Public Key Cryptography

- In symmetric cryptography we use same key for encryption and decryption.

- In asymmetric we use different keys for encryption and decryption.

- In symmetric cryptography symbols are permuted and substituted.

- Asymmetric is based on applying mathematical functions to numbers.

Public Key Cryptography

- Each user has a pair of keys:

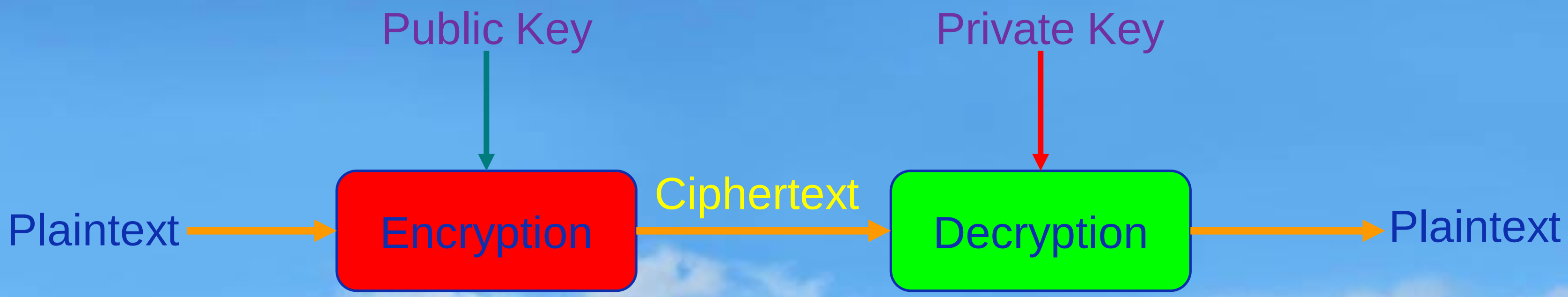
  i. Public key - Known to all and is used to encrypt messages.

  ii. Private key – Known only to the owner and is used to decrypt messages.

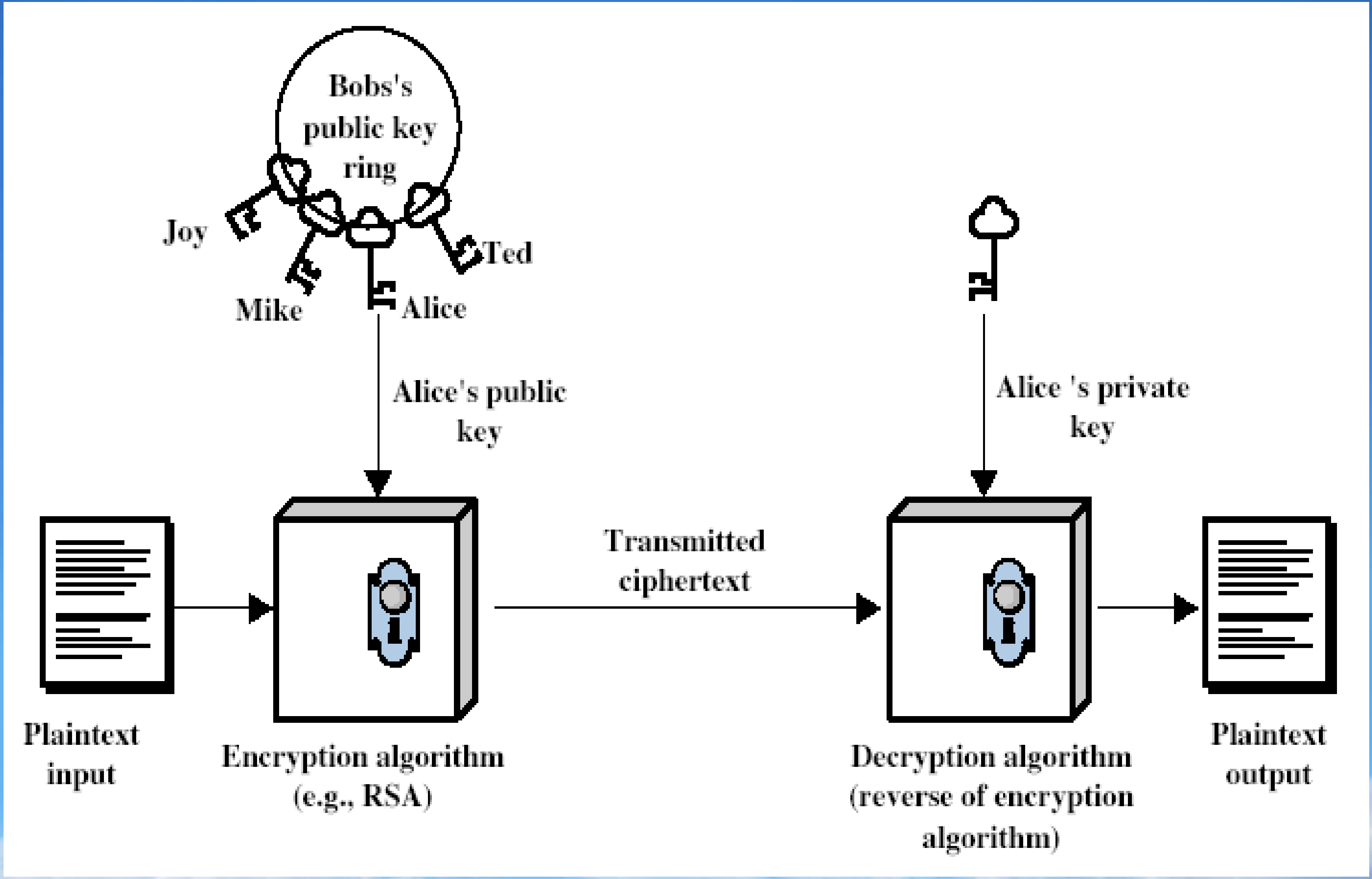
Public Key Cryptography

- Encryption – Sender *encrypts* the message by the *public key of the receiver*.

- Decryption – Only the intended receiver can *decrypt* the message by *his/her private key*.



Public Key Cryptography



- The sender and receiver cannot use the same set of keys for two way communication.

- Receiver needs only one private key to receive messages from anyone in the network.

- Sender needs ‘n’ public keys to communicate to the n entities in the network.

- Each user generates a pair of keys to be used for the encryption and decryption of messages.

- Each user is responsible for communicating the public key to the community.

- PT and CT will be numbers.
- Encryption and decryption are mathematical functions applied over the numbers representing PT and CT.
- The encryption function will be a trapdoor one way function which prevent eve dropping.

- One way function

- A function that satisfies the following properties.
- f is easy to compute , given x we can calculate y= f (x)
- Given y it is computationally infeasible to calculate x= f^{-1} (y)

- Trapdoor one way function is easy to calculate in one direction and infeasible to calculate in other direction if certain additional information are not known.

- $Y = f_k(X)$ easy , if k and X are known
- $X = f^{-1}_k(Y)$ infeasible ,if Y is known but k is unknown .
- $X = f^{-1}_k(Y)$ easy ,if k and Y are known

- Asymmetric cryptosystems are based on computationally hard problems.

Applications

- Encipherment : The sender encrypts a message with the receiver's public key.
- Digital signature: The sender 'signs' the message with its private key.
- Key exchange : Two sides cooperate to exchange a secret key.

Security

- Brute force attack is not possible since keys used are too large.
- It requires the use of **very large numbers.**
- Hence is **slow** compared to private key schemes.
- Asymmetric is used for the encryption of short messages.

The RSA Public Key System

- Most well-known and popular public key encryption.
- Introduced in 1978 by Rivest, Shamir, Adleman.
- The plaintext and cipher text are integers between 0 and n-1 for some n.
- A typical size for n is 309 decimal digits or 1024 bits.

- RSA is designed based on the fact that it is relatively easy to calculate the product of two prime numbers, but determining the original prime numbers, given the product, is hard.

- Underlying security of RSA is on the factorization problem.

- Encryption and decryption are done using commutative ring $R = < Z_n , + , \times >$.

- The ring is public since n is public.

- RSA uses a multiplicative group $G = < Z_{\Phi(n)}{}^{*} , \times)$ for key generation.

- The group is hidden since Φ (n) value is private.

RSA Key Generation

- Choose two large prime numbers, ***p*** and ***q*** such that $p \neq q$.
- Calculate their product, $\boldsymbol{n} = \boldsymbol{p} \times \boldsymbol{q}$.
- Compute $\boldsymbol{\varphi(n)} = (\boldsymbol{p} - 1) \times (\boldsymbol{q} - 1)$.
- Choose an integer, ***e***, $\boldsymbol{1 < e < \varphi(n)}$ such that ***e*** is relatively prime to $\boldsymbol{\varphi(n)}$.
- Calculate a value, ***d***, such that $\boldsymbol{e} \times \boldsymbol{d} \equiv 1 \bmod \boldsymbol{\varphi(n)}$.
- Public key $\rightarrow$ $\boldsymbol{(e,n)}$
- Private key $\rightarrow$ ***d***

RSA Algorithm

- Public key consists of the pair (***e***,***n***).
- A plaintext, P is encrypted to ciphertext, C as follows:

$$C = P^{\boldsymbol{e}} \bmod \boldsymbol{n}$$

- P ‘s size should be less than n.
- The private key is d.
- The plaintext is recovered from the ciphertext as follows:

$$P = C^{\boldsymbol{d}} \bmod \boldsymbol{n}$$

- The security of the system relies on the fact that it is almost impossible to calculate ***d*** if only (***e***,***n***) is known.

- p, q two prime numbers (private and chosen)
- n = p* q (public and calculated)
- e, with gcd (ϕ (n), e) =1; (public and chosen)
- e x d ≡ mod ϕ (n) (private and calculated)

- RSA uses modular exponentiation for encryption and decryption.

- To attack eve need to calculate $\sqrt[e]{C} \bmod n$

- Modular exponentiation using fast exponentiation is feasible in polynomial time.

- Modular logarithm is as hard as factoring the modulus n for which there is no polynomial time algorithm.

RSA Example

- Let, $p = 17$ and $q = 11$.
- So, $n = p \times q = 187$ and $\varphi(n) = (p - 1) \times (q - 1) = 160$.
- Choose ***e*** from Z_{160}^{*} , choose $e = 7$.
- Calculate d such that, $e \times d \equiv 1 \bmod \varphi(n)$.

How to calculate ***d***, given ***e*** and ***φ(n)***?

- Use Extended Euclidean algorithm to find the value of d.
- In this eg, φ(n) = 160 and e =7.

  Find gcd(φ(n),e) = gcd(160,7)

  160 = 22 x 7 + 6

  7 = 1 x 6 + 1

  6 = 6 x 1 + 0

  gcd(160,7) = 1

- Back trace the steps.

$1 = 7 - 1 \times 6$

$= 7 - 1 \times [160 - (22 \times 7)]$

$= -160 + 23 \times 7$

Here d = 23.

Encryption is as follows,

- Let P = 88 be the plaintext.

- Ciphertext, C = 88^7 mod 187

  = 11.

Decryption is as follow's

- Plaintext , P = 11^{23} mod 187

  = 88

Security of RSA

- Factorization attack

- The security of RSA is based on the fact that it’s infeasible to factor n in reasonable time.

- RSA is secure as long as an efficient algorithm for factorization has not been found.