[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Dipean Dasgupta (202151188) — Lecture 17,18 (Week 10)

1 Digital Signature Algorithms

Digital signatures on documents and messages are created using Digital Signature Algorithms (DSA). A document will only be signed by you if you are using a DSA. All will be able to confirm your signature, nevertheless. The formal definition of a DSA is that it is a tuple (P, S, K, Sign, V) where,

$$\text{P} \rightarrow \text{Plaintext} \qquad \text{S} \rightarrow \text{Signature Text} \qquad \text{K} \rightarrow \text{Key Space}$$
$$\text{Sign} \rightarrow \text{Signing Algorithm} \qquad \text{V} \rightarrow \text{Verification Algorithm}$$

The signing algorithm Sign, takes plaintext P and a key from key space in order to generate a signature. i.e $Sign(p, k) = s$
The verification algorithm V takes p, s and k and produces output as 1 or 0 which can be written as:

(if s = sign(p,k))
V(p, s, k) = 1
else 0 (othercases)

The RSA Signature technique is one such technique used for digital signatures.

1.1 RSA Signature Algorithm

Alice

$PK : e, n$

$SK : p, q, d$

$S = m^d \pmod{n}$

Bob

$v = S^e \pmod{n}$

if $v = m,$ output 1

else output 0

Bob and Alice would like to confirm Alice's digital signature after she signs a message. He would wish to sign the message m, and Alice will produce two huge primes, p and q. Alice will calculate $n = p \cdot q \; and \phi(n) = (p-1) \cdot (q-1)$ using p and q. To compute d, Alice will now select an integer e such that $e \cdot d \equiv 1 \; mod \; \phi(n)$. Alice is going to sign the message in a way that makes it verifiable to others. Alice will compute,

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

Since d is Alice's secret key, only Alice is able to do this calculation. Bob will receive S from Alice. Using Alice's public key, e, which Bob knows, he will compute $v = S^e \mod$. The verification algorithm's output, 1, indicates that the signature has been validated if $v = m$. If not, it will produce 0. In order for Bob to be verified, he must be aware of the message m. You can transfer this message using any encryption scheme.

As you can see, with RSA, we computed $y = x^e \ mod$, or the recipient's public key, to encrypt the message, and we computed $x = y^d \ mod$, or the recipient's secret key, to decrypt it. However, the RSA Signature Algorithm uses the signee's (the person who must sign) secret key during signing and the signee's public key after verification. This indicates that the calculations for the RSA Signature Algorithm are different from those for RSA encryption and decryption. In each DSA, the signee's public key is used for verification while the signee's secret key is used for signing.

1.2 RSA Encryption

If two messages are encrypted using RSA, we get,

$$c_1 = m_1^e \ mod \ n$$
$$c_2 = m_2^e \ mod \ n$$

If we multiply the two ciphertexts, we will get,

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

This makes it easy to see that the ciphertext for the message $m_1 \cdot m_2$ is $c_1 \cdot c_2$. Consequently, we can guarantee that the ciphertext for the message $m_1 \cdot m_2$ will be obtained by multiplying the two ciphertexts. Without actually encrypting the data, we can calculate the encryption of the multiplication of two messages. The term **Computation on Encrypted Data** refers to this mechanism.

It cannot be guaranteed that $c_1 + c_2$ will be the ciphertext for the message $m_1 + m_2$ if we only have c_1 and c_2. On the other hand, we can get the ciphertext of the message $a \cdot m$ without actually conducting the encryption if we multiply a constant a to the message m (matching ciphertext is c). The ciphertext for $a \cdot m$ will be c multiplied by a^e.

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 shall receive a signature on the multiplication of two messages that can be independently verified by anybody. **Computation on the Authenticated Data** is the term for this.

Forging a signature on a message using the RSA Signature Algorithm is simple since it allows us to compute the signature on $m_1 \cdot m_2$ without actually signing. The message $m_1 \cdot m_2$ can be signed even if we don't truly know the signee's secret key.

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 message's verifier will be aware of m_1. After computing $h(m_1)$, the verifier will ascertain whether $s_1^e \ mod \ n = h(m_1) \ mod \ n$. Verification of the message occurs if they are equal. Only the signee's public key will be needed to validate the message.

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,

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

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.

If gcd(a, m) divides b, equation 2 can be solved; if not, there is no solution. Since b is divided by gcd(a, m), we can say,

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

Multiplying Eq.3 by t, we get,

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

Thus, given an equation to solve, we shall first determine whether or not gcd(a, m) divides b. There is only a solution if it is splitting. Then, using the Extended Euclidean Algorithm, we will determine x_0 and y_0. We will then multiply these values by $t = \frac{b}{gcd(a,m)}$ to obtain 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,

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

where n is an integer. The aforementioned x and y will satisfy Eq. 2 for any value of n. Thus, the generic solutions to Eq. 2 are x and y.

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. It is required 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,

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

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,

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

Under $m_1 \cdot m_2$, x_0 is congruent modulo to any x. Every solution to the above system of equations will always be congruent to x_0 under modulo $m_1 \cdot m_2$ since x is the universal solution of the two equations.

2.1 Chinese Remainder Theorem

Consider the system of equations,

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

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,

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

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

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

Our goal is to compute the $gcd(\frac{M}{m_j}, m_j)$. It will obviously equal 1 as m_i are pairwise co-prime. Since M is the product of all m_i, $\frac{M}{m_j}$ does not include m_j. Hence,

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

This indicates that $\frac{M}{m_j}$ has an inverse that we can find under modulo m_j. Let b_j be $\frac{M}{m_j}$'s multiplicative inverse, 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:

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

It is obvious that the residual after dividing δ_j by m_j will be 1. Moreover, the remainder will be 0 if we divide δ_j by any $m_i, i \neq j$. 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.

Uniqueness of Solution
We shall now demonstrate the solution's uniqueness. Assuming x' to be an additional solution to the provided system of equations, we can infer from our conclusion regarding a system of two equations that all solutions will be congruent to x_0 if we know a solution for 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

We only need to employ the Square and Multiply Algorithms; RSA was really simple in that way. The Square and Multiply Algorithm makes computations incredibly simple, and the rigorous procedures that ensure accuracy and security are similarly simple.

We're going to define elliptic curve cryptography right now. Instead of using integers for the

calculations in this case, we will use a curve. From there, we can see that we can create the currently in use Diffie-Hellman Key Exchange and Signature algorithms. Elliptic Curve Diffie-Hellman (ECDH) is used for the key exchange, and Elliptic Curve Digital Signature Algorithm (ECDSA) is used for signatures. We utilize Elliptic Curve techniques rather than RSA because they offer more security while requiring a comparatively smaller prime value.

Let's examine the operation of elliptic curve cryptography. Since cryptography is always built on discrete systems, we will start with real numbers and create a discrete structure from there.

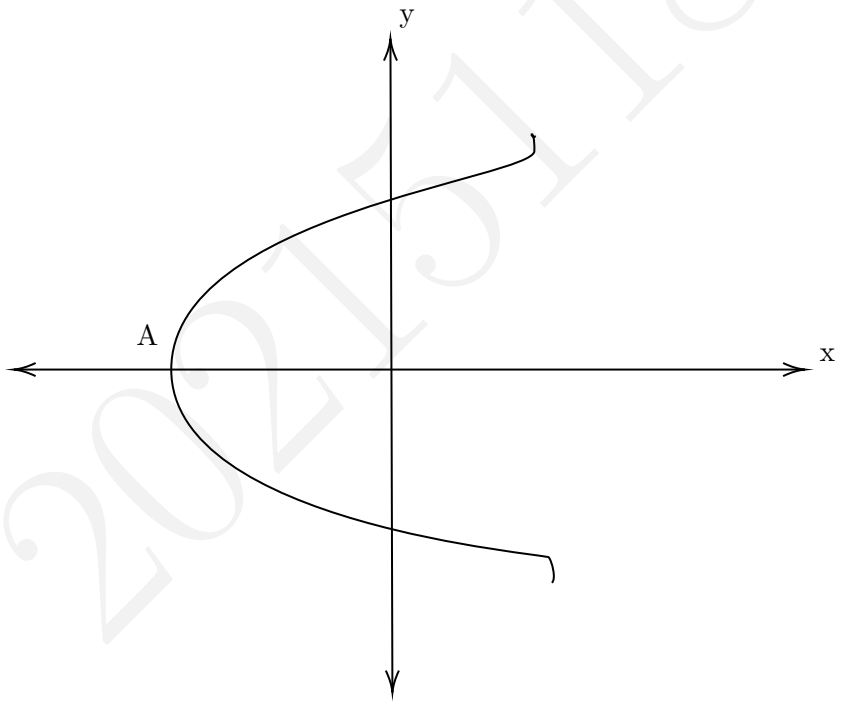
Let's define two number a and b such that,

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

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.

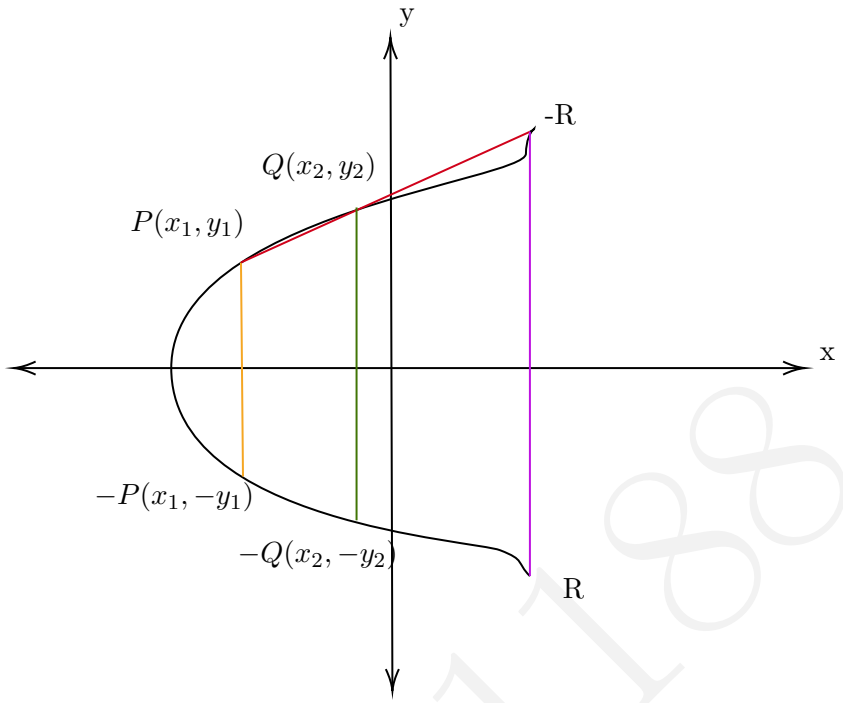


At point A, $y = 0x^3 + ax + b = 0$ (Eq.1). This equation will have three roots and the roots will be either:

- 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.

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

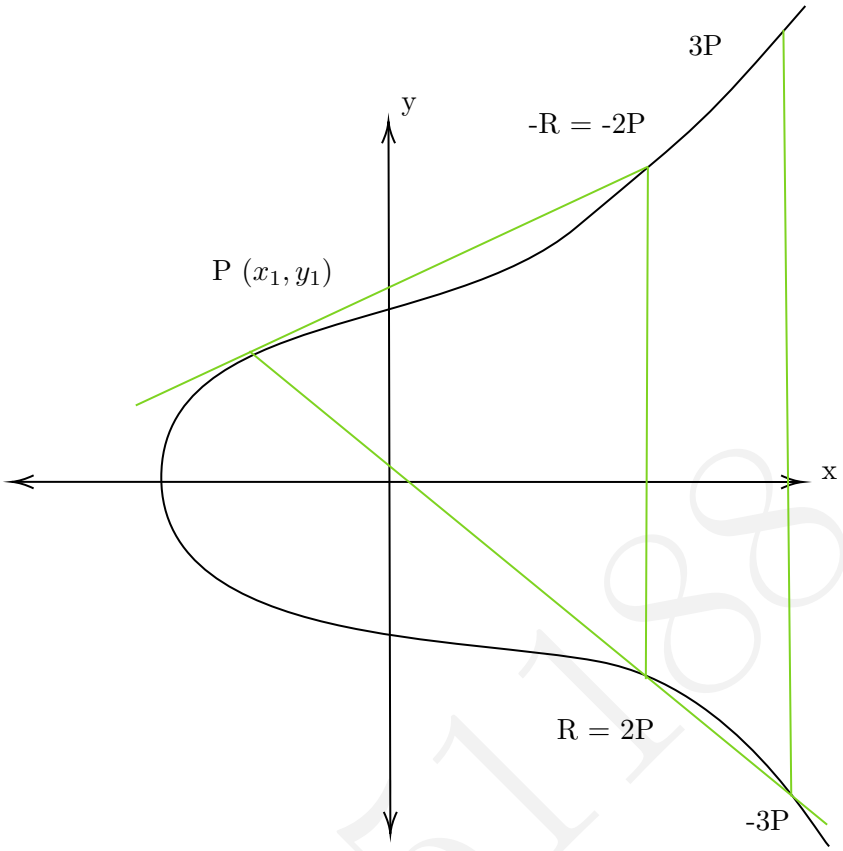


If we unite two points P and Q on the curve with a straight line, we will cut the curve again at a point, say -R. The point -X is the mirror image of X, using the x-axis as a mirror. Alternatively, we might state that the perpendicular from point X to the x-axis will cut the curve again at point -X.

1. $P \boxed{+} Q = R$. The binary operator $\boxed{+}$ is described as taking two points and joining them using a straight line. The line will intersect the curve again at some time. The output point is represented by its picture on the x-axis.
2. Θ is referred to as the point of infinity. If we combine P and -P, the resulting straight line will be parallel to the y-axis. The elliptic curve is infinite, and we assume that the straight line will intersect 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. $P \boxed{+} Q = Q \boxed{+} P$

Using a graphical tool, one may demonstrate the associativity and commutativity of the $\boxed{+}$ operator. Consider Θ as the identity element and $-P$ as its inverse. Hence, the curve with the $\boxed{+}$ operator forms a commutative group.

If we need to find $P \boxed{+} P$, we draw the tangent to the curve at P, and wherever the tangent cuts the curve again, the picture on the x-axis represents the result. $P \boxed{+} = R \implies 2P = R$. Let us see it in the graph:



In the figure above, P and P coincide, and we draw the tangent to obtain its image. To find 3P, we can use the equation $2P \boxed{+} P$, as shown in the picture. So, NP = $(N-1)P \boxed{+} P$.

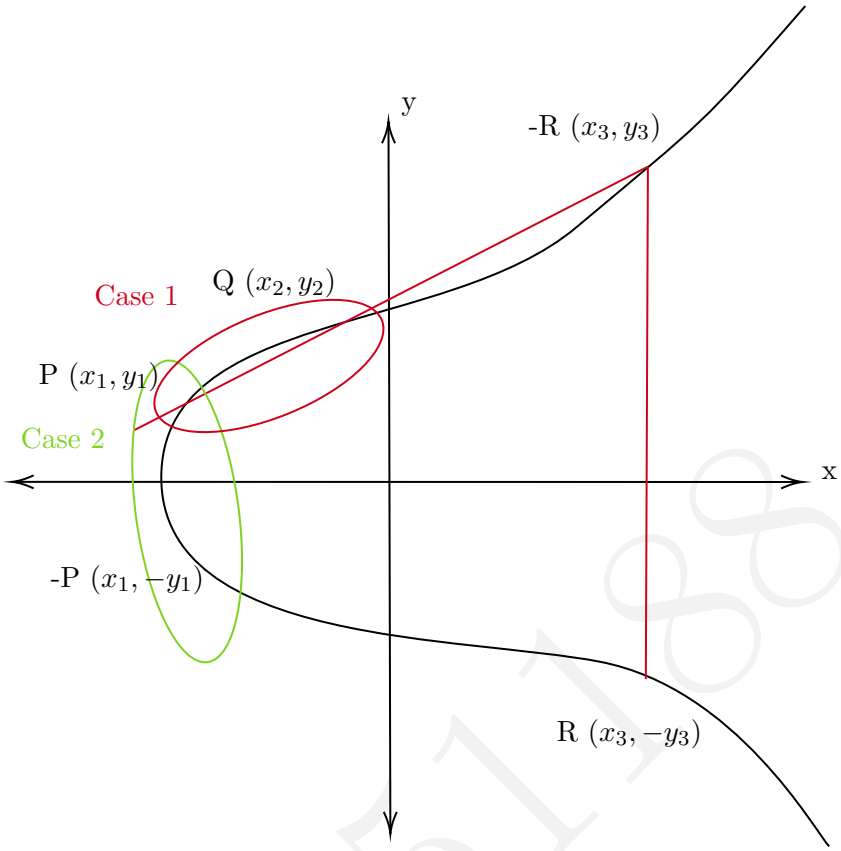
3.0.1 Mathematical Aspects

Elliptic Curve:

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

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:

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

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,

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

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

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

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

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

Case-2:

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

Case-3:

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

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

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:

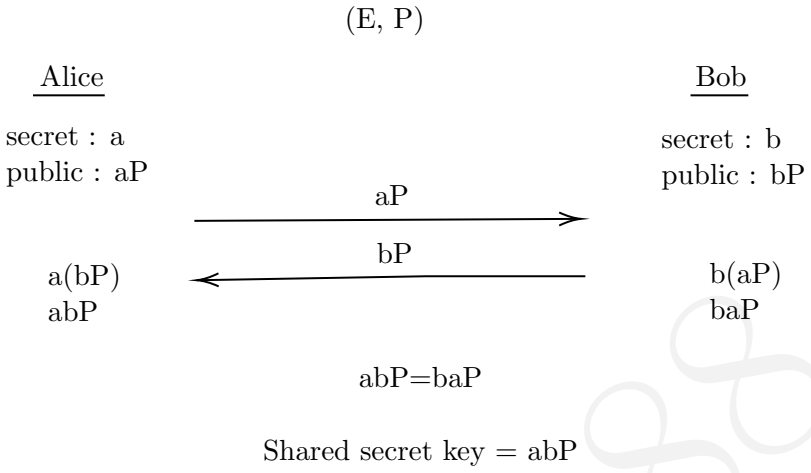
$$x^3 = m^2 - x_1 - x_2$$
$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

We do not divide here; instead, we take the inverse modulo P. Because x_2, x_1 are distinct values, $x_2 - x_1$ will be non-zero, and we will be able to determine its inverse under mod P because P is prime, therefore 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.



In the preceding case, E and P were public, although a and b were secret. Because a, b, and P are all discrete, we may calculate aP (a times P), bP (b times P), and so on. Because they are discrete, abP and baP are identical. Alice and Bob have successfully exchanged messages now that they have reached the same place on the curve.

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.
Secret Key : a
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

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

Secret Key : d_A
Public Key Q_A : d_A

Now let us see a scenario between Alice and Bob:

(E, G, n) is known to everyone

Alice	Bob
secret : d_A public : $Q_A = d_A G$	
messange : m	

Let us now see the process of Signature:

1. e = Hash(m)
2. $\rightarrow$ 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. $s = K^{-1} [+ r \cdot d_A]$ 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. $\rightarrow$ L_n leftmost bits of e
4. $u_1 = \cdot s^{-1}$ mod n
   $u_2 = r \cdot s^{-1}$ mod n
5. $(x_2, y_2) = u_1 G + u_2 Q_A$
   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:

$$\begin{gathered}
\mathrm{c} = u_1 G \ + \ u_2 Q_A \\
\mathrm{c} = u_1 G \ + \ u_2 d_A G \\
\mathrm{c} = (u_1 \ + \ u_2 d_A)\mathrm{G} \\
\mathrm{c} = (\cdots^{-1} \ + \ r s^{-1} d_A)\mathrm{G} \\
\mathrm{c} = (\ + r d_A) s^{-1}\mathrm{G} \\
\text{Substituting } s^{-1} \\
\mathrm{c} = (\ + r \cdot d_A)(K^{-1}(\ + r \cdot d_A))^{-1}\mathrm{G} \\
\mathrm{c} = (\ + r \cdot d_A)(\ + r \cdot d_A)^{-1}\mathrm{KG} \\
\mathrm{s\ c} = \mathrm{K{\cdot}G}
\end{gathered}$$

Hence, proved.