Week 8: Modern Asymmetric Ciphers

PhD. Ngoc-Tu Nguyen

tunn@uit.edu.vn

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
- Logarithm Based Cryptography (P2)
  - ElGamal cipher;
  - Diffie-Hellman key exchange;
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

Why Public-Key Cryptosystems?

To overcome two of the most difficult problems associated with symmetric encryption:

- **Key distribution (key for sysmetric encryption)**
  - How to have secure communications in general without having to trust a KDC with your key
- **Digital signatures**
  - How to verify that a message comes intact from the claimed sender

Whitfield Diffie and Martin Hellman: proposed a method that addressed both problems (1976)

Moden Asymmetric ciphers

Symmetric cipher vs Asymmetric cipher

Hommophic, Searchable encryption,..

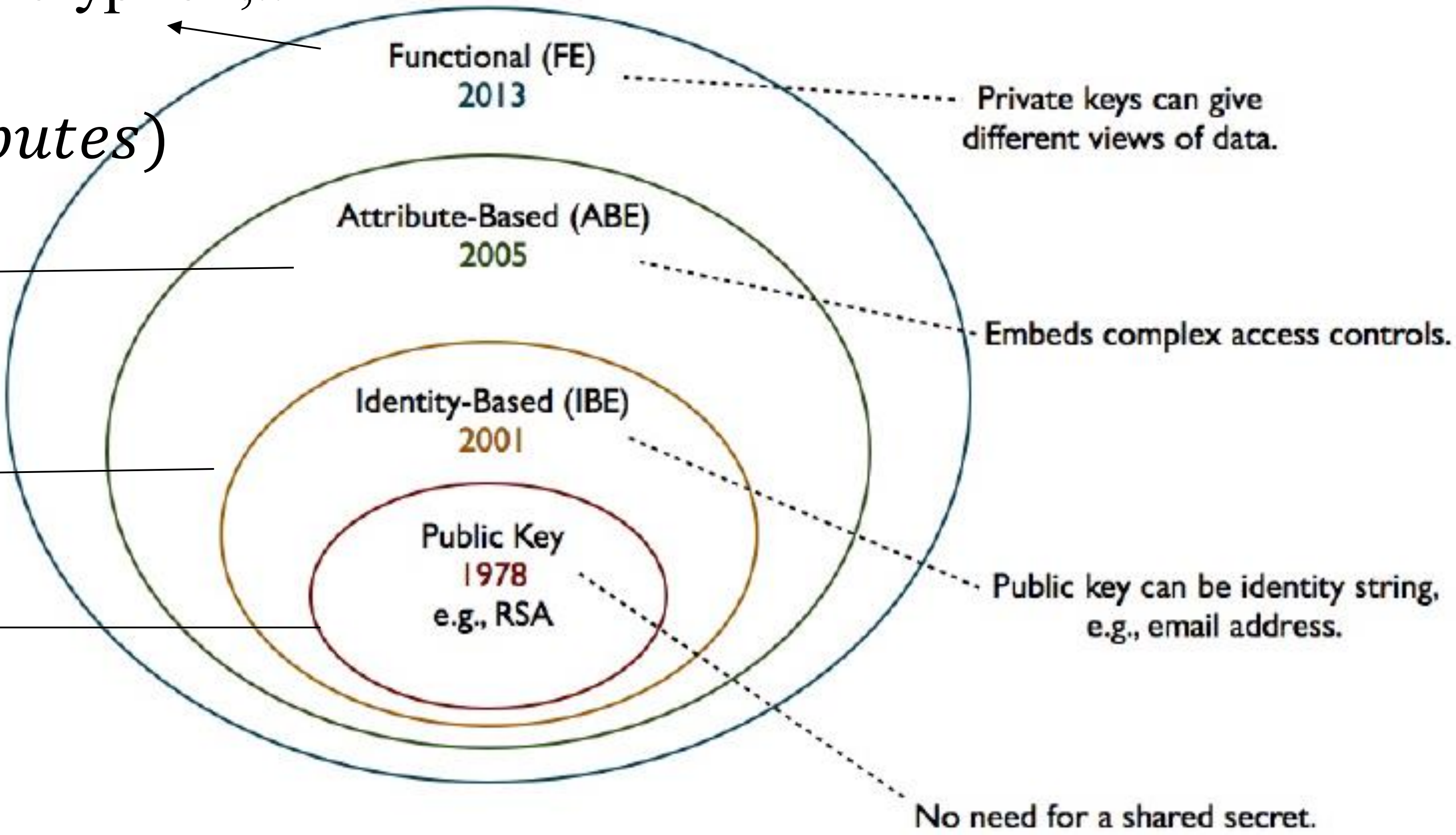
$ID_A(atributes)$ $ID_B(atributes)$

$(PK_A, \{SK_A, SK_B, ..\})$

ID

(PK_A, SK_A)

(PK_A, SK_A)



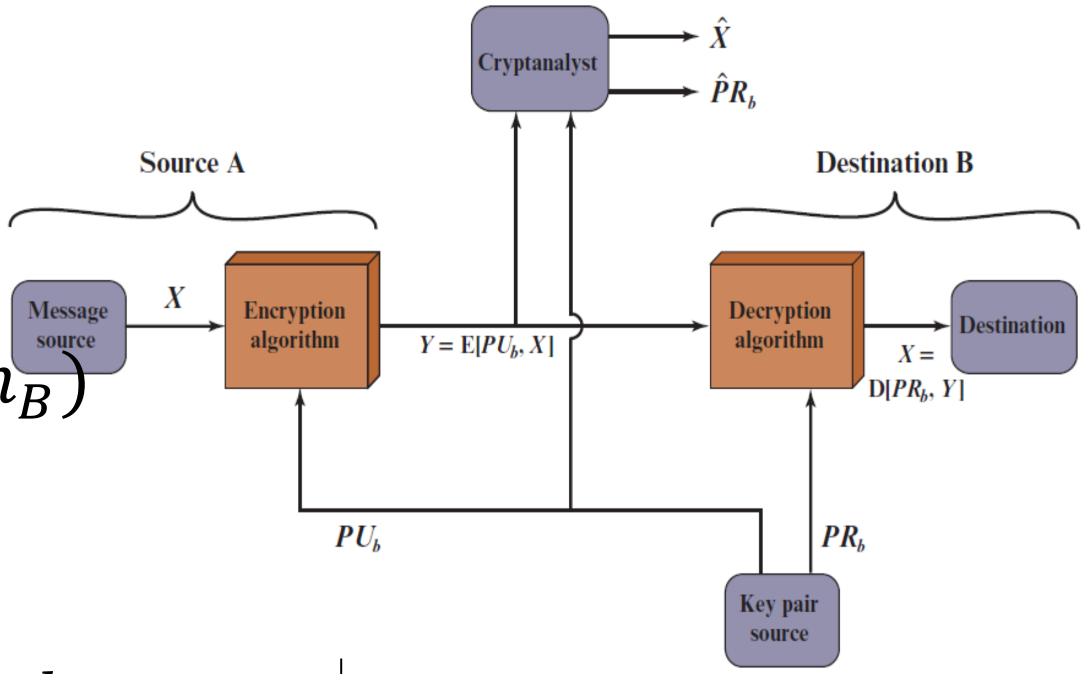
RSA: Confidentiality

A d_A / n_A, e_A

$e_A . d_A = 1 \bmod \phi(n_A)$

B d_B / n_B, e_B

$e_B . d_B = 1 \bmod \phi(n_B)$



$$C = m^{e_B} \bmod n_B$$

$$C^{d_B} \bmod n_B = (m^{e_B})^{d_B} \bmod n_B$$
$$= m$$

- **Protect secret key?**
- **Distribute public keys?**

RSA: Authentication

A **B**

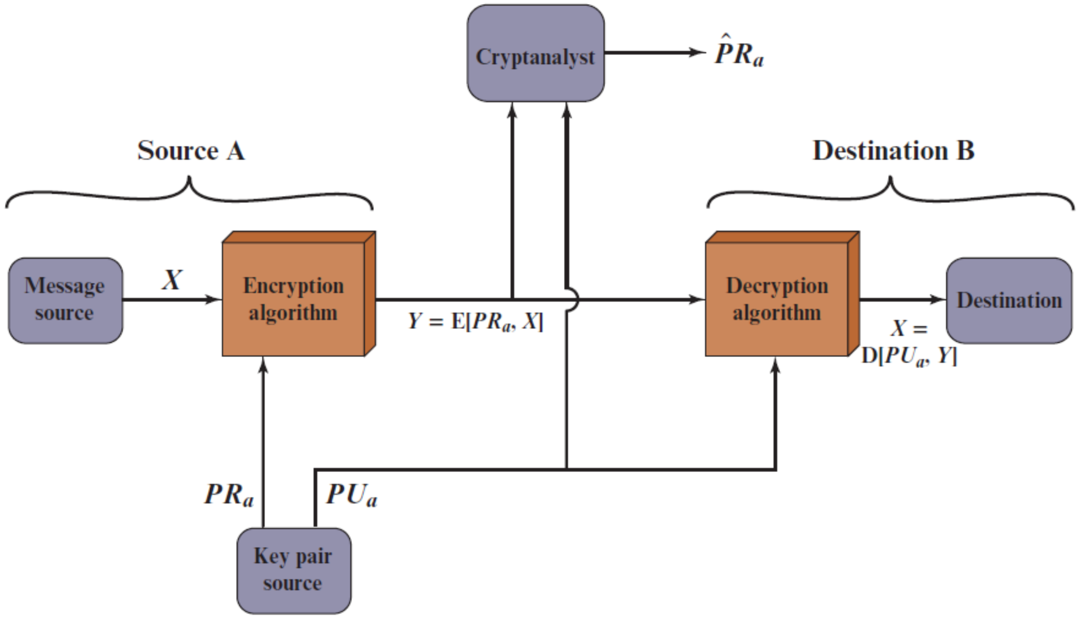
d_A / n_A, e_A d_B / n_B, e_B

$S = m^{d_A} \bmod n_A$ (m, S)

Verify message ***m***

$$S^{e_A} \bmod n_A = (m^{d_A})^{e_A} \bmod n_A$$
$$= m'$$
$$? = m$$

- **Sent by A**
- **Original (integrity)**



- **Protect secret key?**
- **Distribute public keys?**

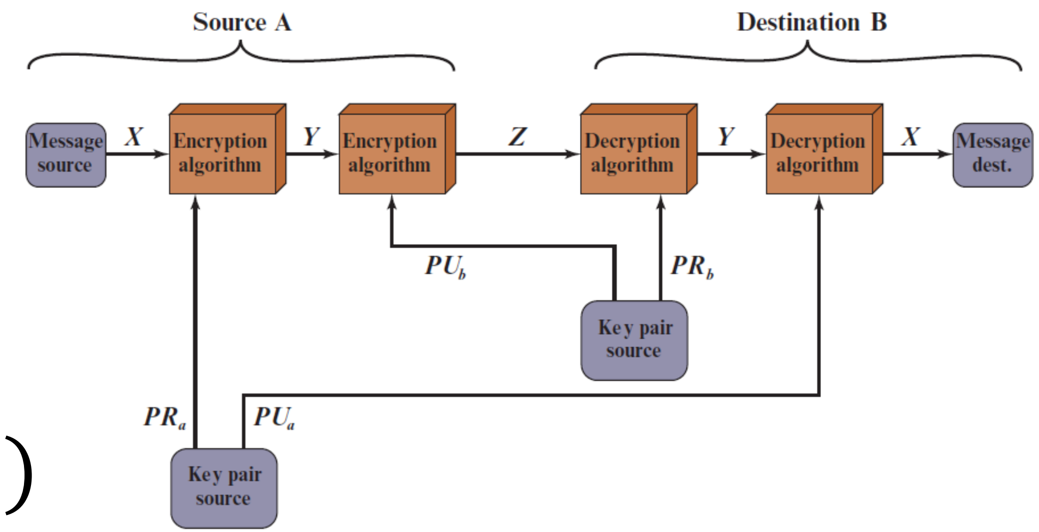
RSA: Authentication and Secrecy

A **B**

d_A / n_A, e_A d_B / n_B, e_B

$$S = k^{d_A} \, mod \, n_A$$

$$(C_1 = k^{e_B}, S_1 = S^{e_B})$$



Limitation?

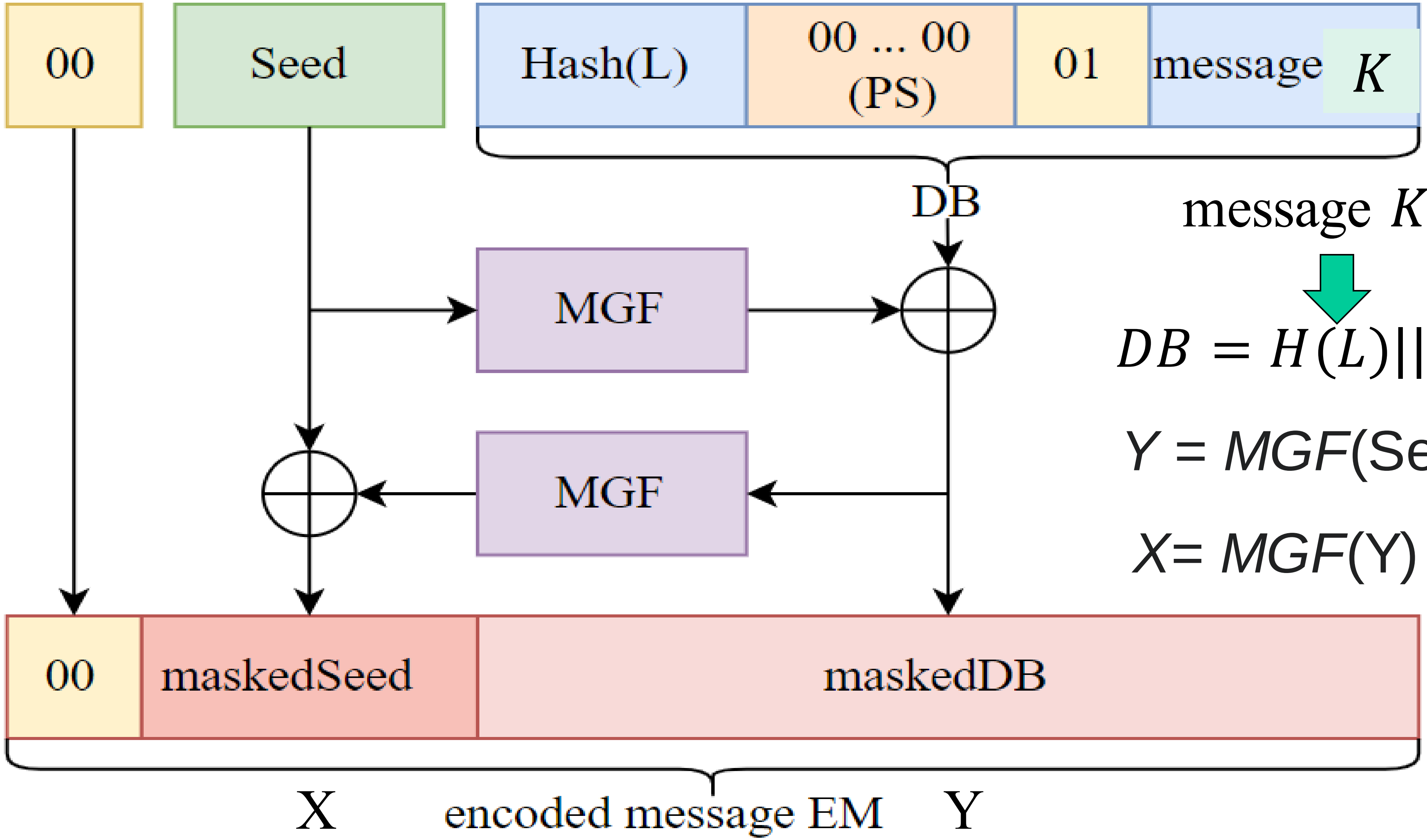
Decrypt and verify the secret key k

$$C_1^{d_B} \, mod \, n_B = (k^{e_B})^{n_B} \, mod \, n_B = k;$$

$$S_1^{d_B} \, mod \, n_B = (S^{e_B})^{n_B} \, mod \, n_B = S;$$

$$S^{e_A} \, mod \, n_A = (k^{d_A})^{e_A} \, mod \, n_A = k' =? \, k$$

Encryption Using Optimal Asymmetric Encryption Padding (OAEP)



MGF: a hash function

message K

$$DB = H(L)||0000..||01||K$$

$Y = MGF(\text{Seed}) \oplus DB$

$X = MGF(Y) \oplus \text{Seed}$

?

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1,2)
- Logarithm Based Cryptography (P3)
  - ElGamal cipher;
  - Diffie-Hellman key exchange;
- Elliptic Curve Cryptography (P4)
- Some advanced cryptography system (quantum resistance)

Discrete Logarithm problem

Finitemultiplicative group $(G,.) =< g > = \{g^n : n \in \mathbb{Z}\}$

Example: $G = \mathbb{Z}_p \backslash \{0\} = \{1,2,\ldots,p-1\} =< g >$

$$g, n \xrightarrow{\text{Easy to compute}} y_0 = g^n \bmod p$$

$$g^n = y_0 \bmod p \xleftarrow{\text{Hard to solve } n} g, y_0, p$$

$\boxed{\textbf{Hard to solve equation } g^x = a \bmod p \textbf{ in finite field!}}$

Outline

- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
  - RSA signature;
- Logarithm Based Cryptography (P2)
  - ElGamal cipher
  - Diffie-Hellman key exchange;
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

ElGamal cipher

ElGamal parameters

Large prime number: p

Multiplicative group

$$G = \langle g \rangle = \mathbb{Z}_p \backslash \{0\} = \{1,2, \dots, p-1\}$$

Key generation (

Secret key: $x \in_R [1, p-1]$

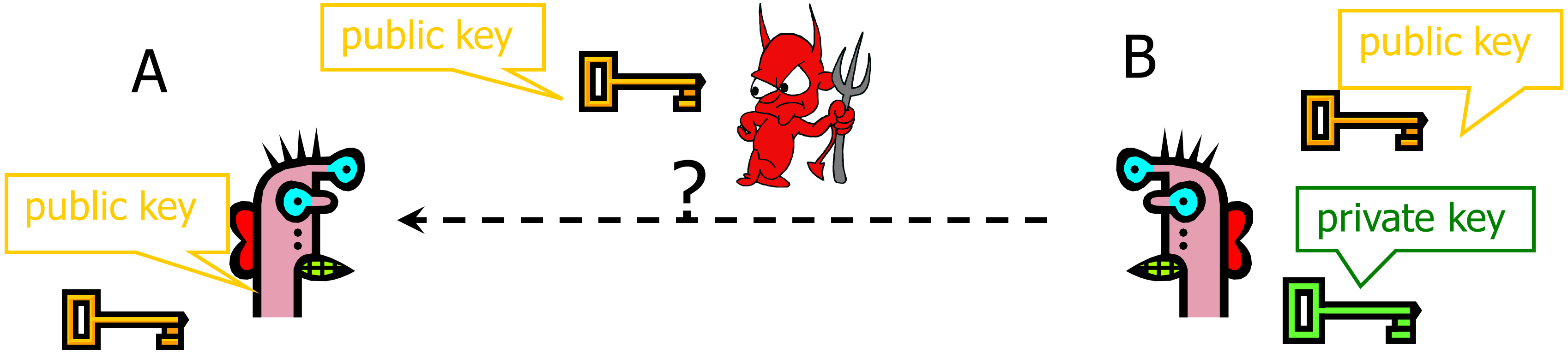
Public key: $h = g^x \bmod p \in \mathbb{Z}_p$

ElGamal cipher

- **Encryption message $\boldsymbol{m < p - 1}$ (using public key $\boldsymbol{h = g^{x}}$)**
  - Choose a random number: $r \in_R [1, p-1]$
  - Compute $C_1 = g^r \bmod p$;
  - Computer $C_2 = m.h^r \ mod\ p$
  - Output cipher message (C_1, C_2)
- **Decryption** (C_1, C_2) **(using secret key $\boldsymbol{x}$)**
  - Compute $(C_1)^x mod\ p = g^{r.x} \bmod p$;
  - Computer $\dfrac{C_2}{(C_1)^x}\ mod\ p = \dfrac{m.g^{x.r}}{g^{r.x}}\ mod\ p$

    $$= m$$
  - Output message m

ElGamal cipher



$PU_B = h_B, p$

$(PR_B = x_B, PU_B = h_B = g^{x_B}, p)$

Input: $M < p$

Select a random number: $r < p - 1$

Compute: $C_1 = g^r \bmod p$

$C_2 = m.h_B^r \bmod p$

$\xrightarrow{(C_1, C_2)}$

Compute:

$$\frac{C_2}{(C_1)^{x_B}} \bmod p$$

$$= \frac{m.g^{x_B.r}}{g^{r.x_B}} \bmod p = m$$

Outline

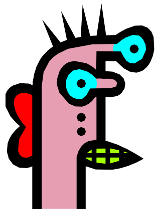
- Why asymmetric cryptography?
- Factoring Based Cryptography (P1)
  - RSA signature;
- Logarithm Based Cryptography (P2)
  - ElGamal cipher
  - Diffie-Hellman key exchange;
- Elliptic Curve Cryptography (P3)
- Some advanced cryptography system (quantum resistance)

Diffie-Hellman key exchange

- A and B never met and share no secrets;
- Public info: the prime number p and g
  - p is a large prime number, g is a generator of Z_p^*
    - $Z_p^* = \{1, 2 \ldots p\text{-}1: \forall a \in Z_p^* \; \exists i$ such that $a = g^i \bmod p\}$

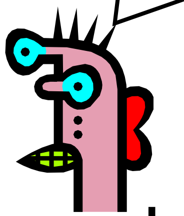
Pick secret, random x

Pick secret, random y

A 

$$X = g^x \bmod p \longrightarrow$$

$$\longleftarrow Y = g^y \bmod p$$

 B

Compute

$$k_A = Y^x \bmod p = (g^y)^x \bmod p$$
$$= g^{y.x} \bmod p$$

Compute

$$k_B = X^y = (g^x)^y \bmod p$$
$$= g^{xy} \bmod p$$

Session key $K = k_A = k_B = g^{x.y}$ ➡ Symmetric key

Diffie-Hellman exchange Protocol (DHE)

$p = 1606938044258990275541962092341162602522202993782792835301301$

$g = 123456789$

g^a mod p =
78467374529422653579754596319852702575499692980085777948593

g^b mod p =
560048104293218128667441021342483133802626271394299410128798

$a =$
685408003627063
761059275919665
781694368639459
527871881531452

$b =$
362059131912941
987637880257325
269696682836735
524942246807440

$(g^b)^a \, mod \, p$

g^{ab} mod p =
437452857085801785219961443000
845969831329749878767465041215

$(g^a)^b mod \, p$

Why Is Diffie-Hellman Secure?

- Discrete Logarithm (DL) problem:

  given g^x mod p, it's hard to extract x
  - There is no known efficient algorithm for doing this
  - This is not enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem:

  given g^x and g^y, it's hard to compute g^{xy} mod p
  - … unless you know x or y, in which case it's easy
- Decisional Diffie-Hellman (DDH) problem:

  given g^x and g^y, it's hard to tell the difference between g^{xy} mod p and g^r mod p where r is random

Properties of Diffie-Hellman

- Assuming DDH problem is hard, Diffie-Hellman protocol is a secure key establishment protocol against <u>passive</u> attackers
  - Eavesdropper can't tell the difference between the established key and a random value
  - Can use the new key for symmetric cryptography
- Basic Diffie-Hellman protocol does not provide authentication
  - IPsec combines Diffie-Hellman with signatures, anti-DoS cookies, etc.

Man-in-the middle attacks the DHE

(a, g^a)

$$g^a \bmod p \longrightarrow$$

$$\longleftarrow g^b \bmod p$$

(b, g^b)

A(Putin)

$$sk = (g^b)^a = (g^a)^b = g^{ab} \bmod p$$

B()

man-in-the-middle attack!

$$g^a \bmod p \longrightarrow$$

$$\longleftarrow g^m \bmod p$$

$$g^m \bmod p \longrightarrow$$

$$\longleftarrow g^b \bmod p$$

$$sk_1 = g^{am} \bmod p$$

$$sk_2 = g^{bm} \bmod p$$

Advantages of Pblic-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - Can use this for key establishment, avoiding the “chicken-or-egg” problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
- Authentication without shared secrets
- Encryption keys are public, but must be sure that Alice’s public key is really <u>her</u> public key
  - This is a hard problem… Often solved using public-key certificates

Disadvantages of Public-Key Crypto

- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - SSL, IPsec, most other systems based on public crypto
- Keys are longer
  - 3072 bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - Factoring, RSA problem, discrete logarithm problem, decisional Diffie-Hellman problem…