

# MODULE 2

## Pseudorandom number Generators

- **Linear Congruential Generators**

$m$	the modulus	$m > 0$
$a$	the multiplier	$0 < a < m$
$c$	the increment	$0 \leq c < m$
$X_0$	the starting value, or seed	$0 \leq X_0 < m$

The sequence of random numbers  $\{X_n\}$  is obtained via the following iterative equation:

$$X_{n+1} = (aX_n + c) \bmod m$$

If  $m$ ,  $a$ ,  $c$ , and  $X_0$  are integers, then this technique will produce a sequence of integers with each integer in the range  $0 \leq X_n < m$ .

- Three tests to be used in evaluating a random number generator:

- T<sub>1</sub>: The function should be a full-period generating function. That is, the function should generate all the numbers from 0 through  $m - 1$  before repeating.
- T<sub>2</sub>: The generated sequence should appear random.
- T<sub>3</sub>: The function should implement efficiently with 32-bit arithmetic.

With appropriate values of  $a$ ,  $c$ , and  $m$ , these three tests can be passed. With respect to  $T_1$ , it can be shown that if  $m$  is prime and  $c = 0$ , then for certain values of  $a$  the period of the generating function is  $m - 1$ , with only the value 0 missing. For 32-bit arithmetic, a convenient prime value of  $m$  is  $2^{31} - 1$ . Thus, the generating function becomes

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

$$X_1 = (aX_0 + c) \bmod m$$

$$X_2 = (aX_1 + c) \bmod m$$

$$X_3 = (aX_2 + c) \bmod m$$

# Blum Blum Shub Generator

- Then the BBS generator produces a sequence of bits  $B_i$  according to the following algorithm:

```
 $x_0 = s^2 \bmod n$ 
for  $i = 1$  to  $\infty$ 
 $x_i = (x_{i-1})^2 \bmod n$ 
 $b_i = x_i \bmod 2$ 
```

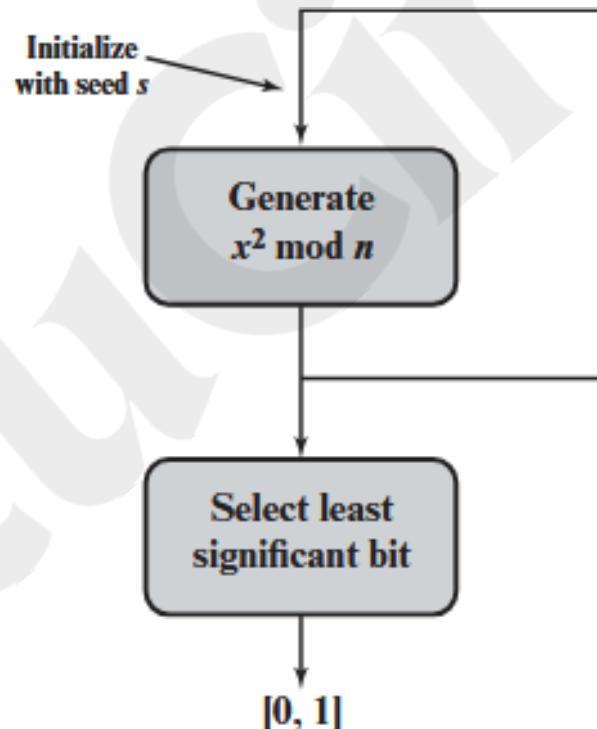


Figure 8.3 Blum Blum Shub Block Diagram

### Example Operation of BBS Generator

$i$	$X_i$	$B_i$
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0

$i$	$X_i$	$B_i$
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

# Public key cryptography

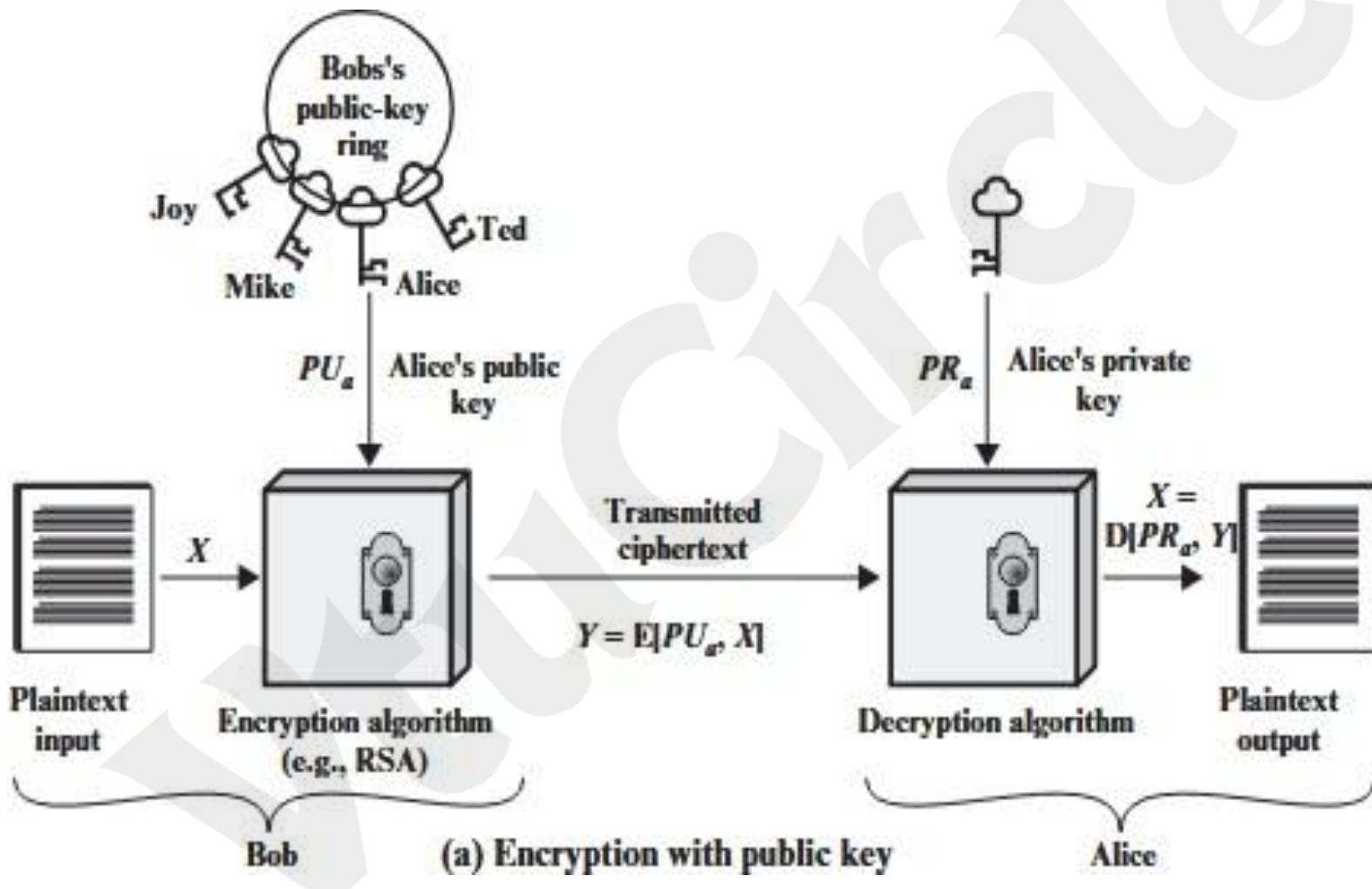
## Principles of Public-Key Cryptosystems

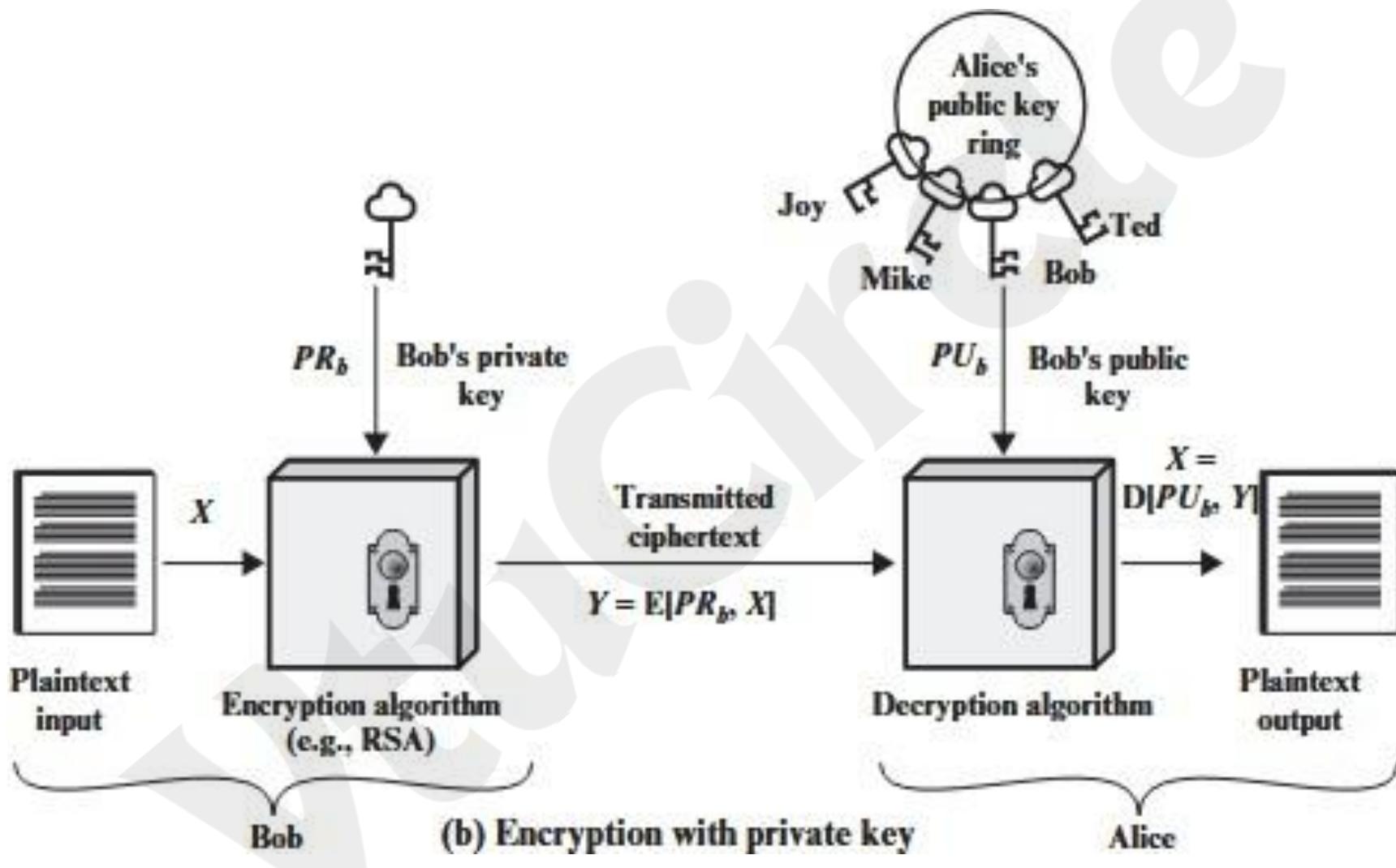
- Developed to address two key issues:
  1. Key distribution – how to have secure communications in general without having to trust a KDC with your key.
  2. Digital signatures – how to verify a message comes intact from the claimed sender.

# Public-Key Cryptosystems

- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.
  - It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
  - In addition, some algorithms, such as RSA, also exhibit the following characteristic.
    - Either of the two related keys can be used for encryption, with the other used for decryption.

# Public-Key Cryptography





- A public-key encryption scheme has ingredients:

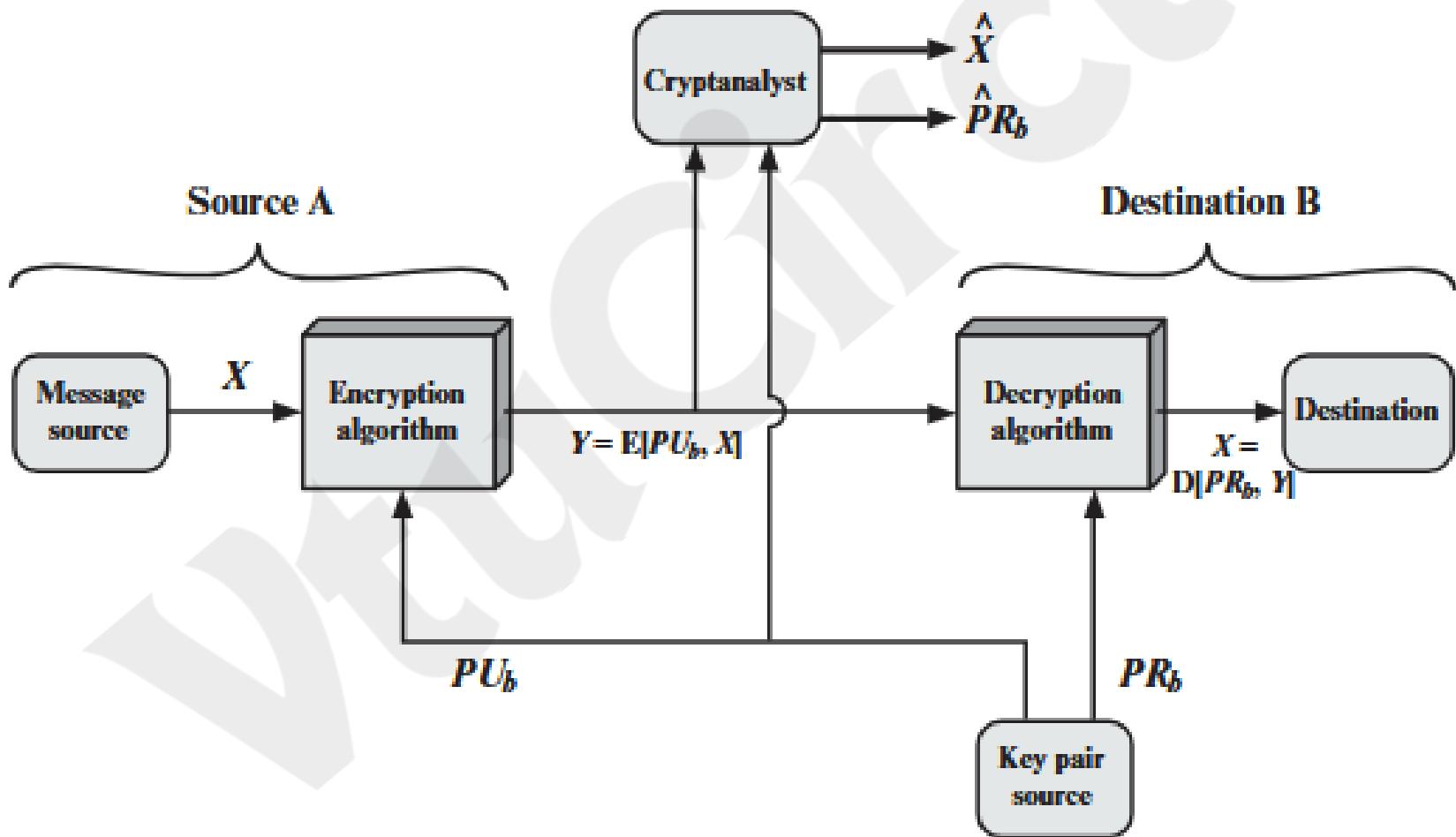
1. **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
2. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
3. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
4. **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
5. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

- The essential steps are the following.
  1. Generates a pair of keys to be used for the encryption and decryption of messages.
  2. Places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private, each user maintains a collection of public keys obtained from others.
  3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
  4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

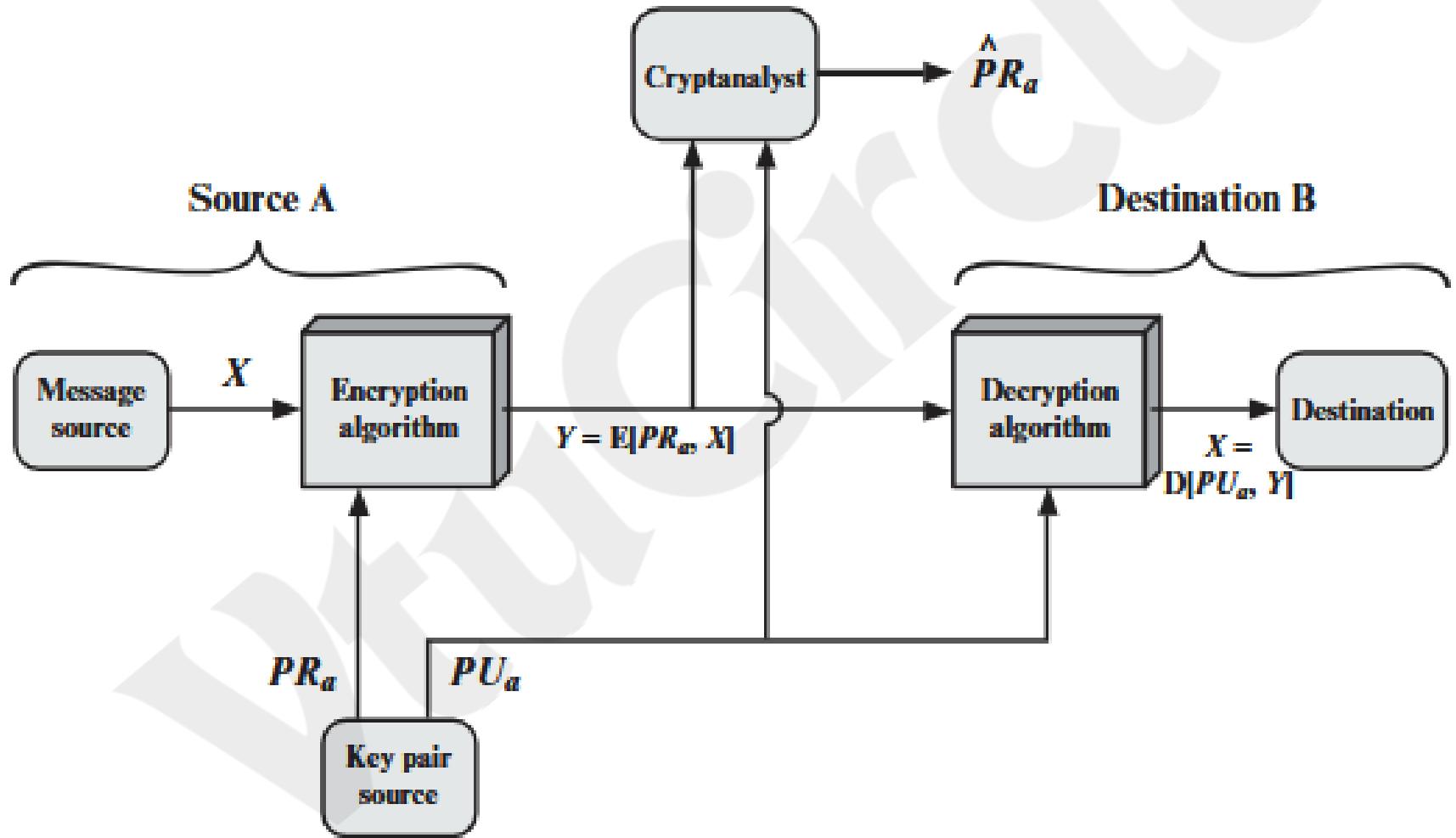
# Comparison

Conventional Encryption	Public-Key Encryption
<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"><li>1. The same algorithm with the same key is used for encryption and decryption.</li><li>2. The sender and receiver must share the algorithm and the key.</li></ol> <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"><li>1. The key must be kept secret.</li><li>2. It must be impossible or at least impractical to decipher a message if the key is kept secret.</li><li>3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.</li></ol>	<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"><li>1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption.</li><li>2. The sender and receiver must each have one of the matched pair of keys (not the same one).</li></ol> <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"><li>1. One of the two keys must be kept secret.</li><li>2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret.</li><li>3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.</li></ol>

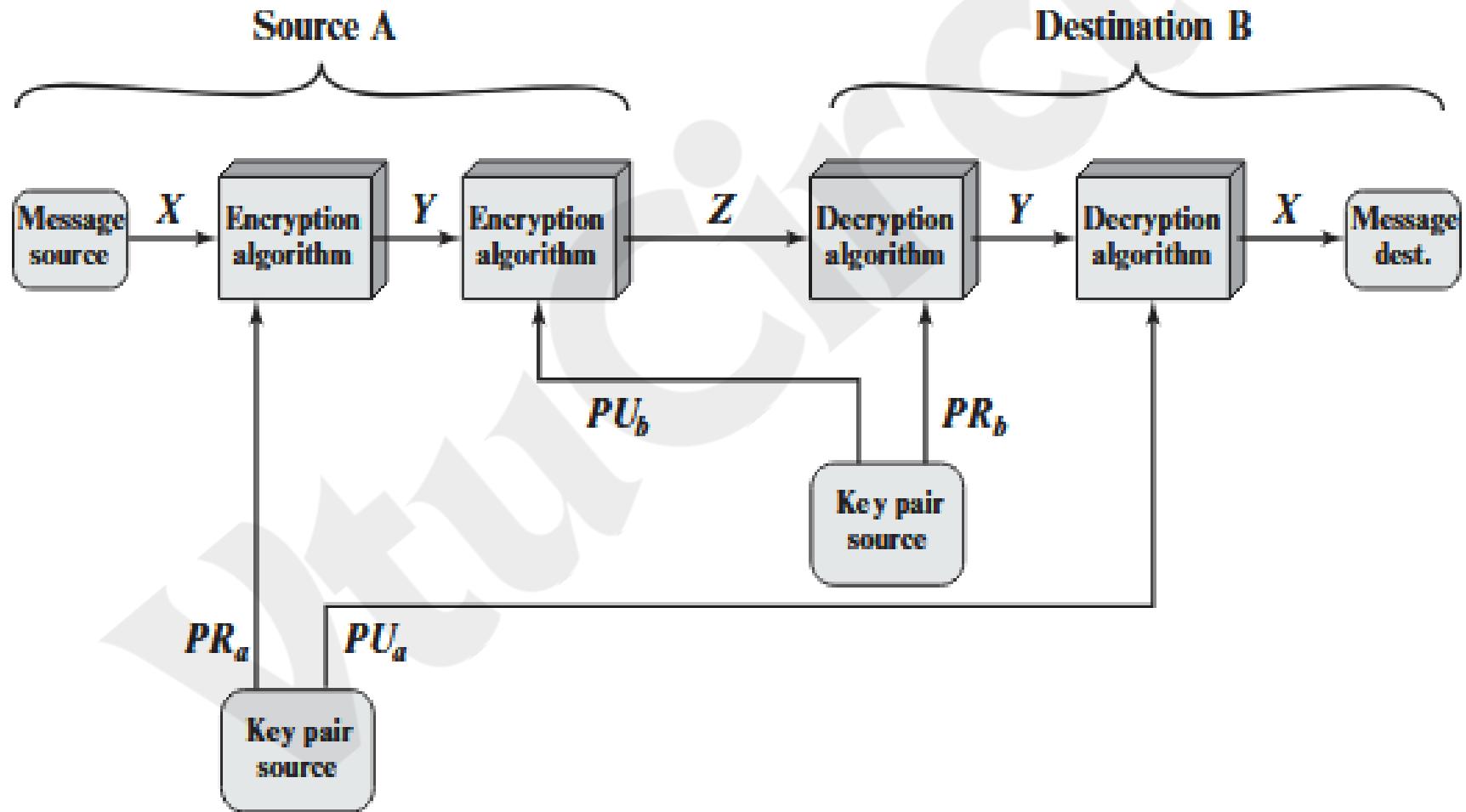
# Public-Key Cryptosystem: Confidentiality



# Public-Key Cryptosystem: Authentication



# Public-Key Cryptosystem: Authentication and Secrecy



# Applications of public-key cryptosystems

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key, and the recipient decrypts the message with the recipient's private key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key, which is a secret key for symmetric encryption generated for use for a particular transaction (or session) and valid for a short period of time. Several different approaches are possible, involving the private key(s) of one or both parties.

# Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a 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:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

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,  $M$ .
6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

- need a trap-door one-way function
- one-way function has

$$Y = f(X) \text{ easy}$$

$$X = f^{-1}(Y) \text{ infeasible}$$

- A trap-door one-way function has

$$Y = f_k(X) \text{ easy, if } k \text{ and } X \text{ are known}$$

$$X = f_k^{-1}(Y) \text{ easy, if } k \text{ and } Y \text{ are known}$$

$$X = f_k^{-1}(Y) \text{ infeasible,, if } Y \text{ known but } k \text{ not known}$$

- A practical public-key scheme depends on a suitable trap-door one-way function

# Public-Key Cryptanalysis

- Vulnerable to a brute-force attack (with small key size), Key size is larger enough to avoid brute-force attack.
- Public key method depend on the use of mathematic equation or some function.
- Public-key encryption is currently confined to key management and signature applications.

# The RSA Algorithm

- Best known & widely used public-key scheme
- Invented by Rivest, Shamir and Adleman in year 1977 and hence name **RSA** algorithm.
- Plaintext and ciphertext are integers between 0 and  $n - 1$  for some  $n$ .
- RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number  $n$ .
  - block size must be less than or equal to  $\log_2(n) + 1$
- For some plaintext block  $M$  and ciphertext block  $C$ .

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- Both sender and receiver must know the value of  $n$ .
- The sender knows the value of  $e$ , and only the receiver knows the value of  $d$ .
- This is a public-key encryption algorithm with a public key of  $PU = \{e, n\}$  and a private key of  $PR = \{d, n\}$ .

**public key  $PU = \{e, n\}$  and private key  $PR = \{d, n\}$**

- Ingredients of RSA:



# The RSA Algorithm

## Key Generation by Alice

Select  $p, q$

$p$  and  $q$  both prime,  $p \neq q$

Calculate  $n = p \times q$

Calcuate  $\phi(n) = (p - 1)(q - 1)$

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 by Bob with Alice's Public Key

Plaintext:

$M < n$

Ciphertext:

$C = M^e \pmod{n}$

## Decryption by Alice with Alice's Public Key

Ciphertext:

$C$

Plaintext:

$M = C^d \pmod{n}$

## Example:

1. Select two prime numbers,  $p = 17$  and  $q = 11$ .
2. Calculate  $n = pq = 17 * 11 = 187$ .
3. Calculate  $f(n) = (p - 1)(q - 1) = 16 * 10 = 160$ .
4. Select  $e$  such that,  $\text{GCD}(160, e) = 1$ . we choose  $e = 7$ .
5. Determine  $d$  such that  $de = 1 \pmod{160}$  and  $d < 160$ . The correct value is  $d = 23$ , because  $23 * 7 = 161$   
$$161 = 1 \pmod{160}$$
6. Publish public key:  $\text{PU} = \{e, n\} = \{7, 187\}$
7. Keep private key secret:  $\text{PR} = \{d, n\} = \{23, 187\}$

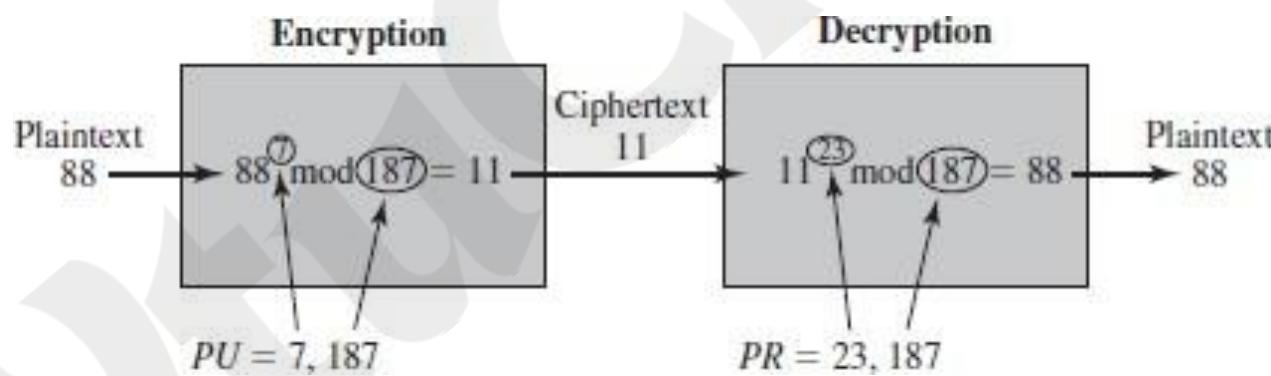
- Given message  $M = 88$

- Encryption:

$$C = 88^7 \bmod 187 = 11$$

- Decryption:

$$M = 11^{23} \bmod 187 = 88$$



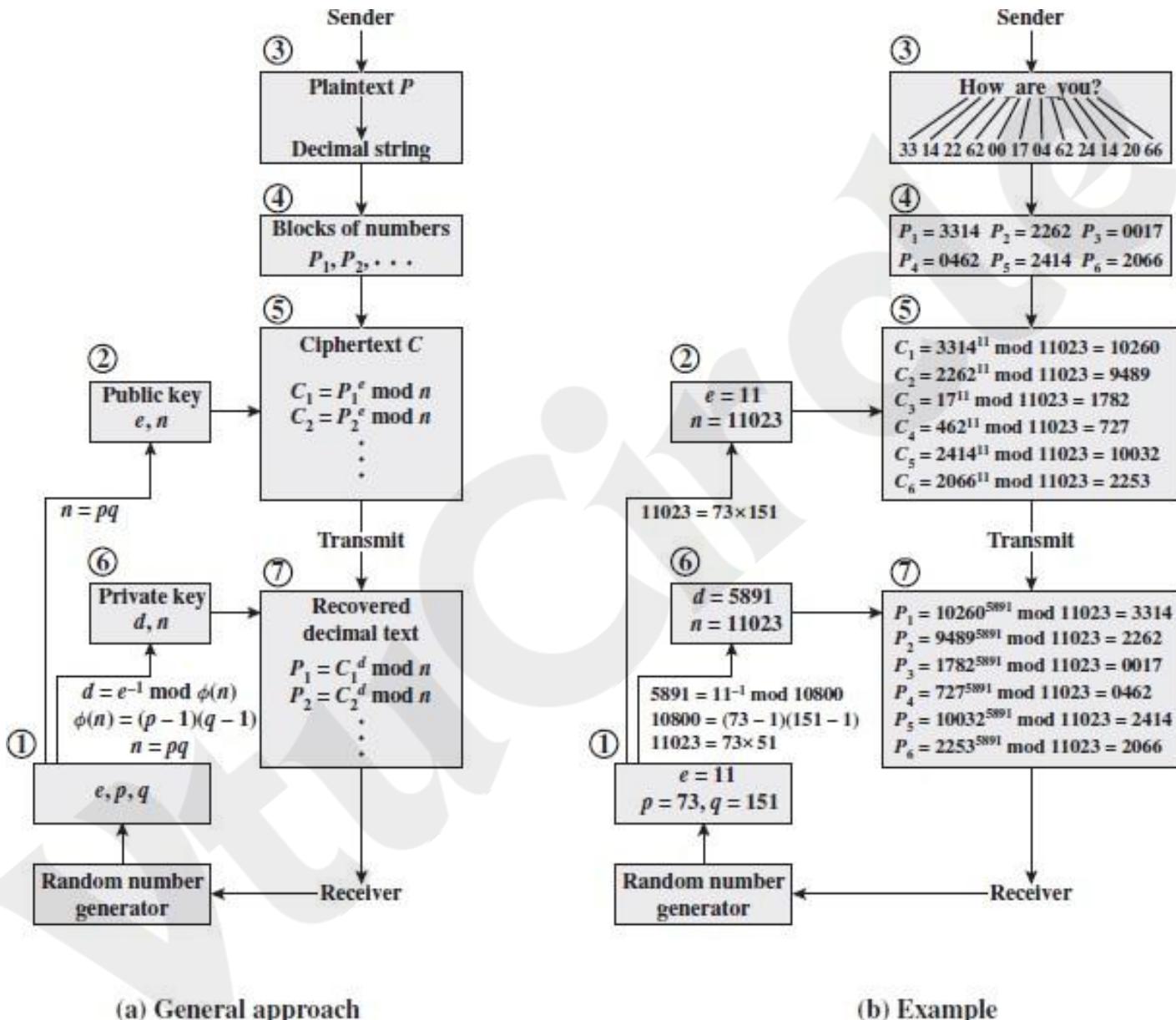


Fig: RSA Processing of Multiple Blocks

# Computational Aspects

## 1. Encryption/decryption

### MODULAR ARITHMETIC :

$[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$  - reduces intermediate results modulo n.

Efficient Operation Using The Public Key : To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made, So the number of multiplications required to perform exponentiation is minimized.

Efficient Operation Using The Private Key : there is a way to speed up Decryption computation using the CRT.

## 2. Key generation

Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks.

- Determining two prime numbers, p and q.
- Selecting either or calculating the others.

# The Security of RSA

- Five possible approaches for attacking the RSA algorithm are:
  - 1.Brute force:** This involves trying all possible private keys.
  - 2.Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
  - 3.Timing attacks:** These depend on the running time of the decryption algorithm.
  - 4.Hardware fault-based attack:** This involves inducing hardware faults in the processor that is generating digital signatures.
  - 5.Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

# The Factoring Problem

- We can identify three approaches to attacking RSA mathematically.
  1. Factor  $n$  into its two prime factors. This enables calculation of  $\Phi(n) = (p - 1) * (q - 1)$ , which in turn enables determination of  $d \equiv K e^{-1} \pmod{\Phi(n)}$ .
  2. Determine  $\Phi(n)$  directly, without first determining  $p$  and  $q$ . Again, this enables determination of  $d \equiv K e^{-1} \pmod{\Phi(n)}$ .
  3. Determine  $d$  directly, without first determining  $\Phi(n)$ .
- currently assume 1024–2048-bit RSA is secure
  - ensure  $p, q$  of *similar* size and matching other constraints

# Timing Attacks

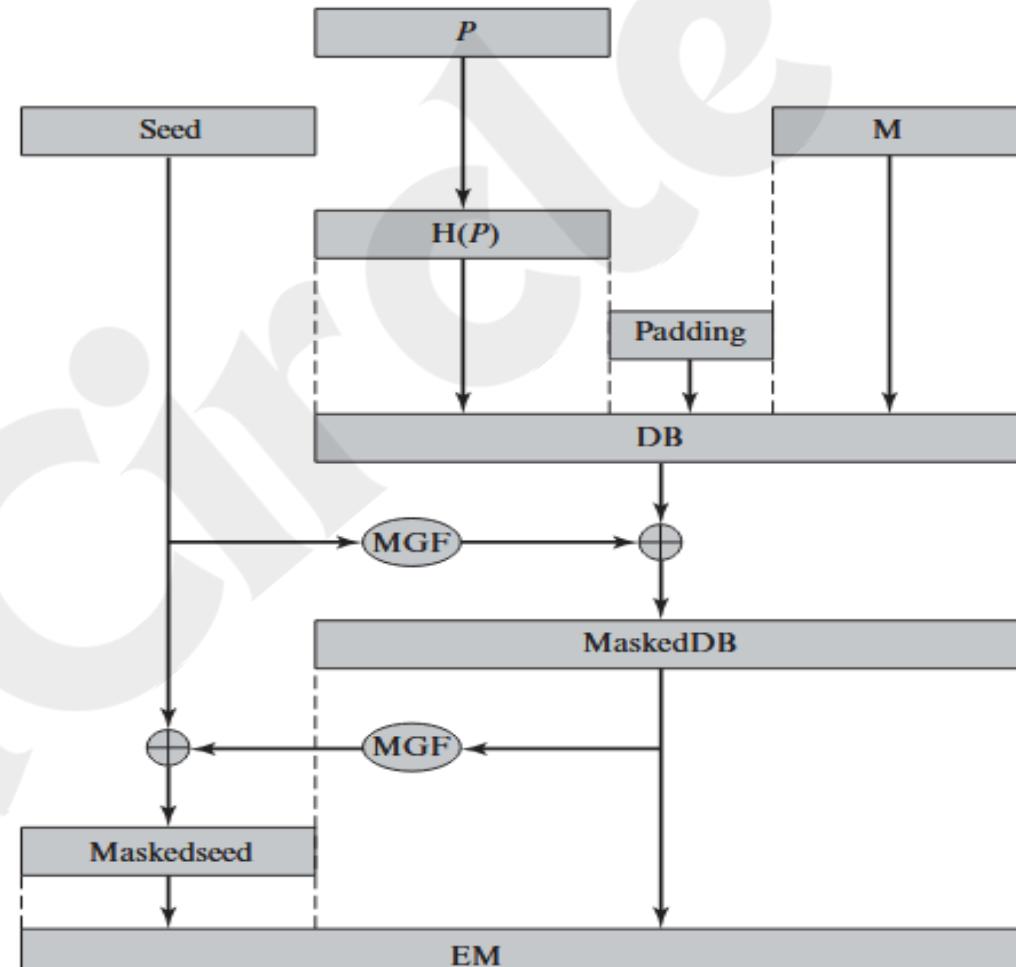
- developed by Paul Kocher in mid-1990's
- exploit timing variations in operations
  - eg. multiplying by small vs.. large number
  - or varying which instructions executed
- infer operand size based on time taken
- For RSA, exploits time taken for exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

# Chosen Ciphertext Attacks

- RSA is vulnerable to a Chosen Ciphertext Attack (CCA)
- based on  $C(P_1 \times P_2) = C(P_1) \times C(P_2)$
- attacker chooses ciphertexts and gets decrypted plaintext back
- choose ciphertext to exploit properties of RSA to provide info to help cryptanalysis
- can counter with random pad off plaintext
- or use Optimal Asymmetric Encryption Padding (OAEP)

# Encryption Using Optimal Asymmetric Encryption Padding (OAEP)

- To counter such attacks, RSA Security Inc., a leading RSA vendor and former holder of the RSA patent, recommends modifying the plaintext using a procedure known as optimal asymmetric encryption padding (OAEP).



**P** = encoding parameters

**M** = message to be encoded

**H** = hash function

**DB** = data block

**MGF** = mask generating function

**EM** = encoded message

## Problems on RSA:

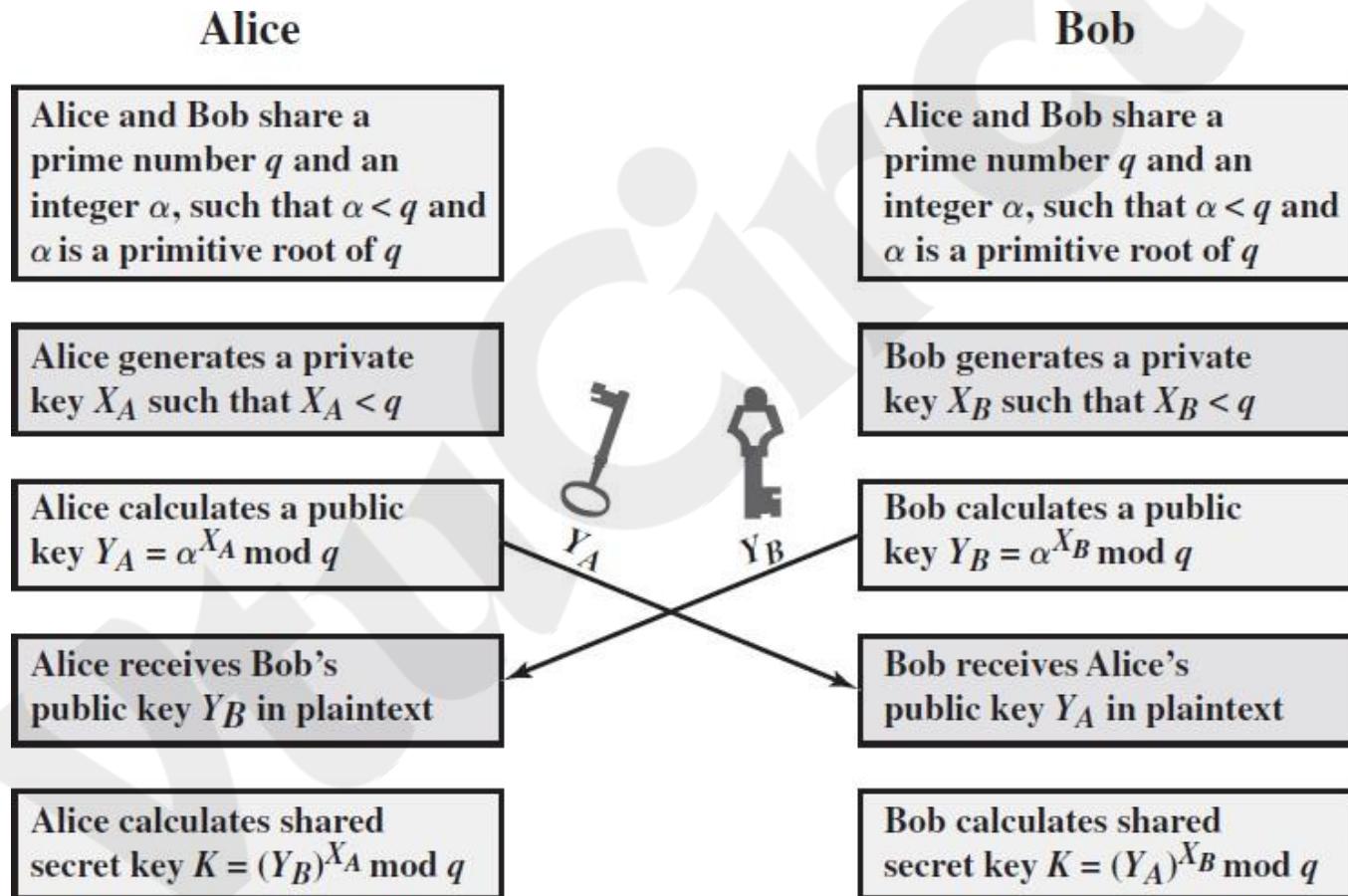
1. Perform encryption and decryption using the RSA algorithm for the following:
  - i.  $p = 3; q = 11, e = 7; M = 5$
  - ii.  $p = 5; q = 11, e = 3; M = 9$
  - iii.  $p = 7; q = 11, e = 17; M = 8$
  - iv.  $p = 11; q = 13, e = 11; M = 7$
  - v.  $p = 17; q = 31, e = 7; M = 2$
2. In a public-key system using RSA, you intercept the ciphertext  $C = 10$  sent to a user whose public key is  $e = 5, n = 35$ . What is the plaintext  $M$ ?

# **Diffie-Hellman key exchange**

- First public-key type scheme proposed by Diffie & Hellman in 1976 for key distribution only.
- Purpose of the algorithm is to enable two users to exchange a secret key securely that then can be used for subsequent encryption of messages.
- The algorithm itself is limited to the exchange of the keys.

# The Algorithm

- There are two publicly known numbers: a **prime number  $q$**  and an **integer  $\alpha$**  that is a primitive root of  $q$ .
- Suppose the users A and B wish to create a shared key.



## Example:

- Let's take  $q = 353$  and  $\alpha = 3$  (a primitive root of 353).
- A and B select private keys  $X_A = 97$  and  $X_B = 233$ , respectively.
- Each computes its public key:
  - A computes  $Y_A = 3^{97} \text{ mod } 353 = 40$ .
  - B computes  $Y_B = 3^{233} \text{ mod } 353 = 248$ .
- After they exchange public keys, each can compute the common secret key:
  - A computes  $K = (Y_B)^{X_A} \text{ mod } 353 = 248^{97} \text{ mod } 353 = 160$ .
  - B computes  $K = (Y_A)^{X_B} \text{ mod } 353 = 40^{233} \text{ mod } 353 = 160$ .
- An attacker would have available the following information but not Private Keys:
$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248.$$

## Man-in-the-Middle Attack

- Suppose Alice and Bob wish to exchange keys, and Darth is the adversary.
- The attack proceeds as follows
  1. Darth prepares for the attack by generating two random private keys  $X_{D1}$  and  $X_{D2}$  and then computing the corresponding public keys  $Y_{D1}$  and  $Y_{D2}$ .
  2. Alice transmits  $Y_A$  to Bob.
  3. Darth intercepts  $Y_A$  and transmits  $Y_{D1}$  to Bob. Darth also calculates  $K2 = (YA)^{XD2} \bmod q$ .
  4. Bob receives  $Y_{D1}$  and calculates  $K1 = (Y_{D1})^{XB} \bmod q$ .
  5. Bob transmits  $Y_B$  to Alice.
  6. Darth intercepts  $Y_B$  and transmits  $Y_{D2}$  to Alice. Darth calculates  $K1 = (Y^B)^{XD1} \bmod q$ .
  7. Alice receives  $Y_{D2}$  and calculates  $K2 = (Y_{D2})^{XA} \bmod q$ .

- At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key  $K_1$  and Alice and Darth share secret key  $K_2$ .
- All future communication between Bob and Alice is compromised in the following way.
  1. Alice sends an encrypted message  $M$ :  $E(K_2, M)$ .
  2. Darth intercepts the encrypted message and decrypts it to recover  $M$ .
  3. Darth sends Bob  $E(K_1, M)$  or  $E(K_1, M')$ , where  $M'$  is any message.
- In the first case, Darth simply wants to eavesdrop on the communication without altering it.
- In the second case, Darth wants to modify the message going to Bob.



Private key  $X_A$   
Public key  
 $Y_A = \alpha^{X_A} \text{ mod } q$



Private keys  $X_{DI}, X_{D2}$   
Public keys  
 $Y_{DI} = \alpha^{X_{DI}} \text{ mod } q$   
 $Y_{D2} = \alpha^{X_{D2}} \text{ mod } q$



Secret key  
 $K2 = (Y_{D2})^{X_A} \text{ mod } q$

Secret key  
 $K2 = (Y_A)^{X_{D2}} \text{ mod } q$

Private key  $X_B$   
Public key  
 $Y_B = \alpha^{X_B} \text{ mod } q$



Secret key  
 $KI = (Y_B)^{X_{DI}} \text{ mod } q$

Secret key  
 $KI = (Y_{DI})^{X_B} \text{ mod } q$

Alice and Darth  
share  $K2$



Bob and Darth  
share  $KI$

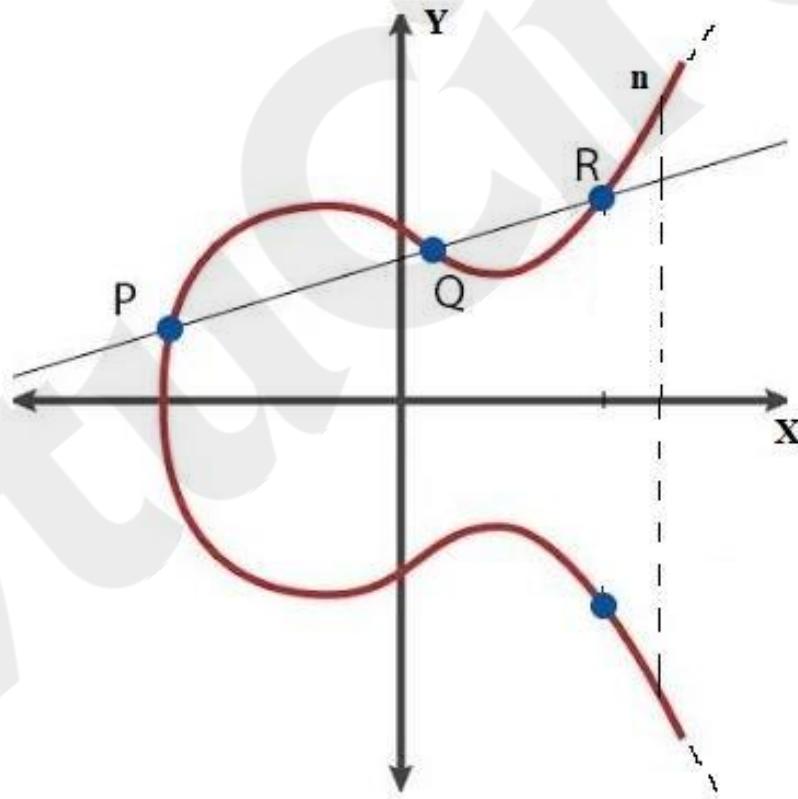


# Problems on Diffie-Hellman algorithm

1. Users A and B use the Diffie-Hellman key exchange technique with a common prime  $q = 71$  and a primitive root  $\alpha = 7$ .
  - a. If user A has private key  $X_A = 5$ , what is A's public key  $Y_A$ ?
  - b. If user B has private key  $X_B = 12$ , what is B's public key  $Y_B$ ?
  - c. What is the shared secret key?
2. Consider a Diffie-Hellman scheme with a common prime  $q = 11$  and a primitive root  $\alpha = 2$ .
  - a. If user A has public key  $Y_A = 9$ , what is A's private key  $X_A$ ?
  - b. If user B has public key  $Y_B = 3$ , what is the secret key  $K$  shared with A?

# Elliptic Curve Cryptography

- Key exchange using elliptic curves
- It is a Public Key Cryptosystem
- It provides Equal security with smaller key size compared to Non ECC Algorithms
- It makes use of Elliptic curves
- Elliptic curves are defined by some mathematical functions.  $y^2 = x^3 + ax + b$



# Analog of Diffie-Hellman key Exchange

- Key exchange using elliptic curves can be done in the following manner.
- First pick a large integer  $q$ , which is either a prime number  $q$  or an integer of the form  $2^m$ , and elliptic curve parameters  $a$  and  $b$ . This defines the elliptic group of points  $E_q(a, b)$ . Next, pick a base point  $\mathbf{G} = (x_1, y_1)$  on an elliptic curve. The order  $n$  of a point  $G$  on an elliptic curve is the large positive integer  $n$  and these parameters of the cryptosystem known to all participants.

A key exchange between users A and B can be accomplished as follows

1. A selects an integer  $n_A$  less than  $n$ . This is A's private key. A then generates a public key  $P_A = n_A \times G$ ; the public key is a point in  $E_q(a, b)$ .
2. B similarly selects a private key  $n_B$  and computes a public key  $P_B$ .
3. A generates the secret key  $k = n_A \times P_B$ . B generates the secret key  $k = n_B \times P_A$ .

The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

# ECC Diffie–Hellman Key Exchange

## Global Public Elements

$E_q(a, b)$  elliptic curve with parameters  $a, b$ , and  $q$ , where  $q$  is a prime or an integer of the form  $2^m$

$G$  point on elliptic curve whose order is large value  $n$

## User A Key Generation

Select private  $n_A$   $n_A < n$

Calculate public  $P_A$   $P_A = n_A \times G$

## User B Key Generation

Select private  $n_B$   $n_B < n$

Calculate public  $P_B$   $P_B = n_B \times G$

## Calculation of Secret Key by User A

$$K = n_A \times P_B$$

## Calculation of Secret Key by User B

$$K = n_B \times P_A$$

# Elliptic Curve Encryption/Decryption

- Let the message be  $\mathbf{m}$ .
- First Encode this message as an  $(x, y)$  point  $P_m$  on elliptic curve.
- To encrypt and send a message  $P_m$  to B, A chooses a random positive integer  $k$  and produces the ciphertext  $C_m$  consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

- This point will be sent to the receiver.
- To decrypt the ciphertext, B multiplies the first point in the cyptext pair by B's private key and subtracts the result from the second point:

$$\begin{aligned} & P_m + kP_B - n_B(kG) \\ &= P_m + k(n_B G) - n_B(kG) \\ &= P_m \end{aligned}$$

# Security of Elliptic Curve Cryptography

- The security of ECC depends on how difficult it is to determine  $k$ .
- This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method.
- Considerably smaller key size can be used for ECC compared to RSA.
- There is a computational advantage of using ECC with a shorter key length than a comparably secure RSA.

# Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of $n$ in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15,360$ $N = 512$	15,360	512+

Note:  $L$  = size of public key,  $N$  = size of private key.