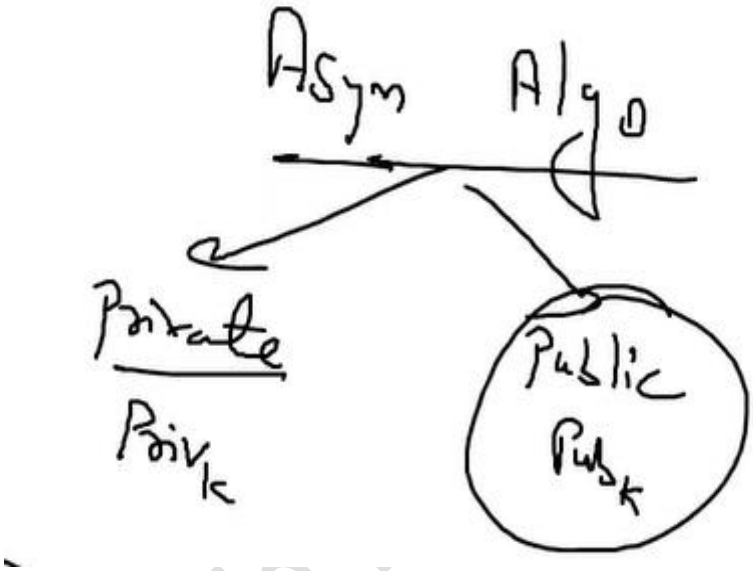
Cryptography Session No.8

Summary 27-07-2022

- Asymmetric Key - Asymmetric Encryption has two keys
  - Public Key
  - Private key



- With Asymmetric key algorithms, We can encrypt a data with public key but to decrypt back this data we will have to use the private key.
  Now to create a "public" and "private key", we will have to first create a private key, and then from the private key, we can create a "public key". This public key we will transfer to the second system through the network, but will keep the "private key" with us only.

- **#openssl genrsa 1024** -> It will create a private key of 1024 bits using the RSA algorithm

- **#openssl genrsa 1024 > myrsa.key** -> It will store the private key in the "myrsa. key" file

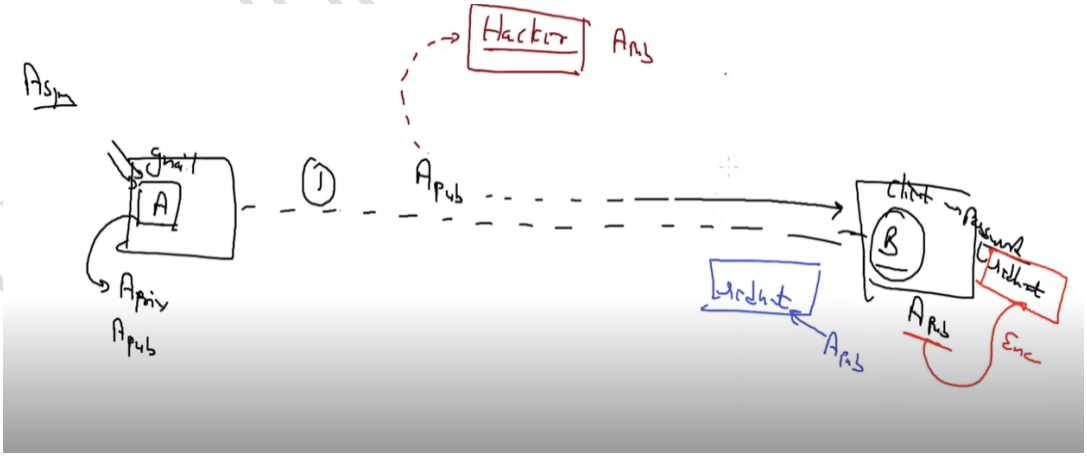
- **#openssl rsa -in myrsa.key -about > mypub.pub** -> It will create a public key from the "private key" and will store the "public key" in "mypub.pub"

- Now as we have both the "private key" and "public key" created, So in the communication of A and B

1) First "A" will share his public key with "B"
2) Now "B" will encrypt his data using this public key
3) "B" will send this encrypted data via n/w to "A"

- Now "A" will decrypt this data using the "private key" he has, even if the hacker in between has sniffed both the "public key" and encrypted data, still he can't decrypt because he doesn't have the private key, So here we have transferred our data very securely.

- Now If "A" has to communicate his data to millions of users, He will send them all his "public keys" They will encrypt their data using this "public key" and will send it back to "A". Now as "A" has the "private key" So it will decrypt all of the user's data. Here "A" will not have to create millions of keys for secure transformation, He will just have to create two keys i.e. "public key" and "private key". So this issue is also solved using symmetric key encryption. But in this setup also we have bugs so here to solve it we will use hybrid encryption.



- **#vim plain.txt** -> "Redhat"

- **#openssl rsautl -encrypt -inkey mypub.pub -pubin -in plain.txt -out cipher.txt** -> Now it will encrypt the data in "plain.txt" using the public key and will store the encrypted cipher text in "cipher.txt"

- **#vim cipher.txt** -> If we try to view this, it would be encrypted data.

- **#openssl rsautl -decrypt -inkey myrsa.key -in cipher.txt** -> It will decrypt now our data in the cipher.txt file using the"private key".

- If we try to decrypt the data in "cipher.txt" with the wrong private key then we receive this error

- Now we have two issues in our previous data transfer
  - The asymmetric key is very slow in performance so encrypting and decrypting big data with the asymmetric key would be a very time taken process, And nowadays we have to transfer videos, and images a lot via N/w in a secure way, So using asymmetric key encryption here it will slow down the process. But Symmetric key algorithms are quite fast in comparison to Asymmetric algorithms.
    #openssl speed rsa -> it will tell the average speed of the RSA algorithm (asymmetric key algorithm)
    #openssl speed aes-128-cbc -> it will tell the average speed of AES algorithm (symmetric key algorithm)
  - Here in our previous connection only "B" can transfer data to "A", but if "A" has to transfer some data to "B" then "B" doesn't have the public key of "A" so it can't transfer data, so in this case, only one-way communication would be done but we need two-way communication
- To solve the above two problems we have another very powerful encryption **Hybrid encryption**

- In **Hybrid Encryption**
    1) 'B' will send a request to 'A'
    2) Then 'A' will send its public key to 'B'
    3) Now 'B' will create a symmetric key and he will encrypt that symmetric key with that public key of 'A' and will send it to 'A'.
    4) Now 'A' will receive this encrypted data, 'A' will decrypt it with his private key, and hurray he has receive the "symmetric key of B" very securely
    5) Now further communication between 'A' and 'B' will be done by encrypting data with the symmetric key.

    So here we have used an asymmetric key just to exchange the "symmetric key" between 'A' and 'B', And now as both A and B have the symmetric key so both can communicate securely with each other by encrypting their data using symmetric key.

- The formula behind the RSA algorithm is **n=p*q**, where p and q are prime numbers now 'p' and 'q', are private keys here and 'n' is the public key. So if we take "n" very big, then guessing the value of p and q would be very hard, if you try to guess by brute-force attack it will take a lot of years, So it(RSA algorithm) is very secure although Quantum computing algorithm like Shor's algorithm has ways to find the p and q (private keys) just by knowing "n" in very limited time.