Public Key Encryption

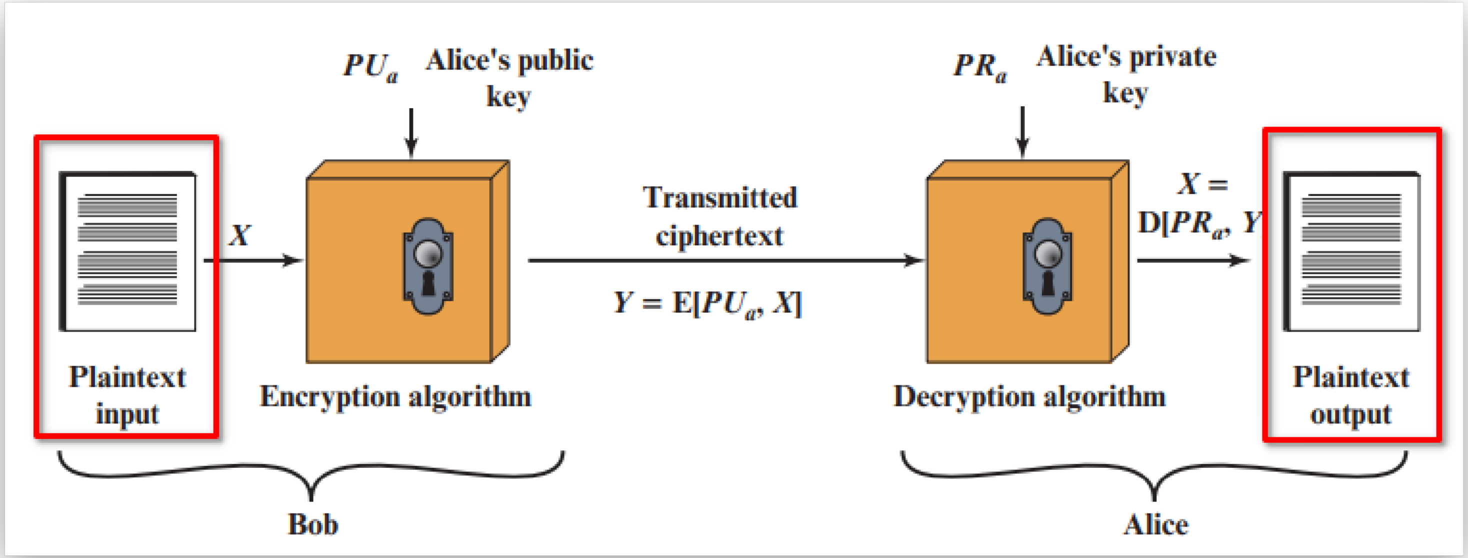
Jiuqin ZHOU

Outline

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

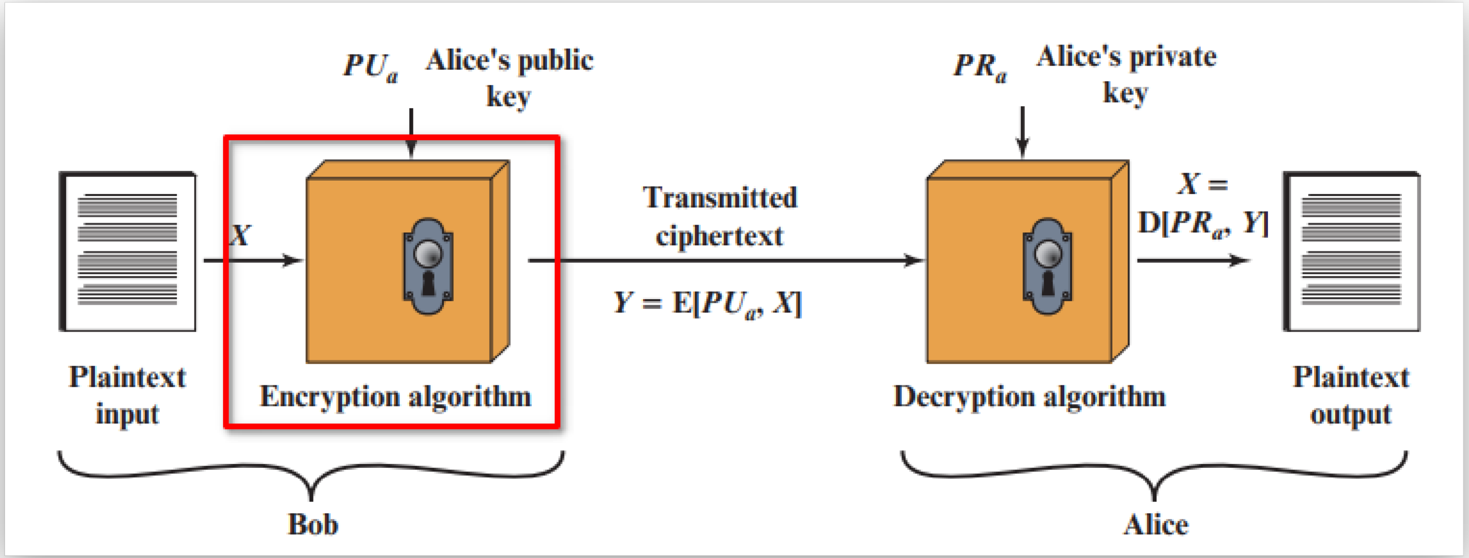


Plaintext: This is the readable message or data that is fed into the algorithm as input.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

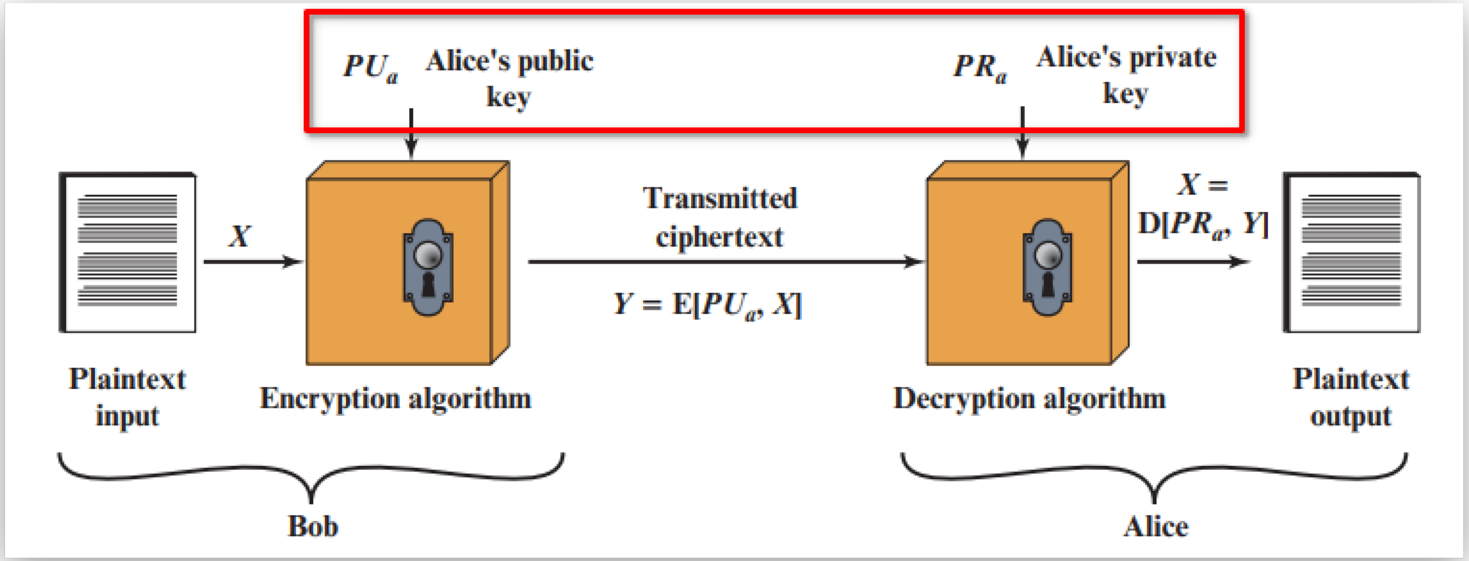


Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

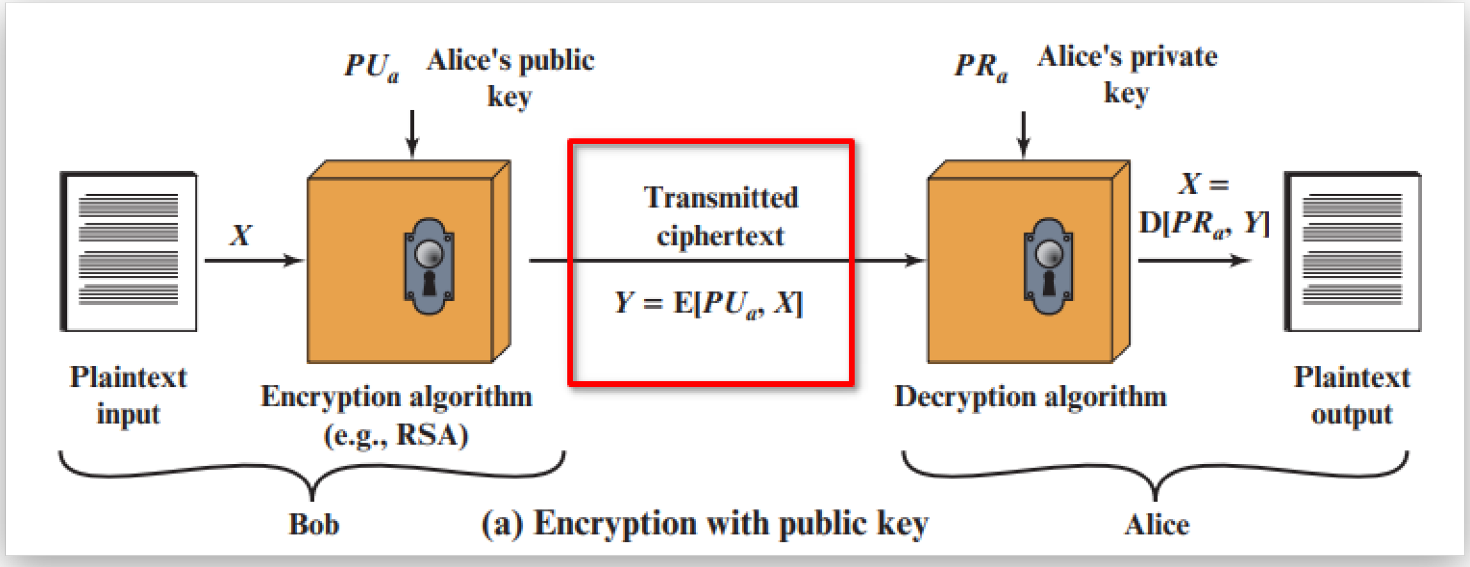


Public and Private Keys: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

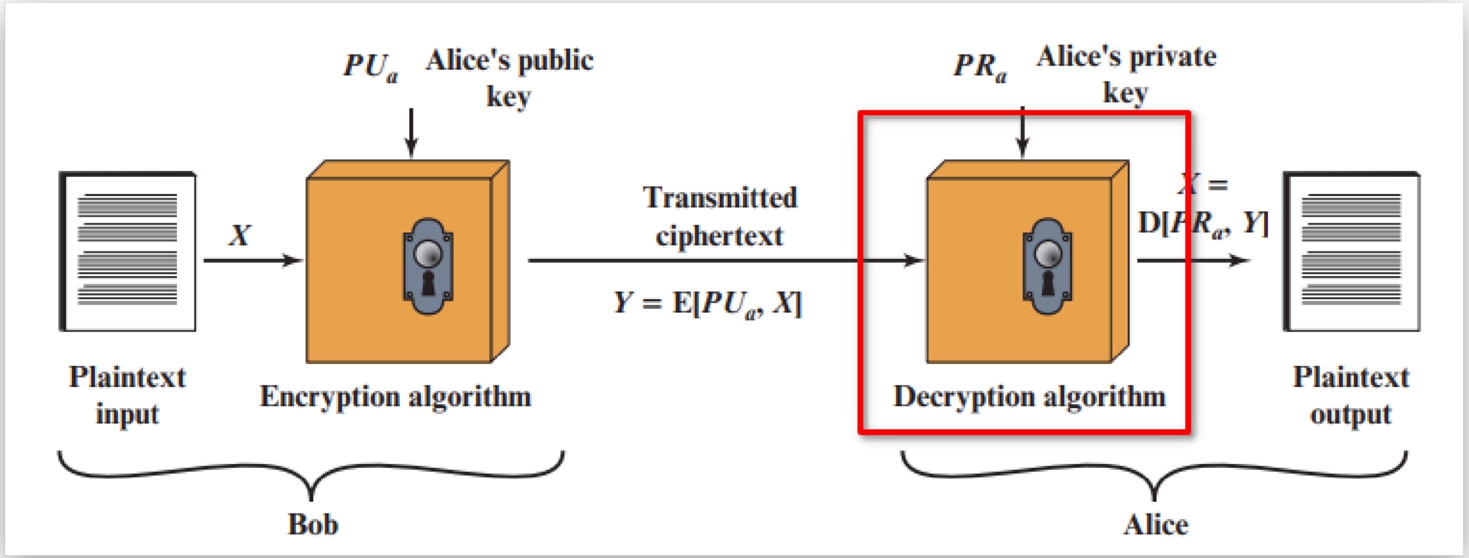


Ciphertext: This is the encrypted message produced as output. It depends on the plaintext and the key.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements



Decryption Algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

Computationally Easy

1. It is computationally easy for a party B to generate a key pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext: $C = E(PU_b, M)$.
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: $M = D(PR_b, C) = D(PR_b, E(PU_b, M))$.

Asymmetric Ciphers » Public-Key Cryptography

Simplified Model

Five Requirements

Computationally Infeasible

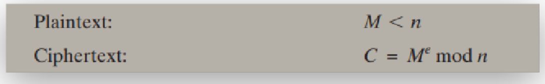
4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

Asymmetric Ciphers » The RSA Algorithm

Encryption & Decryption

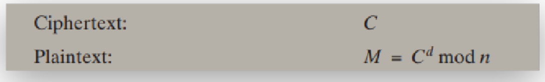
Fast Exponentiation Algorithm

Extended Euclidean Algorithm



Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

RSA Encryption



Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

RSA Decryption

- Suppose that user A has published its public key $\{e, n\}$ and that user B wishes to send the message M to A. Then B calculates $C = M^e \mod n$ and transmits C.
- On receipt of this ciphertext, user A decrypts by calculating $M = C^d \mod n$ using its private key $\{d, n\}$.

Asymmetric Ciphers » The RSA Algorithm

Important Observations

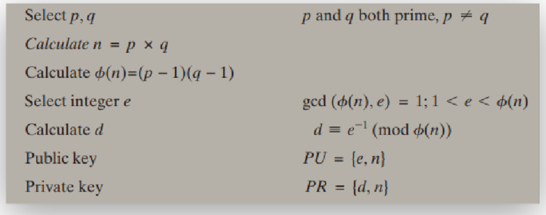
- $M^{ed} = M \mod n$ holds if $ed = 1 \mod \phi(n)$.
- If $n = p * q$, where p and q are two prime numbers, then $\phi(n) = (p-1)(q-1)$.

Asymmetric Ciphers » The RSA Algorithm

Encryption & Decryption

Fast Exponentiation Algorithm

Extended Euclidean Algorithm



Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n)=(p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

RSA Key Generation

1. Select two prime number p and q, and calculate $n = p * q$ and $\phi(n) = (p-1)(q-1)$.
2. Select an integer e that is relatively prime to $\phi(n)$, and calculate its reverse element d such that $ed = 1 \mod \phi(n)$.
3. We get the public key $\{d, n\}$ and private key $\{e, n\}$.

Asymmetric Ciphers » The RSA Algorithm

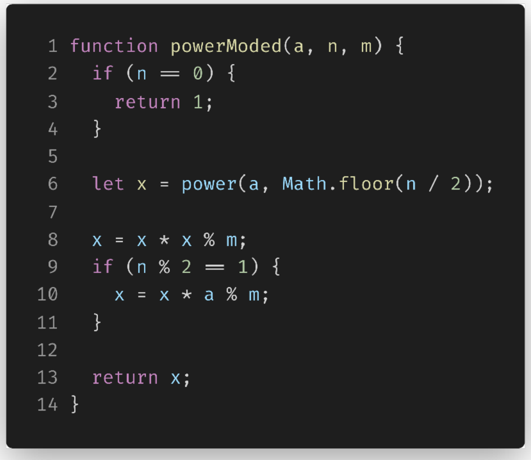
- The speed of the RSA encryption & decryption bounds to the speed of calculating **exponentiation**.
- Let's form the problem in a formal way: Given integers a, n, and m with $n \geq 0$ and $0 \leq a < m$, compute $a^n \mod m$.

Asymmetric Ciphers » The RSA Algorithm

Encryption & Decryption

Fast Exponentiation Algorithm

Extended Euclidean Algorithm



```javascript
function powerModed(a, n, m) {
  if (n == 0) {
    return 1;
  }

  let x = power(a, Math.floor(n / 2));

  x = x * x % m;
  if (n % 2 == 1) {
    x = x * a % m;
  }

  return x;
}
```

(2) **In Javascript**

Simple but not Efficient

$$a^n = \Pi_1^n a \qquad (1)$$

Fast Exponentiation Algorithm

$$a^n = \begin{cases} 1 & \text{n} = 0 \\ (a^{\lfloor n/2 \rfloor})^2 & \text{n is even} \\ (a^{\lfloor n/2 \rfloor})^2 * a & \text{n is odd} \end{cases} \qquad (2)$$

Asymmetric Ciphers » The RSA Algorithm

- From the key generation process, there is one step seems too unclear to us: How to calculate the d, which is the reverse of e under the field $\mod \phi(n)$? By the definition of module, $ed = 1 \mod \phi(n)$ can be transformed into $ed + k\phi(n) = 1$, and we are interested to find a pair (d, k).
- Let's form the problem in a formal way: Given integers a, b, how to find two additional integers x, y, such that $ax + by = gcd(a, b)$.

Asymmetric Ciphers » The RSA Algorithm

Encryption & Decryption

Fast Exponentiation Algorithm

Extended Euclidean Algorithm

Basic Euclidean Equation

$$\gcd(a, b) = \gcd(b\%a, a) \tag{1}$$

Other Important Equations

$$ax + by = \gcd(a, b) \tag{2}$$

$$(b\%a)x_1 + ay_1 = \gcd(b\%a, a) \tag{3}$$

$$b\%a = b - \lfloor b/a \rfloor * a \tag{4}$$

Combine (1), (4), **and** (3)

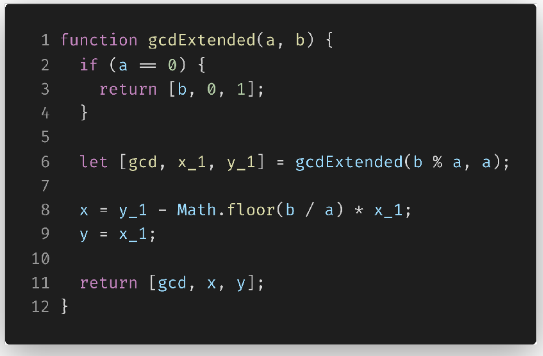
$$a(y_1 - \lfloor b/a \rfloor x_1) + bx_1 = gcd(a, b) \tag{5}$$

Asymmetric Ciphers » The RSA Algorithm

Encryption & Decryption

Fast Exponentiation Algorithm

Extended Euclidean Algorithm



```
function gcdExtended(a, b) {
  if (a == 0) {
    return [b, 0, 1];
  }

  let [gcd, x_1, y_1] = gcdExtended(b % a, a);

  x = y_1 - Math.floor(b / a) * x_1;
  y = x_1;

  return [gcd, x, y];
}
```

(6), (7) in Javascript

Compare (2) and (5)

$$\begin{aligned} x &= y_1 - \lfloor b/a \rfloor x_1 \\ y &= x_1 \end{aligned} \tag{6}$$

The Ending Case

$$\begin{aligned} x &= 0 \\ y &= 1 \qquad \text{if b\%a} == 0 \\ gcd(a, b) &= b \end{aligned} \tag{7}$$

Asymmetric Ciphers » Other Public-Key Crypto-systems

Diffie-Hellman Key Exchange

Elgamal Cryptographic System

- For this scheme, there are two publicly known numbers: a prime number q and an integer a that is a primitive root of q.
- Alice selects a random integer $X_A < q$ and computes $Y_A = a^{X_A} \mod q$. Similarly, Bob independently selects a random integer $X_B < q$ and computes $Y_B = a^{X_B} \mod q$.
- Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, X_A is Alice's private key and Y_A is Alice's corresponding public key, and similarly for Bob.
- Alice computes the key as $K = (Y_B)^{X_A} \mod q$ and Bob computes the key as $K = (Y_A)^{X_B} \mod q$. These two calculations produce identical results.

Asymmetric Ciphers » Other Public-Key Crypto-systems

Diffie-Hellman Key Exchange

Elgamal Cryptographic System

- The global elements of ElGamal are a prime number q and a, which is a primitive root of q.
- User A generates a private/public key pair as follows: Generate a random integer X_A, such that $1 < X_A \leq q-1$. Compute $Y_A = a^{X_A} \mod q$. A’s private key is X_A and A’s public key is $\{q, a, Y_A\}$.

Asymmetric Ciphers » Other Public-Key Crypto-systems

Diffie-Hellman Key Exchange

Elgamal Cryptographic System

- Choose a random integer k such that $1 \leq k \leq q-1$. Compute a one-time key $K = (Y_A)^k \mod q$. Encrypt the message M as the pair of integers (C_1, C_2) where $C_1 = a^k \mod q$; $C_2 = KM \mod q$.
- User A recovers the plaintext as follows: Recover the key by computing $K = (C_1)^{X_A} \mod q$. Compute $M = (C_2 K^{-1}) \mod q$.