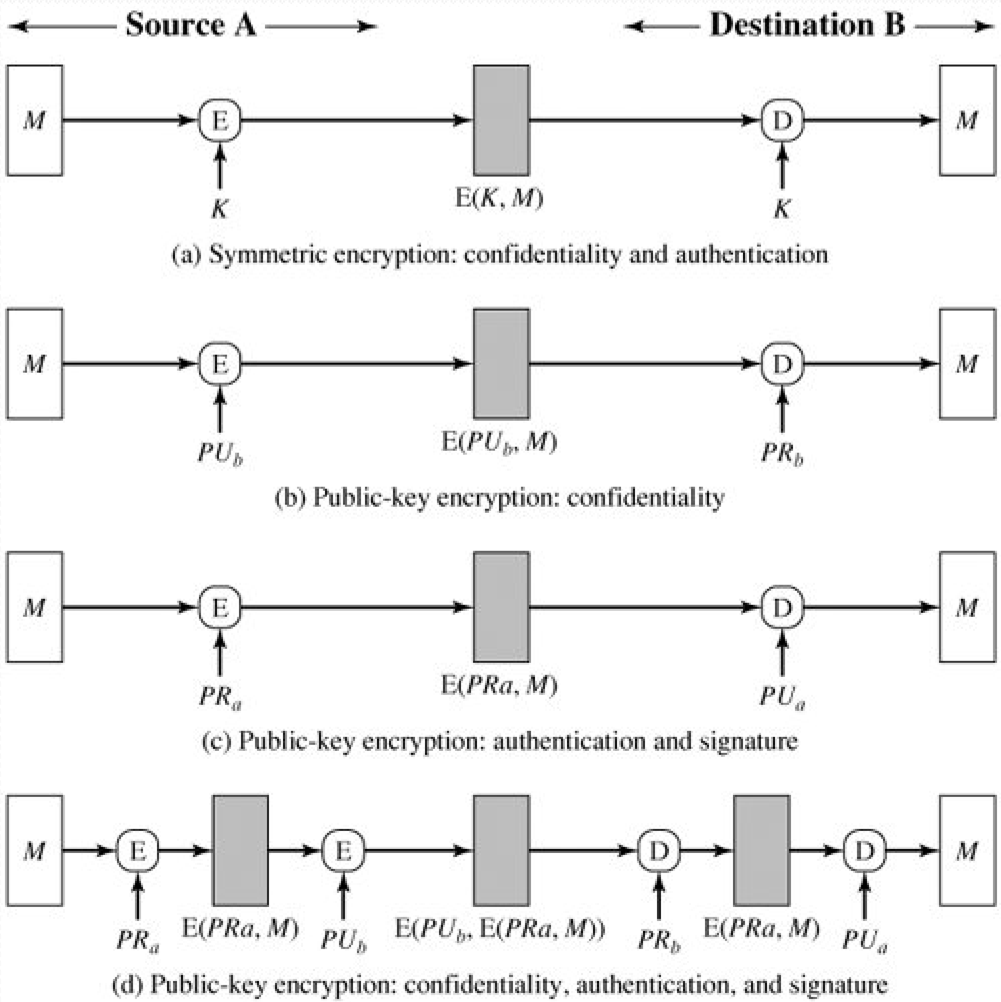
CS ??? Computer Security

Public Key Cryptography

Yasser F. O. Mohammad

REMINDER 1: Different Uses of Encryption



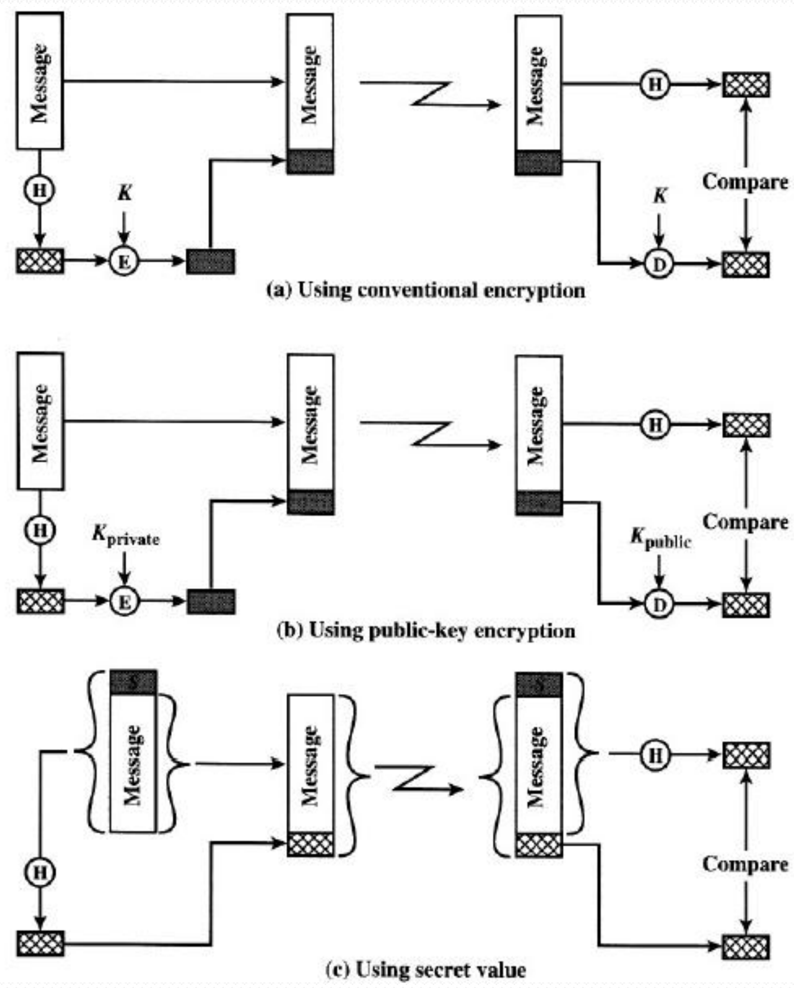
REMINDER: One Way Hash Functions

a) Only we know *k*
- *Most conventional*

b) Uses Public Keys only
- *Offers Nonrepudiation*
- *No key distribution*

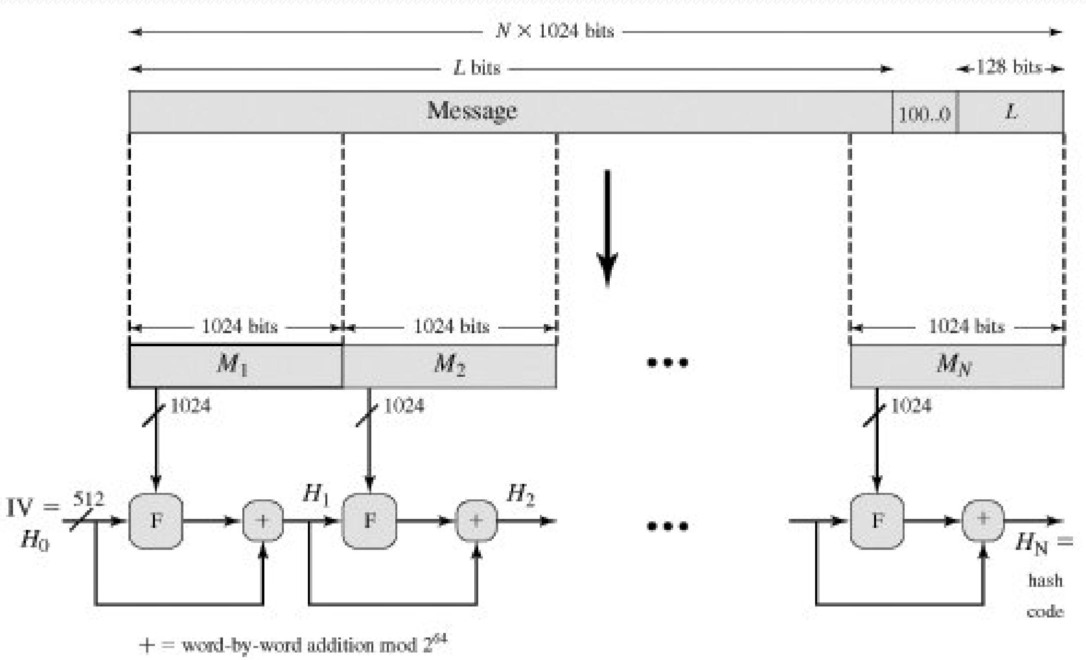
c) Only we know the secret
- *No encryption*
- *Used in HMAC adopted by IP security*

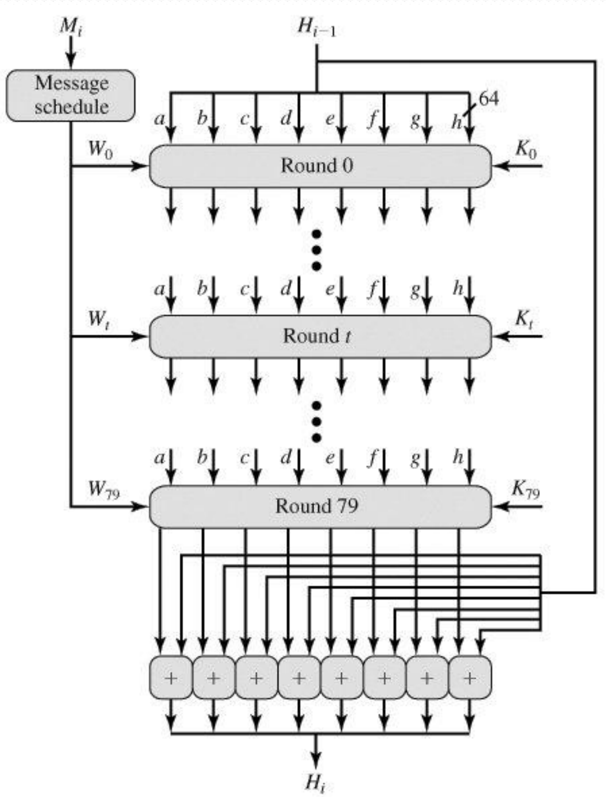
- *Why No Encryption?*
1. *Encryption is slow*
2. *Encryption is expensive*
3. *Encryption is optimized for large*
4. *Patents & export control*



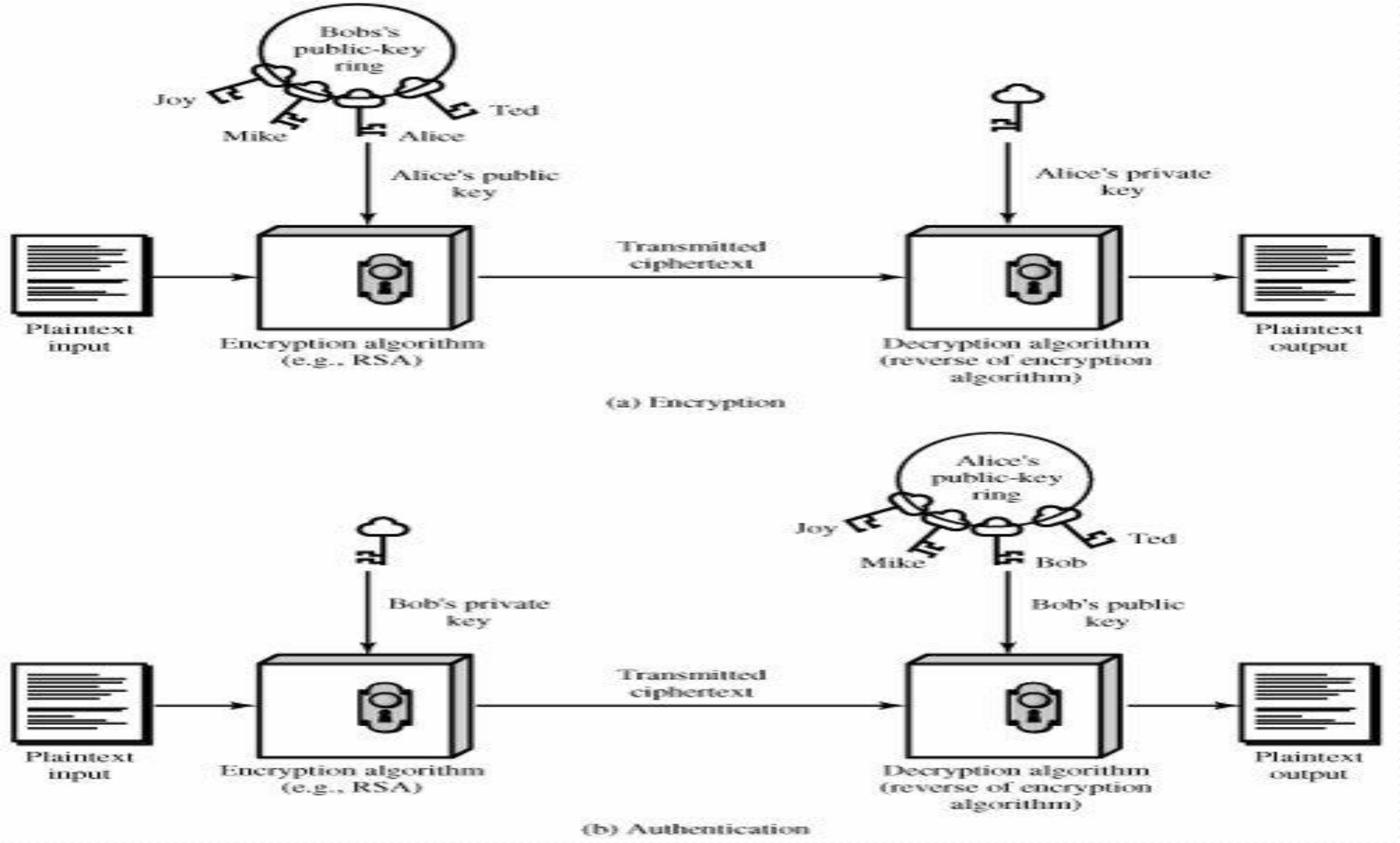
REMINDER 3: Modern Hash Functions

- SHA-1 (self read the algorithm)
  - Maximum input is 2^{64}
  - Digest size = 160 bits
  - Block size is 512 or 1024 bits





Public Key Encryption



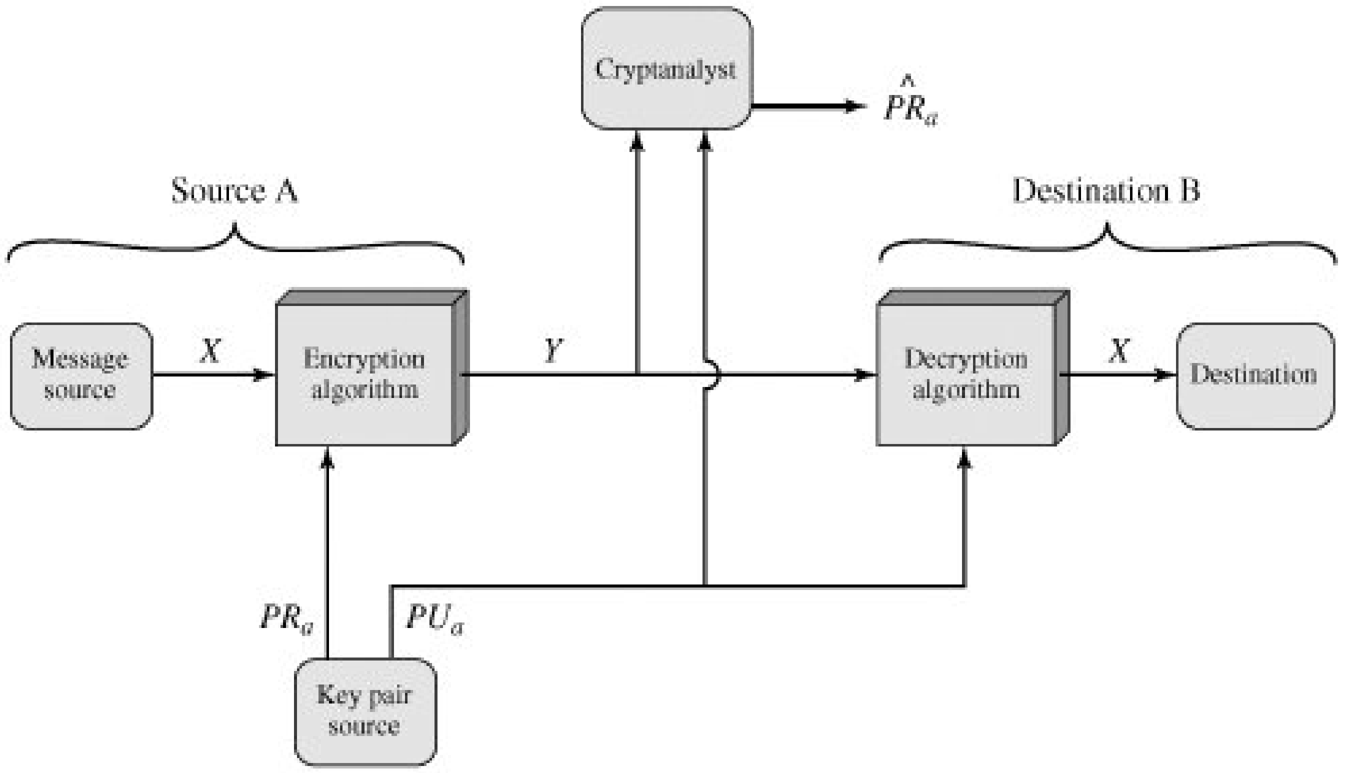
Public vs. Shared Key

Conventional Encryption	Public-Key Encryption
Needed to Work: **1.** The same algorithm with the same key is used for encryption and decryption. **2.** The sender and receiver must share the algorithm and the key.	**Needed to Work:** **1.** One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. **2.** The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: **1.** The key must be kept secret. **2.** It must be impossible or at least impractical to decipher a message if no other information is available. **3.** Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	**Needed for Security:** **1.** One of the two keys must be kept secret. **2.** It must be impossible or at least impractical to decipher a message if no other information is available. **3.** Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

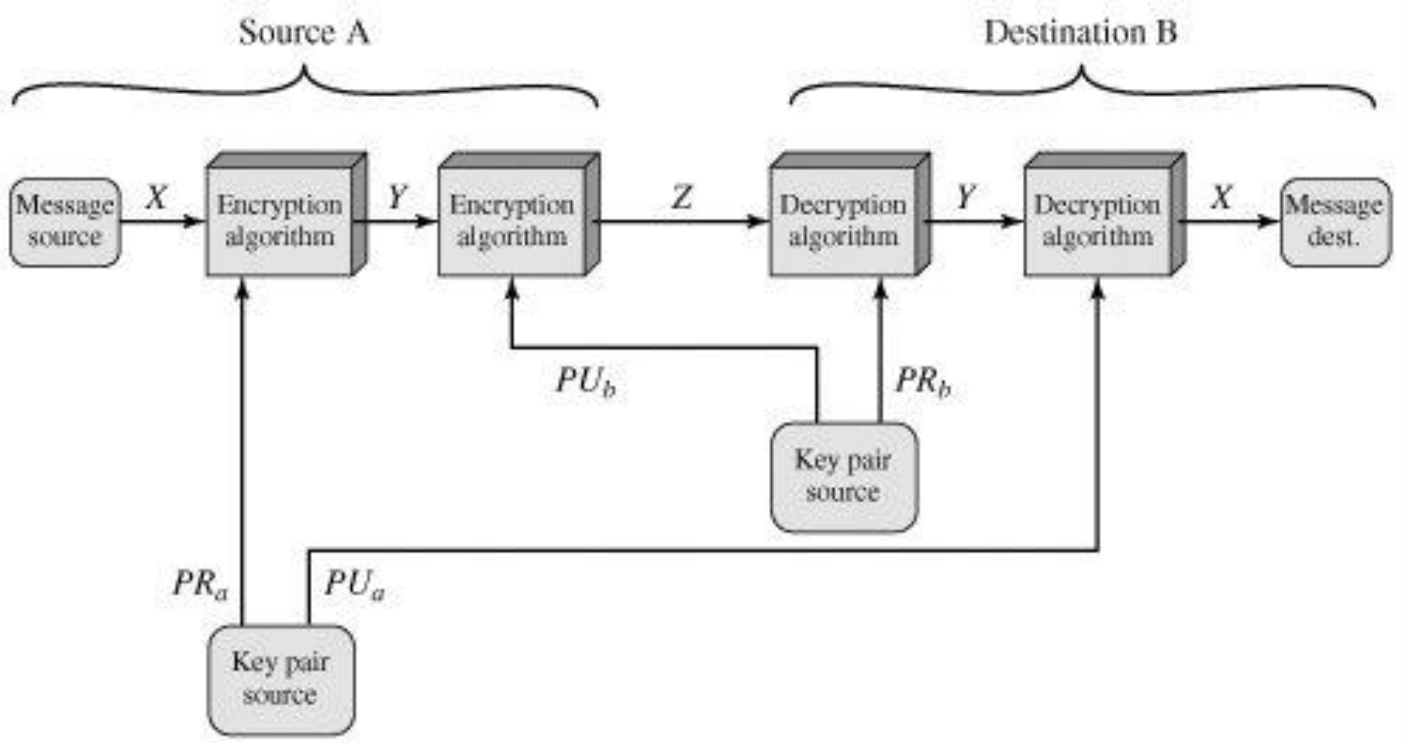
Uses of Public Key Encryption

- Encryption/Decryption
- Digital Signature
- Shared-Key Exchange

Public Key for Authentication



Public Key for Confident. + Auth.



Applications of Public Key Systems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements of Public Key Systems

- Easy to generate key pairs
- Easy to encrypt and decrypt
- Knowing the public key we cannot guess the private key
- Knowing a cipher and the public key we cannot get the plain text

- [Optional] the two keys can be applied in either order

RSA

- Developed in 1977
- By
  - Ron Rivest
  - Adi Shamir
  - Len Adelman
- Plain and ciphertexts are numbers between 0 and $2^n - 1$ (usually n=1024)
- General Purpose Public Key system
- Depends on the difficulty to factorize large numbers

RSA Algorithms

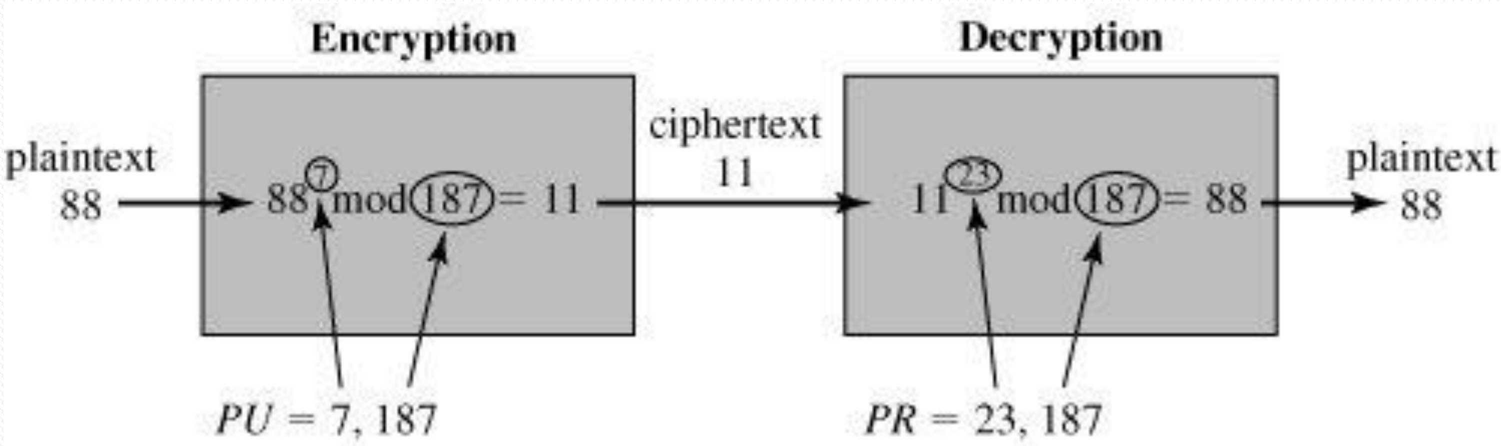
Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

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

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

Example

- p=17, q=11
- n=pq=187
- $\Phi(n)=(p-1)*(q-1)=160$
- e is prime less than $\Phi(n)$ and $GCD(\Phi(n),e)=1$ (e.g. 7)
- $d=e^{-1} \bmod \Phi(n)=23$ $(23*7=161)$



How to Break RSA?

1. Factorize n = Find p and q.
2. Find $\Phi(n)=(p-1)*(q-1)$
3. Find $d=e^{-1} \bmod \Phi(n)$

Now you have the private key!!!!

The only problem is that it is mathematically very difficult to factorize n.

Diffie-Hellman

- Published by Diffie and Hellman in 1976
- First Public Key algorithm
- Can be used only for key exchange
- Depends on the difficulty to calculate discrete logarithms

What is a discrete logarithm?

- a is a primitive root of a prime number p iff its powers generate all numbers from *1* to *p-1*.
- $a \bmod p, a^2 \bmod p, a^3 \bmod p, \ldots\ldots, a^{p-1} \bmod p$ equal *1, 2, ….,* *p-1* in some permutation.

- For every integer $b < p$ and a primitive root a of the prim p there exist a unique number i where:

  $b = a^i \bmod p$ where $0 \le i \le (p-1)$

- Discrete logarithm $dlog_{a,p}(b)=i$ where $b = a^i \bmod p$

- This is difficult and slow!!

Diffie-Hellman

- The point is that users A and B will be able to calculate the secret key using only:
  1. His private key
  2. Other's public key

- Eve needs to do a discrete logarithm because she does not have any of the private keys.

Global Public Elements

q	prime number
α	$\alpha < q$ *and* α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Numeric example

- $q=71$
- $\alpha=7$
- $X_A=5$
- $X_B=12$
- $Y_A=7^5 mod71=51$
- $Y_B=7^{12} mod71=4$

- $K=4^5 mod71=51^{12} mod71=30$

Why it works?

$K = (Y_B)^{X_A} \bmod q$

$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$

$= (\alpha^{X_B})^{X_A} \bmod q$ by the rules of modular arithmetic

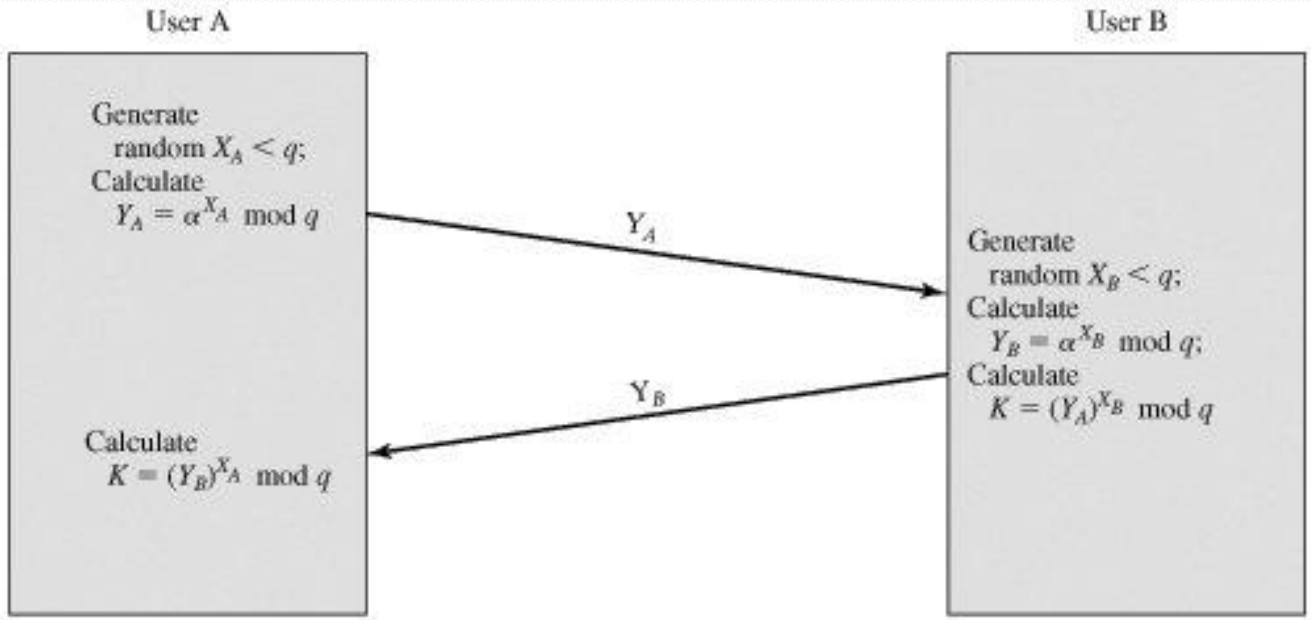
$= (\alpha^{X_B X_A} \bmod q$

$= (\alpha^{X_A})^{X_B} \bmod q$

$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$

$= (Y_A)^{X_B} \bmod q$

Key exchange using Diffie-Hellman



- Can be broken using Man-in-the-Middle Attack

Man-in-the-Middle Attack

$$A \to E : Y_A$$

$$E \to B : Y_{D1}$$

$$\begin{cases} E : K_2 = {Y_A}^{X_{D2}} \bmod q \\ B : K_1 = {Y_{D1}}^{X_B} \bmod q \end{cases}$$

$$B \to E : Y_B$$

$$E \to A : Y_{D2}$$

$$\begin{cases} E : K_1 = {Y_B}^{X_{D1}} \bmod q \\ A : K_2 = {Y_{D2}}^{X_A} \bmod q \end{cases}$$

Now E has K_1 shared with B and
K_2 shared with A

A and B think that they share the key with each other

$$A \to E : E(K_2; M)$$

$$E \to B : E(K_1; M)$$

or

$$E \to B : E(K_1; M')$$

Other Public Key systems

- Digital Signature Standard (DSS)
  - Only for signature

- Elliptic Curve Cryptography (ECC)
  - General Purpose Public Key Encryption Algorithm
  - More difficult to understand than RSA
  - Provides similar security for smaller key size
  - Not tested as much as RSA

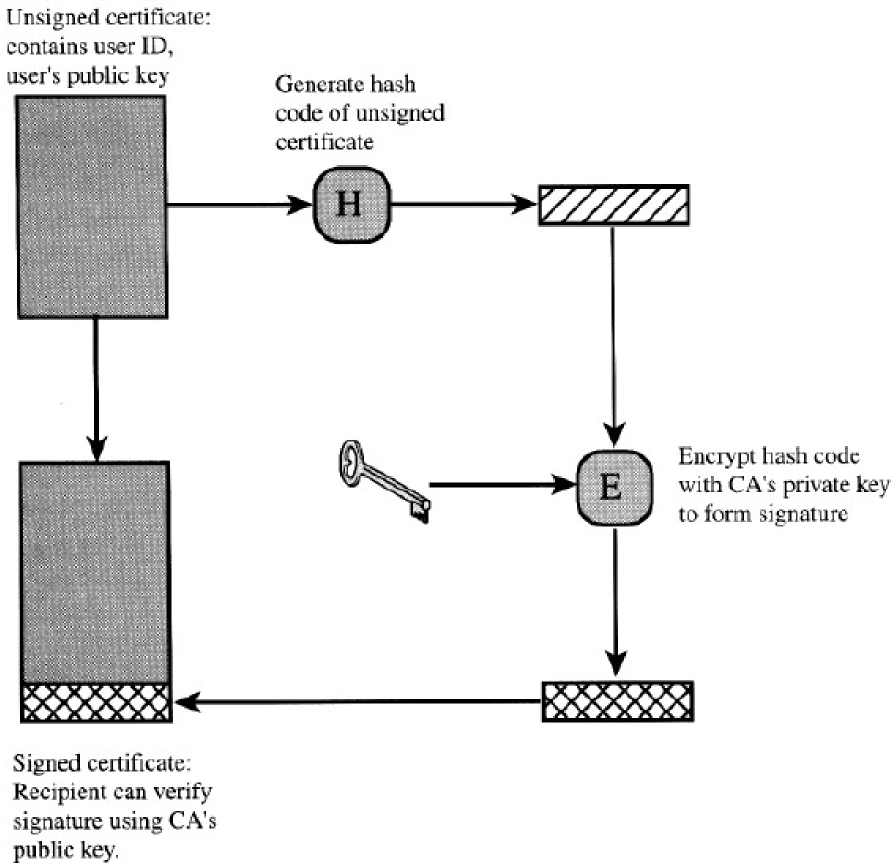
Digital Signatures

1. Encrypt Whole message
   - $C=Ep(Pr_A:M)$
   - C and M must be kept to prove the signature
   - C provides NO confidentiality. Why?

2. Encrypt an authenticator
   - $C=Ep(Pr_A:H(M))$
   - Only M and H(M) need to be kept
   - H used is usually SHA-1
   - No confidentiality. Why?

Distribution of Public Keys

- Public Key Certificates
- CA=Certification Authority
- CA's sign public keys of users with its private key
- X.509 standard
- Used in SSL, Secure Electronic Transaction (SET), S/MIME



Distribution of Shared Keys

1. Use Diffie-Hellman

2. Use Public Key Encryption (Like RSA or ECC)

$$A \rightarrow B : E(k, M) + E_p\left(K_B^{pub}, k\right)$$

Can you see any problem in this exchange in terms of authentication?