DSA is that it is a tuple (P, S, K, Sign, V) where,

P → Plaintext
S → Signature Text
K → Key Space
Sign → Signing Algorithm
V → Verification Algorithm

The signing algorithm Sign, takes plaintext P and a key from key space to generate a signature.

Sign(p, k) = s

The verification algorithm V takes p, s and k and produces output as 1 or 0.

V(p, s, k) = 1 (if s = sign(p,k)) or 0 (otherwise)

One such algorithm used for digital signatures is RSA Signature Algorithm.

1.1 RSA Signature Algorithm

Alice

Sign on m
$n = p \cdot q$
$\phi(n) = (p-1) \cdot (q-1)$
$e \cdot d \equiv 1 \; mod \; \phi(n)$

PK: e, n
SK: p,q,d
$S = m^d \; mod \; n$

$\xrightarrow{S}$

Bob

$v = S^e \; mod \; n$
if v = m, output 1
else output 0

Alice will like to sign on message and m and Bob will like to verify Alice's digital signature. Alice will generate two large primes p and q and he will like to sign on the message m. Using p and q, Alice will compute $n = p \cdot q \; and \; \phi(n) = (p-1) \cdot (q-1)$. Now, Alice will choose an integer e and compute d, such that $e \cdot d \equiv 1 \; mod \; \phi(n)$. Alice will sign on the message in such a way that others can verify. Alice will compute,

Signing Algorithm: $S = m^d \; mod \; n$

Since d is secret key of Alice, only Alice can perform this computation. Alice will send S to Bob. Bob will compute $v = S^e \ mod \ n$ as Bob knows e, public key of Alice. If $v = m$, then output of verification algorithm will be 1 signifying that the signature is verified. Otherwise, it will output 0. This means Bob must know the message m for verification. This message can be transferred using any encryption algorithm.

Notice, in RSA, during encryption, we computed $y = x^e \ mod$, i.e., we used the public key of receiver to encrypt the message, while during decryption, we computed $x = y^d \ mod \ n$, i.e, we used the secret key of the receiver. However, in RSA Signature Algorithm, during signing, we are using the secret key of signee (person who has to sign), while during verification, we are using the public key of signee. This means, in RSA Signature Algorithm the computations are opposite to that in RSA encryption and decryption. In every DSA, signing is done using the secret key of signee and the verification is done using the public key of signee.

1.2 RSA Encryption

Let's revisit the RSA encryprion once again.

$$\begin{gathered} \text{PK: n, e} \\ \text{SK: p, q, d} \\ n = p \cdot q \\ \phi(n) = (p-1) \cdot (q-1) \\ e \cdot d \equiv 1 \ mod \ \phi(n) \\ c = m^e \ mod \ n \end{gathered}$$

If we encrypt two messages using RSA, we get,

$$\begin{gathered} c_1 = m_1^e \ mod \ n \\ c_2 = m_2^e \ mod \ n \end{gathered}$$

If we multiply the two ciphertexts, we will get,

$$c_1 \cdot c_2 = (m_1^e \cdot m_2^e) \ mod \ n = (m_1 \cdot m_2)^e \ mod \ n$$

Here, we can easily observe that $c_1 \cdot c_2$ is the ciphertext for the message $m_1 \cdot m_2$. Therefore, if we have two ciphertexts and we multiply them, we can ensure that it will be the ciphertext for the message $m_1 \cdot m_2$. We are able to compute the encryption of multiplication of two messages without performing encryption actually. This mechanism is known as **Computation on Encrypted Data**.

If we have c_1 and c_2, we can't ensure that $c_1 + c_2$ will be the ciphertext for the message $m_1 + m_2$. However, if we multiply a constant a to the message m (corresponding ciphertext is c), we can find the ciphertext of the message $a \cdot m$ without actually performing the encryption. We will multiply c with a^e and it will be the ciphertext for $a \cdot m$.

There are certain algorithms for which additiom, multiplicaton and constant multiplication are possible. These are known as Fully Homomorphic Encryption Algorithms. RSA is not a fully homomorphic encryption algorithm as addition is not possible over RSA.

Similarly, if we consider two signatures,

$$s_1 = m_1^d \; mod \; n$$
$$s_2 = m_2^d \; mod \; n$$

The signature on the message $m_1 \cdot m_2$ will be,

$$s_1 \cdot s_2 = (m_1 \cdot m_2)^d \; mod \; n$$

We will get signature of multiplication of two messages which will be verifiable by everyone. This is known as **Computation on the Authenticated Data**.

In RSA Signature Algorithm, we are able to compute signature on $m_1 \cdot m_2$ without actually signing, that means, forging a signature on message in RSA Signature Algorithm is easy. We can produce signature on the message $m_1 \cdot m_2$ without actually knowing the secret key of the signee.

To prevent the forging, the sign is done on the hash of the message.

$$s_1 = (h(m_1))^d \; mod \; n$$
$$s_2 = (h(m_2))^d \; mod \; n$$

Now, if we try to multiply the two signatures, we will get,

$$s_1 \cdot s_2 = (h(m_1) \cdot h(m_2))^d \; mod \; n$$

Since, $h(m_1) \cdot h(m_2) \neq h(m_1 \cdot m_2)$, hence,

$$s_1 \cdot s_2 \neq (h(m_1 \cdot m_2))^d \; mod \; n$$

Now, to verify, in verification algorithm, we compute,

$$s_1^e \; mod \; n = h(m_1) \; mod \; n$$

The person verifying the message will know m_1. The verifier will compute $h(m_1)$ and verify if $s_1^e \; mod \; n = h(m_1) \; mod \; n$. If they are equal, the message is verified. The message will be verifiable only with the public key of the signee.

2 System of Modular Equations

We will be solving a system of linear equations of the kind, where we have to find x, such that,

$$a \cdot x \equiv b \; mod \; m \text{ (Eq.1)}$$

Let us first look at the solution of the above equation. The above equation can be written as,

$$a \cdot x - m \cdot y = b \text{ (Eq.2)}$$

where y is an integer. From Bezout's Identiy we know that,

$$a \cdot x_0 + m \cdot y_0 = gcd(a, m) \text{ (Eq.3)}$$

where x_0 and y_0 can be found using Extended Euclidean Algorithm.

The equation 2 is solvable iff gcd(a, m) divides b, otherwise it will not hava a solution. Since, gcd(a, m) divides b, we can say,

$$t \cdot gcd(a, m) = b$$

Multiplying Eq.3 by t, we get,

$$a \cdot (t \cdot x_0) + m \cdot (t \cdot y_0) = t \cdot gcd(a, m) \implies a \cdot X_0 + m \cdot Y_0 = b$$

Therefore, given a equation to solve, first we will check if gcd(a, m) divides b or not. If it is dividing then only a solution exists. Then, we will find x_0 and y_0 using Extended Euclidean Algorithm, and multiply them by $t = \frac{b}{gcd(a,m)}$, to get X_0 and Y_0.

If we know that X_0 and Y_0 are a solution of the Eq.2, then we can substitute x and y as,

$$x = X_0 + \tfrac{m}{gcd(a,m)} \cdot n$$
$$y = Y_0 + \tfrac{a}{gcd(a,m} \cdot n$$

where n is an integer. For any value of n, the above x and y will satisfy the Eq.2. Hence, x and y are general solution for the Eq.2.

Now, let us consider a system of two modular equations,

$$x \equiv a_1 \ mod \ m_1 \text{ (Eq.1)}$$
$$x \equiv a_2 \ mod \ m_2 \text{ (Eq.2)}$$

where m_1 and m_2 are co-prime. We need to find x that satisfies both the equations. If x is a solution of Eq.1, then.

$$x = a_1 + m_1 \cdot y \text{ (Eq.3)}$$

If x is also a solution to Eq.2, then,

$$x \equiv a_2 \ mod \ m_2$$

Let us substitute the value of x from Eq.3,

$$a_1 + m_1 \cdot y \equiv a_2 \ mod \ m_2 \implies m_1 \cdot y \equiv (a_2 - a_1) \ mod \ m_2 \text{ (Eq.4)}$$

Since, gcd(m1, m2) = 1, therefore, from the general solution for only one equation above, we can say Eq.4 has the solution,

$$y = y_0 + \tfrac{m_2}{gcd(m_1,m_2)} \cdot n \implies y = y_0 + m_2 \cdot n$$

From Eq.3 we have,

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

If we have y_0, let's say $x_0 = a_1 + m_1 \cdot y_0$, then,

$$x = x_0 + m_1 \cdot m_2 \cdot n$$
$$x \equiv x_0 \ mod \ m_1 \cdot m_2$$

x_0 is congruent modulo to any x under modulo $m_1 \cdot m_2$. Since, x is the general solution of the two equations, that means, every solution of the given system of equations, will always be congruent to x_0 under modulo $m_1 \cdot m_2$.

2.1 Chinese Remainder Theorem

Consider the system of equations,

$$\begin{gathered} x \equiv a_1 \ mod \ m_1 \\ x \equiv a_2 \ mod \ m_2 \\ . \\ . \\ x \equiv a_r \ mod \ m_r \end{gathered}$$

where $m_1, m_2, \ldots, m_r$ are pairwise co-prime, then the above system has a unique solution modulo $(m_1 \cdot m_2 \ldots m_r)$. The proof for the above result is given below. For each $1 \leq j \leq r$, define δ_j as,

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

Now, we can say that $x = \sum_{j=1}^{r} a_j \cdot \delta_j$ will satisfy the given system of equations. We are claiming that x will satisfy all the equations. Let us expand x,

$$x = \delta_1 \cdot a_1 + \delta_2 \cdot a_2 + \ldots + \delta_r \cdot a_r \text{ (Eq. 1)}$$

Now, consider the j^{th} equation in the given system of equations,

$$x \equiv a_j \ mod \ m_j$$

If we take modulus of x under m_j, we will get,

$$x \equiv (\delta_1 \cdot a_1 + \delta_2 \cdot a_2 + \ldots + \delta_r \cdot a_r) \ mod \ m_j$$

All the $\delta_i, \ for \ i \neq j$ will be 0, and also, $\delta_j = 1$, from the definition of δ. Therefore,

$$x \equiv a_j \ mod \ m_j$$

Hence, we can conclude that x is a solution of the system of equations, given that we know δ_j. Now, let us find $\delta_j \ for \ 1 \leq j \leq r$. Let us say the integer M is equal to:

$$M = m_1 \cdot m_2 \ldots m_r$$

We want to compute the $gcd(\frac{M}{m_j}, m_j)$. Clearly, it will be equal to 1 because m_i are pairwise co-prime. M is product of all m_i, therefore, $\frac{M}{m_j}$ does not contain m_j. Hence,

$$gcd(\tfrac{M}{m_j}, m_j) = 1$$

This means that we can find the inverse of $\frac{M}{m_j}$ under modulo m_j. Let b_j be the multiplicative inverse of $\frac{M}{m_j}$ under modulo m_j. Then,

$$\tfrac{M}{m_j} \cdot b_j \equiv 1 \ mod \ m_j$$

We can use the Extended Euclidean Algorithm to find b_j. Now, $delta_j$ is defined as:

$$\delta_j = \tfrac{M}{m_j} \cdot b_j$$

Clearly, if we divide δ_j by m_j, we will get 1 as remainder. Also, if we take δ_j and divide it by any $m_i, i \neq j$, then remainder will be 0. We can check it as follows.

$$\delta_j \ mod \ m_i = \frac{M}{m_j} \cdot b_j \ (mod \ m_i)$$
$$\delta_j \ mod \ m_i = \{\frac{m_1 \cdot m_2 \ldots m_{j-1} \cdot m_{j+1} \ldots m_i \cdot m_{i+1} \ldots m_r}{m_j} \cdot b_j\} \ mod \ m_i$$
$$\delta_j \ mod \ m_i = t \cdot m_i \ (mod \ m_i) = 0$$

Therefor, the solution is $x = \sum_{j=1}^{r} a_j \cdot \delta_j$, where δ_j is,

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

To solve a given system of equations, follow the steps:

1. Check if m_i are pairwise co-prime. If yes, continue to next step.
2. Calculate $M = m_1 \cdot m_2 \ldots m_r$
3. Calculate b_j for each $\frac{M}{m_j}$ under modulo m_j.
4. Calculate $\delta_j = \frac{M}{m_j} \cdot b_j$
5. Calculate $x = \sum_{j=1}^{r} a_j \cdot \delta_j$
6. Optionally, you can verify your solution by putting the value of x in each equation.

Now, we will prove the uniqueness of the solution. Assume x' is another solution of the given system of equations, then we can say from our conclusion on a system of two equations that, if we have a solution x_0, then every solution will be congruent to x_0. Therefore,

$$x' \equiv x \ mod \ (m_1 \cdot m_2 \ldots m_r)$$

Since, $x \ and \ x'$ are solutions of the system of equations then for any equation we can say,

$$x \equiv a_i \ mod \ m_i$$
$$x' \equiv a_i \ mod \ m_i$$

If we subtract the above two equations, we will get,

$$x' - x \equiv 0 \ mod \ m_i \implies x' \equiv x \ mod \ m_i, 1 \leq i \leq r$$

Since, $(x' - x)$ is divisible by each m_i and m_i are pairwise co-prime, we can conclude that,

$$x' \equiv x \ mod \ (m_1 \cdot m_2 \ldots m_r)$$

Hence, the solution is unique under modulo $(m_1 \cdot m_2 \ldots m_r)$.

3 Elliptic Curve Cryptography

RSA was extremely elementary, we just have to use Square and Multiply Algorithm. Using the Square and Multiply Algorithm, the computations are extremely easy and the methodical things providing security and correctness are also elementary.

Now, we are going to define cryptography known as Elliptic Curve Cryptography. In this, we are going to do the computations on a curve instead of integers. From there, we will be able to see that we can develop Diffie-Hellman Key Exchange Algorithm and Signature Algorithm which are used presently. The key exchange is done using Elliptic Curve Diffie-Hellman (ECDH) and the signatures are done using Elliptic Curve Digital Signature Algorithm (ECDSA). Instead of RSA, we use Elliptic Curve methods because they provide better security using a relatively smaller prime number than RSA.

Let us understand how things work in Elliptic Curve Cryptography. We will begin with real numbers, from which we will develop a discrete structure because cryptography is always based on discrete systems.

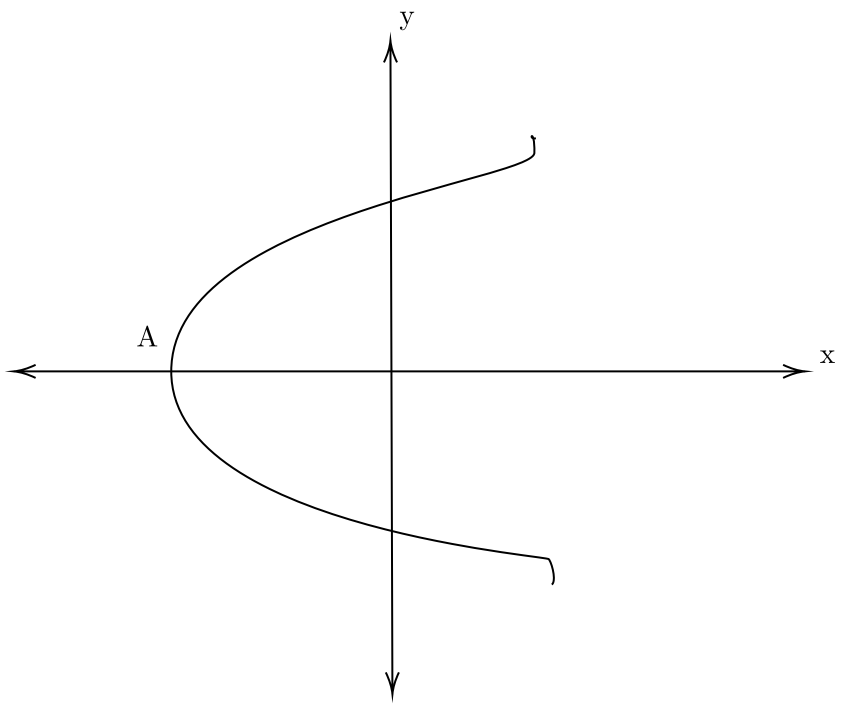
Let's define two number a and b such that,

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

Let us 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, and the other will be discussed later.

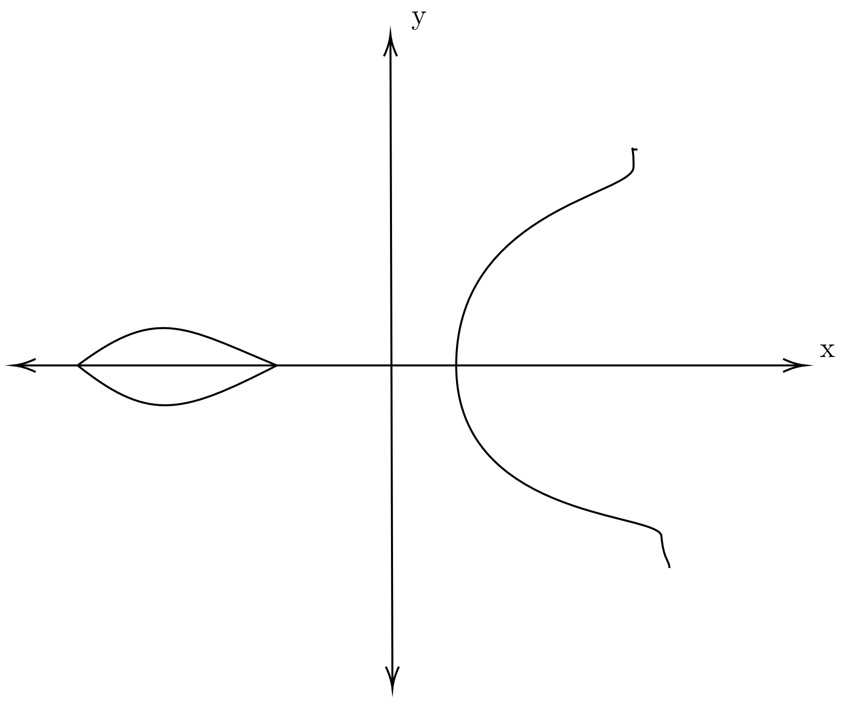


At point A, $y = 0x^3 + ax + b = 0$ (Eq.1). This equation will have three roots and the roots will be either:

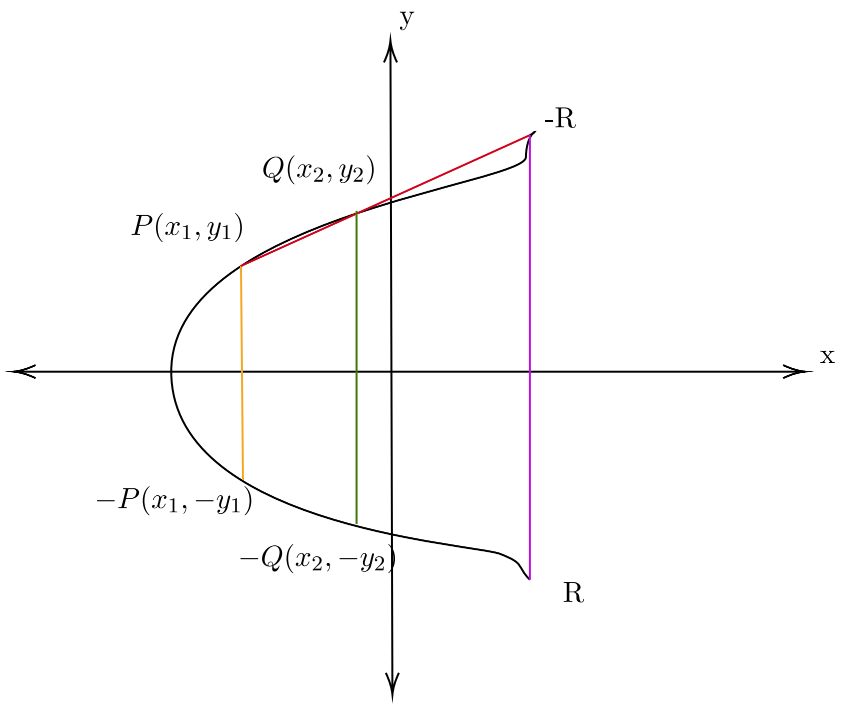
- three real roots
- one real root, two complex roots

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.



If we take two points P and Q on the curve and join them using a straight line, it will cut the curve again at a point, say -R. The point -X is mirror image of X with x-axis as mirror. Alternatively, we can say, the perpendicular from point X on the x-axis will cut the curve again at point -X.

1. $P \boxed{+} Q = R$. The $\boxed{+}$ operation is a binary operator that is defined as: take the two points, join them using a straight line. The line will cut the curve again at some point. The image of this point on the x-axis is the output point.

2. Θ is known as the point of infinity. If we join P and -P, the straight line will be parallel to y-axis. The elliptic curve is infinite curve and we are assuming that the straight line is going to cut the curve at one point which will be the point of infinity.

3. $P \boxed{+} -P = \Theta$

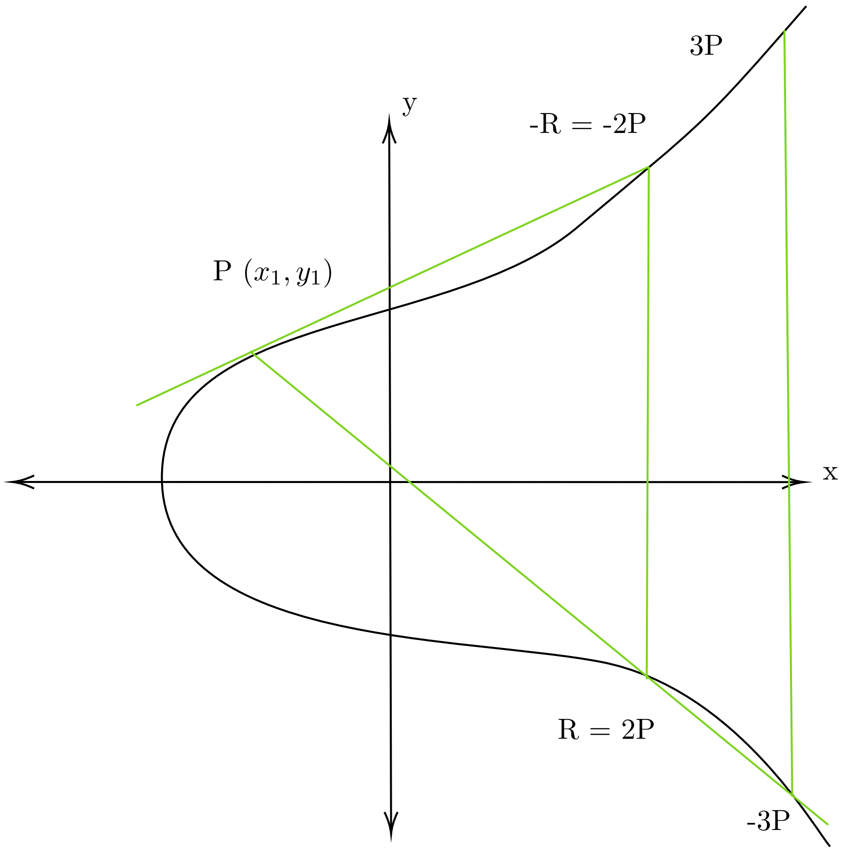
4. $P \boxed{+} \Theta = P$

5. $(P \boxed{+} Q) \boxed{+} R = P \boxed{+} (Q \boxed{+} R)$

6. $P \boxed{+} Q = Q \boxed{+} P$

The associativity and commutativity of the $\boxed{+}$ operator can be proved using a graphical tool. We can think of Θ as an identity element and $-P$ as the inverse of P. Hence, the curve with the $\boxed{+}$ operator is forming a commutative group.

Suppose, we have to find $P \boxed{+} P$, then we draw the tangent to the curve at P, and wherever the tangent cuts the curve again, its image it the x-axis is the result. $P \boxed{+} P = R \implies 2P = R$. Let us see it in the graph:



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 = $(N-1)P \boxed{+} P$.

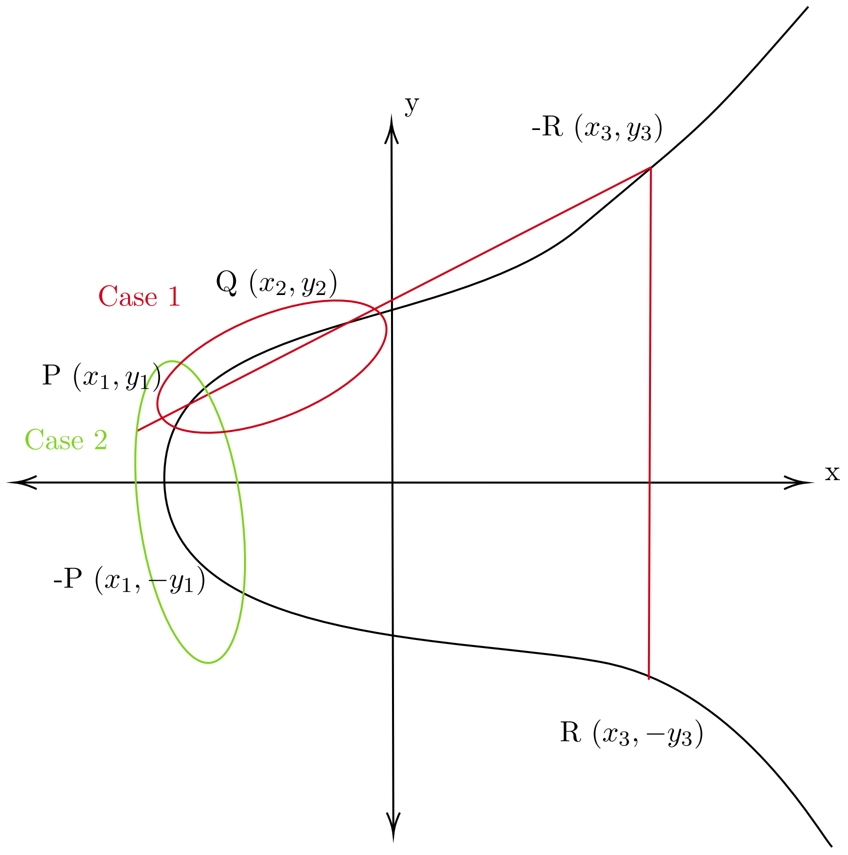
3.0.1 Mathematical Aspects

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. $x_1 = x_2$, $y_1 = y_2$



Case-1:

$$
\begin{gathered}
\text{y = mx + c ...Eqn(a)} \\
\text{m} = \frac{y_2 - y_1}{x_2 - x_1} \\
y_1 = mx_1 + \text{c} \\
\implies c = y_1 - mx_1,\ c = y_2 - mx_2
\end{gathered}
$$

All the points on this line will satisfy this equation of straight line. Equation of straight line(Eqn(a)) will cut the curve at a point, so we substitute value of y in the curve equation.

$$y_2 = x_3 + ax + b$$
$$(mx+c)^2 = x_3 + ax + b$$
$$m^2x^2 + 2mxc + c^2 = x_3 + ax + b$$
$$x^3 - m^2x^2 + (a - 2mc)x + (b - c^2) = 0$$

We already know that $(x_1, y_1), (x_2, y_2)$ will satisfy this equation. If x_3 is another solution of the above system, then,

$$x_1 + x_2 + x_3 = m^2$$
$$\implies x_3 = m^2 - x_1 - x_2$$

We already know that m $= \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_3 - y_1}{x_3 - x_1}$

$$\implies y_3 = y_1 + m(x_3 - x_1)$$

So, we see that we obtained co-ordinate of R(x_3, y_3)

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

Case-2:

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

Case-3:

P $= (x_1, y_1)$
Q $= (x_2, y_2)$
where $x_1 = x_2, y_1 = y_2$

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

Let us substitute in curve

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

Now, let us consider the same curve in $\mathbb{Z}_{\mathbb{P}} \times \mathbb{Z}_{\mathbb{P}}$, where P is a prime number.

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

As we are now working on discrete values, we will not obtain a curve, instead we will obtain points.
Case-1:

$$x^3 = m^2 - x_1 - x_2$$
$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Here we don't divide, we take inverse under modulo P. Since x_2, x_1 are different values, $x_2 - x_1$ will be non-zero and we will be able to find its inverse under mod P as P is prime, so its gcd with $(x_2 - x_1)$ will be 1.

$$m = (y_2 - y_1) \times (x_2 - x_1)^{-1} \; modP$$
$$\implies y_3 = y_1 + m(x_3 - x_1) \; \in \; \mathbb{Z}_{\mathbb{P}}$$

3.0.2 Elliptic Curve Diffie-Hellman(ECDH)

Let us now consider a scenario, where Alice and Bob want to exchange the messages. They have a curve E and a point P and (E, P) is public.

(E, P)

$$\begin{tikzcd}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
\text{secret : a} & & \text{secret : b} \\
\text{public : aP} & & \text{public : bP} \\
{} \arrow[rr, "\text{aP}"] & & {} \\
\text{a(bP)} & & \text{b(aP)} \arrow[ll, "\text{bP}"'] \\
\text{abP} & & \text{baP}
\end{tikzcd}$$

abP=baP

Shared secret key = abP

In the above scenario, E and P were public while a, b were secret. Since a, b and P are all discrete, we can find aP(a times P), bP(b times P) and so on. Since they are discrete, abP and baP are same. Since both Alice and Bob finally reached the same point on the curve, they have successfully exchanged messages.

Note: Security of ECDH depends on the fact that finding xP from P is computationally difficult. This hard problem is known as **Discrete log problem on EC**. Note: We can plot and see the elliptic curve calculations on Jupyter Notebook. We can get the codes from the website called Sagehelp.

3.0.3 Elliptic Curve Digital Signature Algorithm(ECDSA)

Let us first recall the RSA and RSA Signature Algorithm,

- RSA Encryption/Decryption:
  Encryption : c = x^e mod n
  Decryption : x = c^d mod n
- RSA Signature Algorithm:
  Signature : s = x^d mod n
  Verification : x = s^e mod n

In ECDSA, we have (E, P) as public keys.

$$\text{Secret Key : a}$$
$$\text{Public Key : aP}$$

Here, we need

- elliptic curve EC
- a base point-G on the curve. G is such that there exists a large prime number n, such that

$$n = 0$$
$$\implies (n-1)G \boxed{+} G \; = \; nG$$

$$\text{Secret Key : } d_A$$
$$\text{Public Key } Q_A : d_A$$

Now let us see a scenario between Alice and Bob:

(E, G, n) is known to everyone

<u>Alice</u> <u>Bob</u>

secret : d_A
public : $Q_A = d_A G$

messange : m

Let us now see the process of Signature:

1. e = Hash(m)
2. Ƶ$\rightarrow$ L_n leftmost bits of e when L_n is the bit length of n
3. K $\rightarrow$ randomly from [1, n-1]
4. (x_1, y_1) = K·G
5. r = x_1 mod n
   if r = 0 then go to step 3

6. $s\ =\ K^{-1}$ [Ƶ+ $r \cdot d_A$] mod n
   if s = 0, then go to step 3

7. Signature (r, s) on message m

Let us now see verification of ECDSA performed by Bob:

1. Q_A is not equal to 0

2. Q_A lies on the curve EC or not

3. $n \times Q_A = n \cdot (d_A \cdot G) = d_A \cdot (n \cdot G) = 0$

Bob received the message(r, s). To verify, we must follow these steps:

1. verify r, s $\in$ [1, n-1]

2. e = Hash(m)

3. Ƶ $\rightarrow$ L_n leftmost bits of e

4. $u_1\ =$ Ƶ$\cdot s^{-1}$ mod n
   $u_2\ =\ r \cdot s^{-1}$ mod n

5. $(x_2, y_2)\ =\ u_1G\ +\ u_2Q_A$
   if $(x_2, y_2) = 0$, then signature is invalid. Here addition is addition on the curve.

6. If r $\equiv$ x_2 mod n, then signature is valid, otherwise invalid.

Let us see the proof now:

$$\text{c} = u_1G\ +\ u_2Q_A$$
$$\text{c} = u_1G\ +\ u_2d_AG$$
$$\text{c} = (u_1\ +\ u_2d_A)\text{G}$$
$$\text{c} = (\text{Ƶ}\cdot\cdot\cdot^{-1}\ +\ rs^{-1}d_A)\text{G}$$
$$\text{c} = (\text{Ƶ} + rd_A)s^{-1}\text{G}$$
$$\text{Substituting } s^{-1}$$
$$\text{c} = (\text{Ƶ} + r \cdot d_A)(K^{-1}(\text{Ƶ}\ +\ r \cdot d_A))^{-1}\text{G}$$
$$\text{c} = (\text{Ƶ} + r \cdot d_A)(\text{Ƶ}\ +\ r \cdot d_A)^{-1}\text{KG}$$
$$\text{s c} = \text{K}\cdot\text{G}$$

Hence, proved.