MCA : MCA23203

CRYPTOGRAPHY & NETWORK SECURITY

UNIT - 3

Unit-3: Public key cryptography and Authentication requirements

Principles of public key cryptosystems – RSA algorithm- security of RSA – key management – Diffie – Hellman key exchange algoritm – the introductory idea of Elliptic curve cryptography – Elgamel encryption – Message Authentication and Hash function: Authentication requirements – authentication functions – message authentication code – hash functions – birthday attacks – security of hash functions and MACS.

Principles of Public key Cryptography

- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys- one public key and one private key
- Also known as public-key encryption
- It uses mathematical functions rather than substitution and permutation
- More secure from cryptanalysis than the symmetric encryption

Cont…

- **Asymmetric keys**
  - Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification
- **Public key certificate**
  - A digital document issued and digitally signed by the private key of a Certification authority that fixes the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key

Cont…

- **Public key cryptographic algorithm**
  - A cryptographic algorithm that uses two related keys, a public key and a private key
- **Public key infrastructure**
  - A set of policies, processes, server platform, software and workstations used for the purpose of controlling certificates and public-private key pairs, including the ability to issue, maintain, and cancel public certificate

Public-Key Cryptosystems

A public-key encryption scheme has six ingredients

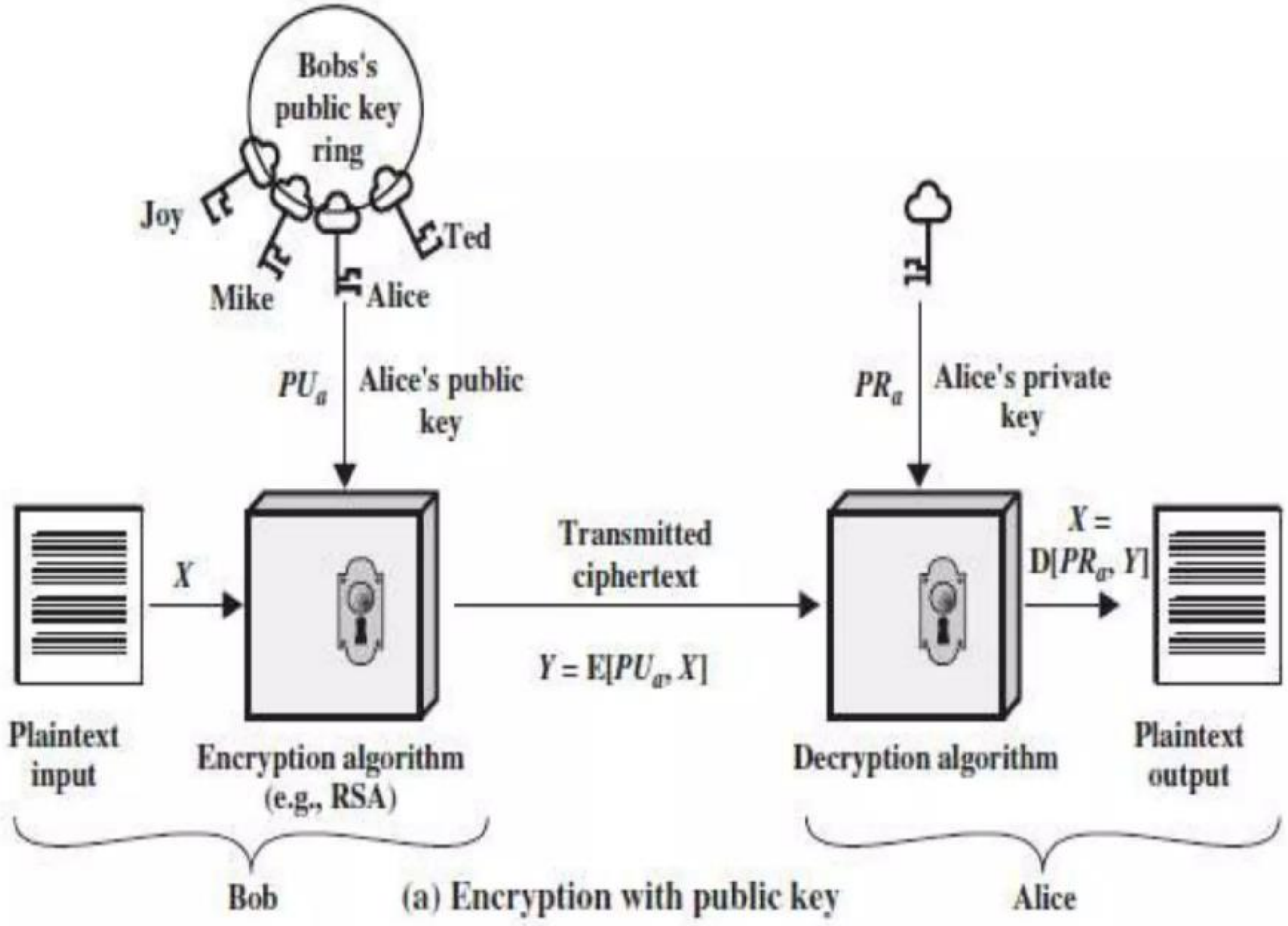
- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for 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.
- **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.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following:

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

ENCRYPTION

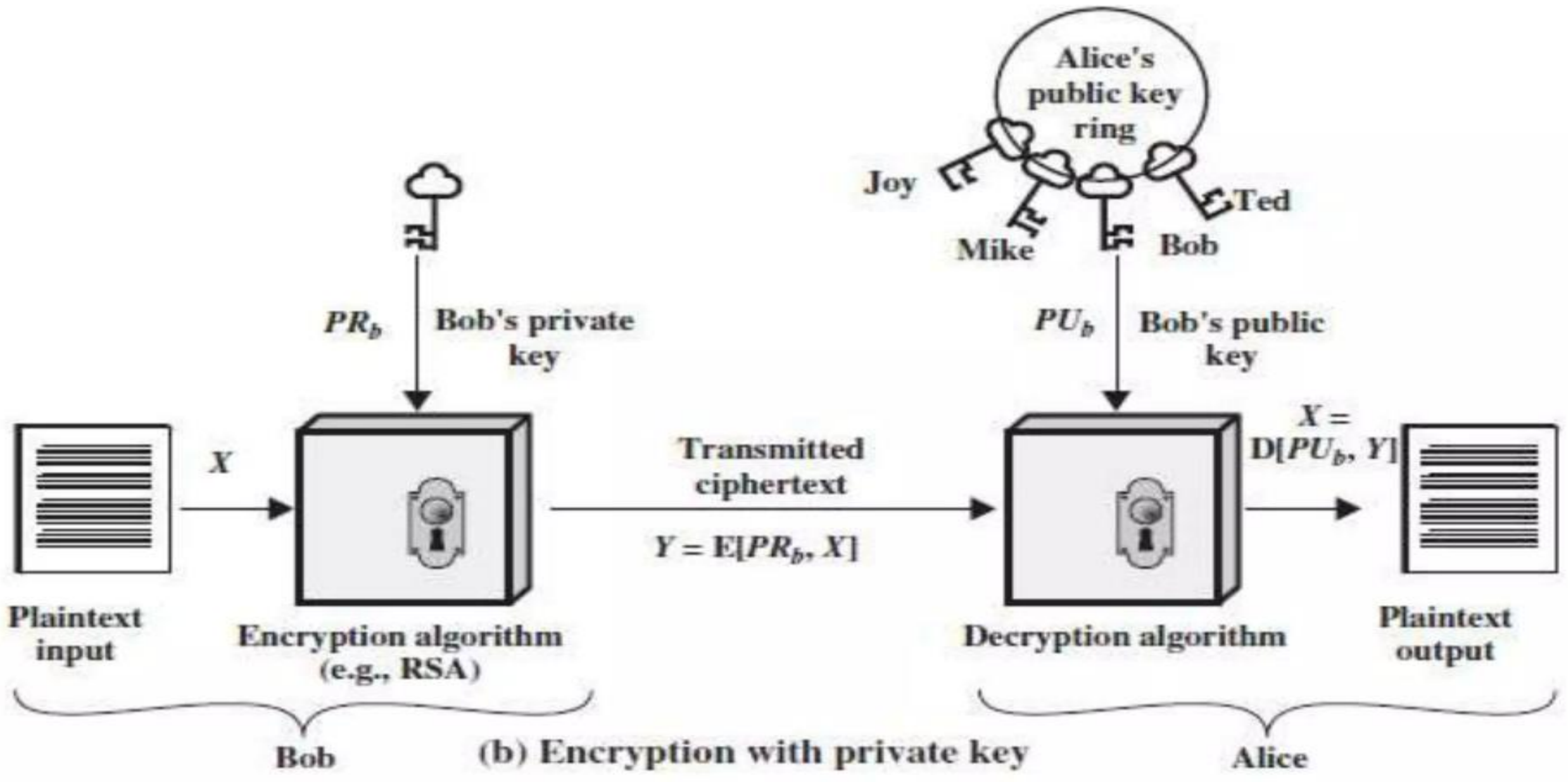
- The plaintext is encrypted with receiver's public key and decrypted using receiver private key.



(a) Encryption with public key

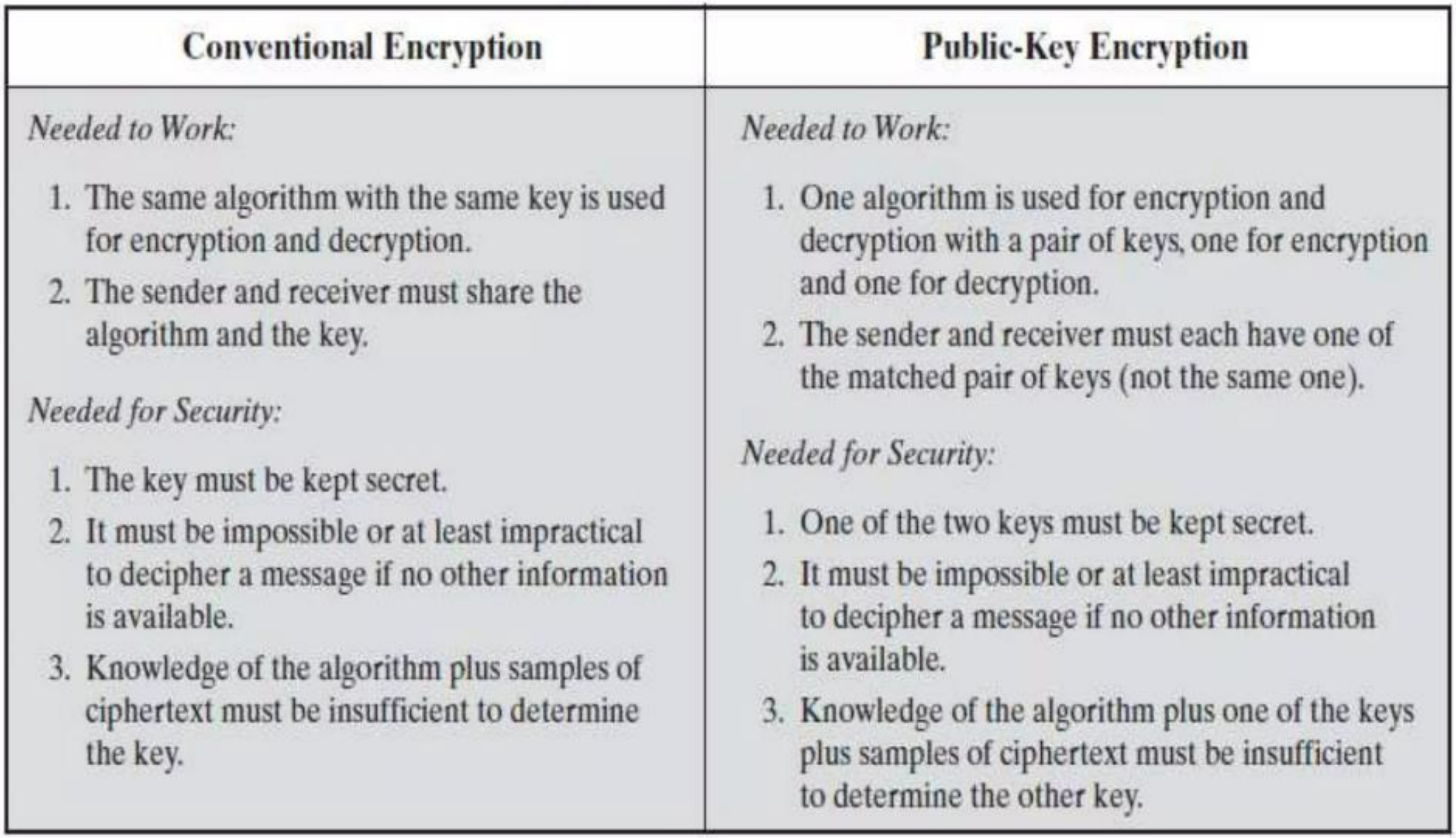
Authentication

- Plaintext is encrypted is sender's private key and decrypted using sender's public key.
- The act of messages ciphertext getting decrypted by senders public key is the proof that the message is actually sent by the designated sender.



(b) Encryption with private key

Symmetric Vs Public-key



Conventional Encryption	Public-Key Encryption
Needed to Work:	*Needed to Work:*
1. The same algorithm with the same key is used for encryption and decryption.	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 share the algorithm and the key.	2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security:	*Needed for Security:*
1. The key must be kept secret.	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.	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.	3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Applications for Public-Key Cryptosystems

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message.
- **Key exchange:** Two sides cooperate to exchange a session key.

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 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.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

The two keys can be applied in either order:

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

The RSA Algorithm

- RSA is a public key encryption algorithm developed by Rivert(R) , Shamir(S) and Adleman (A) in year 1977.
- The RSA scheme is a block cipher in which the plaintext & ciphertext are integers between 0 and n-1 for some 'n'.
- A typical size for 'n' is 1024 bits or 309 decimal digits.

Description of the Algorithm

- RSA algorithm uses an expression with exponentials.
- In RSA plaintext is encrypted in blocks, with each block having a binary value less than some number n. that is, the block size must be less than or equal to log2(n)+1
- RSA uses two exponents 'e' and 'd' where e→public and d→private.
- Encryption and decryption are of following form, for some **PlainText 'M'** and **CipherText block 'C'**

$$\boxed{C = M^e \bmod n}$$

$$\boxed{M = C^d \bmod n}$$

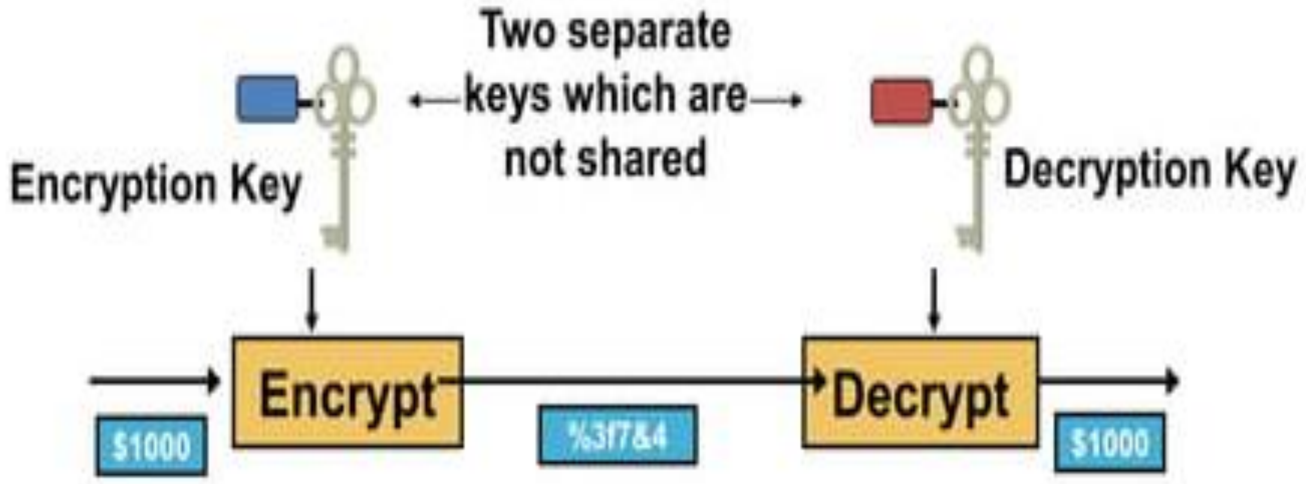
$$\boxed{M = C^d \bmod n = (M^e \bmod n)^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n}$$

- Both sender and receiver must know the value of n.
- The sender knows the value of 'e' & only the reviver knows the value of 'd' thus this is a public key encryption algorithm with a

  Public key PU={e, n}

  Private key PR={d, n}

- RSA is Asymmetric Encryption



RSA Key generation

1) Choose two distinct prime numbers $\boldsymbol{p}$ and $\boldsymbol{q}$

2) Compute $\boldsymbol{n} = p * q$

3) Compute **φ(n)** = (p - 1) * (q - 1)

4) Choose **e** such that 1 < e < φ(n) and e and n are prime.

5) Compute a value for **d** such that (d * e) % φ(n) = 1

❑ Public key is (e, n)

❑ Private key is (d, n)

RSA Key generation Example

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < φ(n) and e and n are prime. Let e = 7
- Compute a value for d such that (d * e) % φ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)

RSA Encryption

- ❑ *m = plaintext*
- ❑ Public key is (e, n)
- ❑ *C= Ciphertext*

- ❑ $C = m^e \,\%\, n$

RSA Encryption Example

- $m = 2$
- Public key is (e, n) => (7, 33)
- $C = 2^7 \% 33 = 29$

RSA Decryption

- *C= Ciphertext*
- *m = plaintext*
- Private key is (d, n)

- $m = C^d \% n$

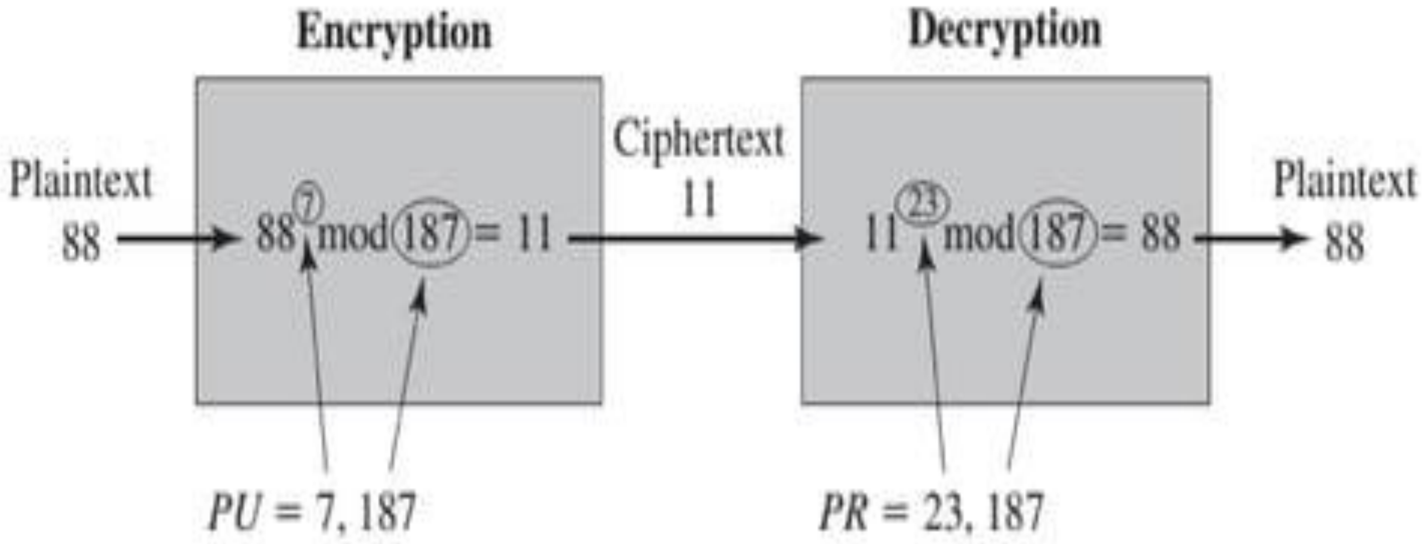
RSA Decryption Example

- $C = 29$
- Private key is (d, n) => (3, 33)
- $m = C^3 \% 33 = 2$

RSA Another Example

- Select two prime numbers, $p = 17$ and $q = 11$.
- Calculate $n = pq = 17 * 11 = 187$.
- Calculate $\varphi(n) = (p - 1)(q - 1) = 16 * 10 = 160$.
- Select e such that e is relatively prime to $\varphi(n) = 160$ and less than $\varphi(n)$; we choose $e = 7$.
- Determine d such that $d.e = 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 * 7 = 161 = (1 * 160) + 1$
- *Public Key*= {7, 187} and Private Key = {23, 187}

RSA Another Example



RSA Security

- Two approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)

What is Key Management?

Key management is the set of techniques and procedures supporting the **establishment** and **maintenance** of keying relationships between **authorized parties**.

What is Key Generation/Establishment?

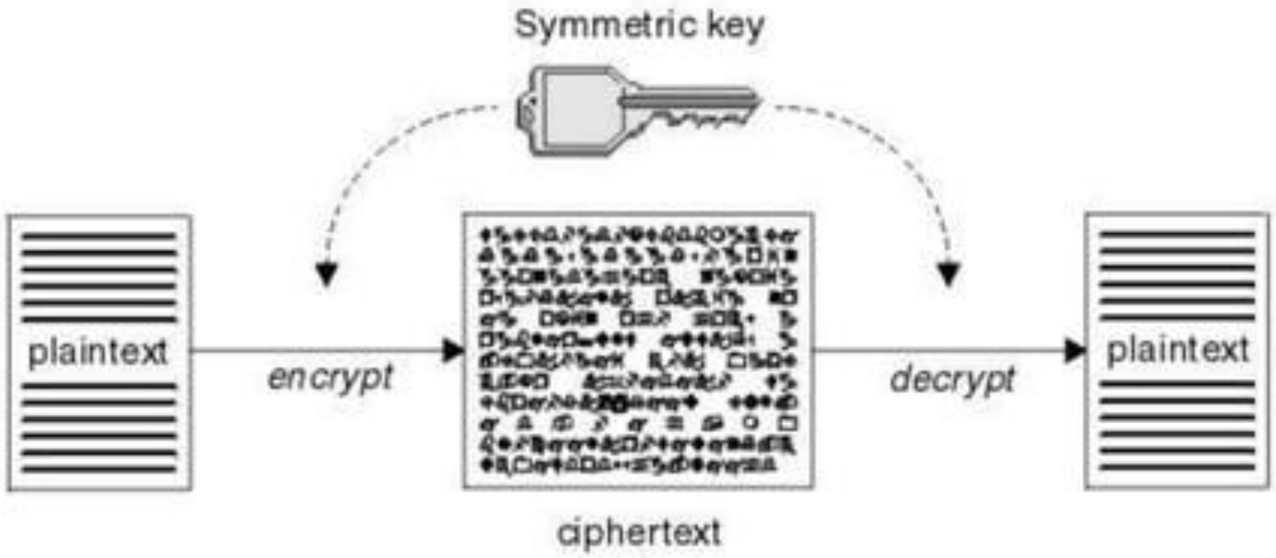
Key generation is the process of **generating keys** for cryptography. The **key** is used to encrypt and decrypt data whatever the data is being encrypted or decrypted.

Types of Keys in Cryptography

1. Symmetric Key
2. Asymmetric Key

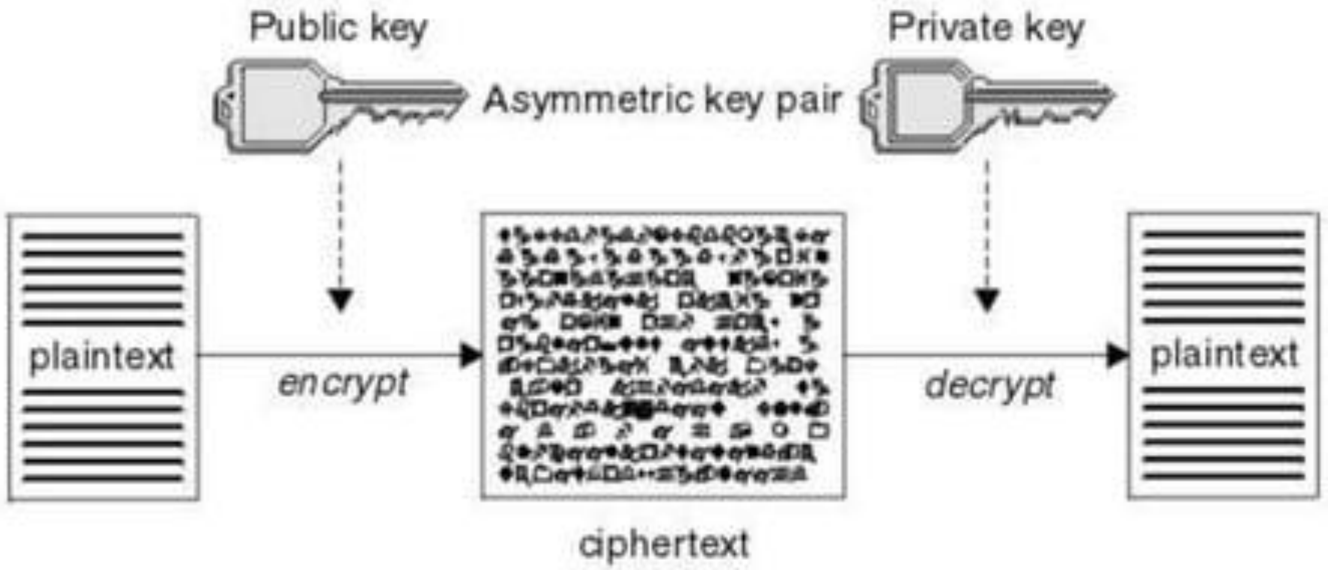
Symmetric Key

Symmetric key is (also called "**secret key / private key / conventional key / single key**") use the same **key** for both encryption and decryption.



Asymmetric Key

Asymmetric key is (also called "**public key / two key**") use different **keys** for encryption and decryption.



Symmetric Key Encryption & Decryption

An **algorithm** in which the sender and receiver of a message share a single, common **key** that is used to **encrypt** and decrypt the message.

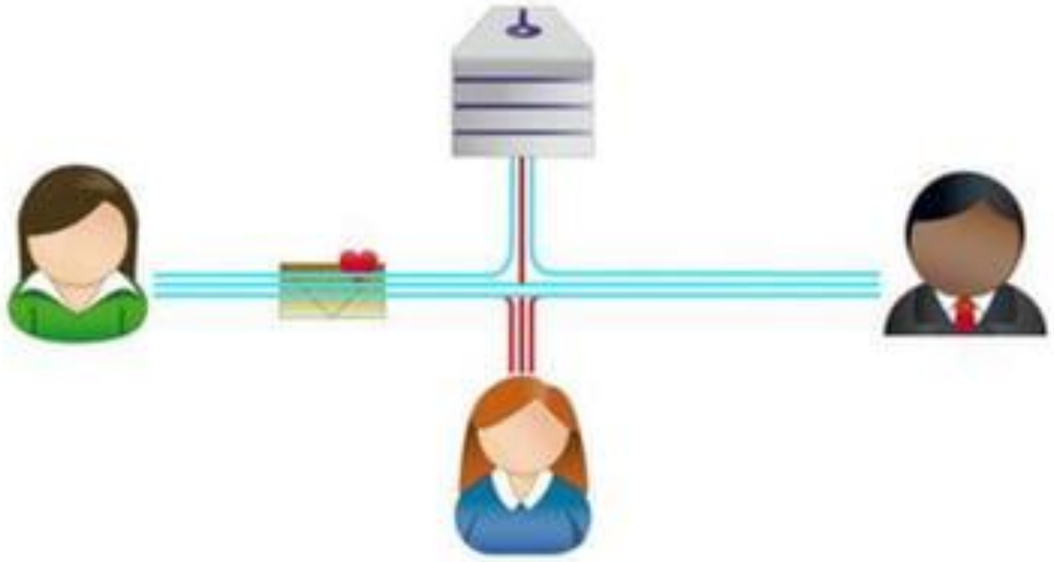
Private Key

Asymmetric Key Encryption & Decryption

An **algorithm** in which the sender and receiver of a message share different **keys (public key & private key)** that is used to **encrypt** and **decrypt** the message.

What is Key Distribution?

Key distribution in cryptography is a system that is responsible for providing keys to the users in a network that shares sensitive or private data.



Types of Key Distribution

- Public Key Distribution

  1. Public Announcement
  2. Publically Available Directory
  3. Public Key Authority
  4. Public Key Certificate

- Private Key Distribution

  1. Simple Secret Key
  2. Secret Key with Confidentiality & Authentication

Elliptic Curve Cryptography (ECC)

- For the same length of keys, faster than RSA
- For the same degree of security, shorter keys are required than RSA
- Standardized in IEEE P1363
- Confidence level not yet as high as that in RSA
- Much more difficult to explain than RSA

- Named so because they are described by cubic equations (used for calculating the circumference of an ellipse)
- Of the form $y^2 + axy + by = x^3 + cx^2 + dx + e$
  where all the coefficients are real numbers satisfying some simple conditions
- Single element denoted O and called the *point a* *infinity* or the *zero point*

- Define the rules of addition over an elliptic curve
  - O serves as the additive identity. Thus $O = -O$; for any point P on the elliptic curve, $P + O = P$.
  - $P_1 = (x,y)$, $P_2 = (x,-y)$. Then, $P_1 + P_2 + O = O$, and therefore $P_1 = -P_2$.
  - To add two points Q and R with different x coordinates, draw a straight line between them and find the third point of intersection P_1. If the line is tangent to the curve at either Q or R, then $P_1 = Q$ or R. Finally, $Q + R + P_1 = O$ and $Q + R = -P_1$.

- Define the rules of addition over an elliptic curve (cont'd)
  - To double a point Q, draw the tangent line and find the other point of intersection S. Then $Q + Q = 2Q = -S$.

- Elliptic curves over finite field
  - Define ECC over a finite field
  - The elliptic group mod p, where p is a prime number
  - Choose 2 nonnegative integers a and b, less than p that satisfy

    $$[4a^3 + 27b^2] \pmod{p} \neq 0$$
  - $E_p(a,b)$ denotes the elliptic group mod p whose element (x,y) are pairs of non-negative integers less than p satisfying

    $y^2 \equiv x^3 + ax + b \pmod{p}$, with O

- Elliptic curves over finite field (cont'd)
  - Example: Let $p = 23$, $a = b = 1$. This satisfies the condition for an elliptic curve group mod 23.

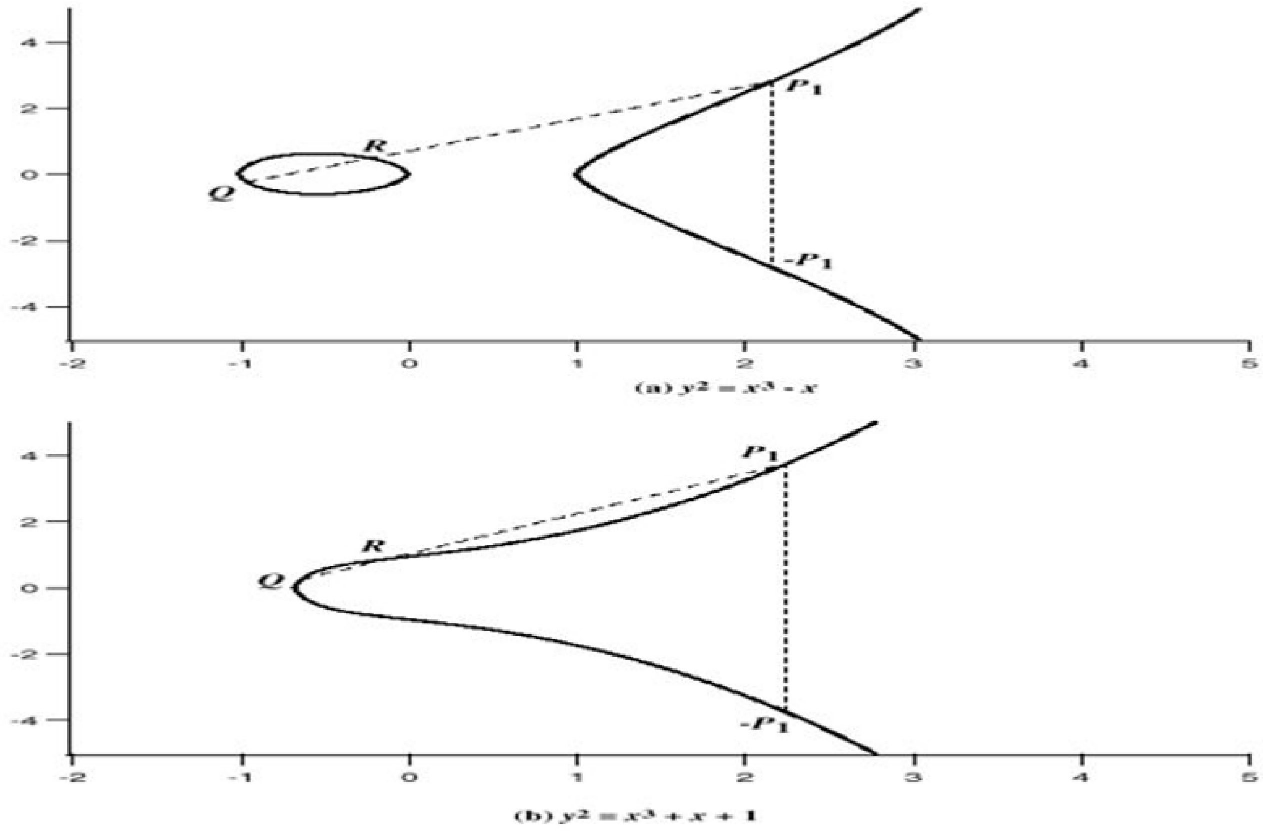


Figure 6.18 Example of Elliptic Curves

- Generation of nonnegative integer points from (0,0) to (p,p) in E_p

1. For each x such that $0 \le x < p$, calculate $x^3 + ax + b \pmod{p}$.
2. For each result from the previous step, determine if it has a square root mod p. If not, there are no points in $E_p(a, b)$ with this value of x. If so, there will be two values of y that satisfy the square root operation (unless the value is the single y value of 0). These (x, y) values are points in $E_p(a, b)$.

- Rules of addition over $E_p(a,b)$

1. $P + O = P$.
2. If $P = (x, y)$, then $P + (x, -y) = O$. The point $(x, -y)$ is the negative of P, denoted as $-P$. Observe that $(x, -y)$ is a point on the elliptic curve, as seen graphically (Figure 6.18b) and in $E_p(a, b)$. For example, in $E_{23}(1, 1)$, for $P = (13,7)$, we have $-P = (13, -7)$. But $-7 \bmod 23 = 16$. Therefore, $-P = (13, 16)$, which is also in $E_{23}(1, 1)$.

Table 6.4 Points on the Elliptic Curve $E_{23}(1, 1)$

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

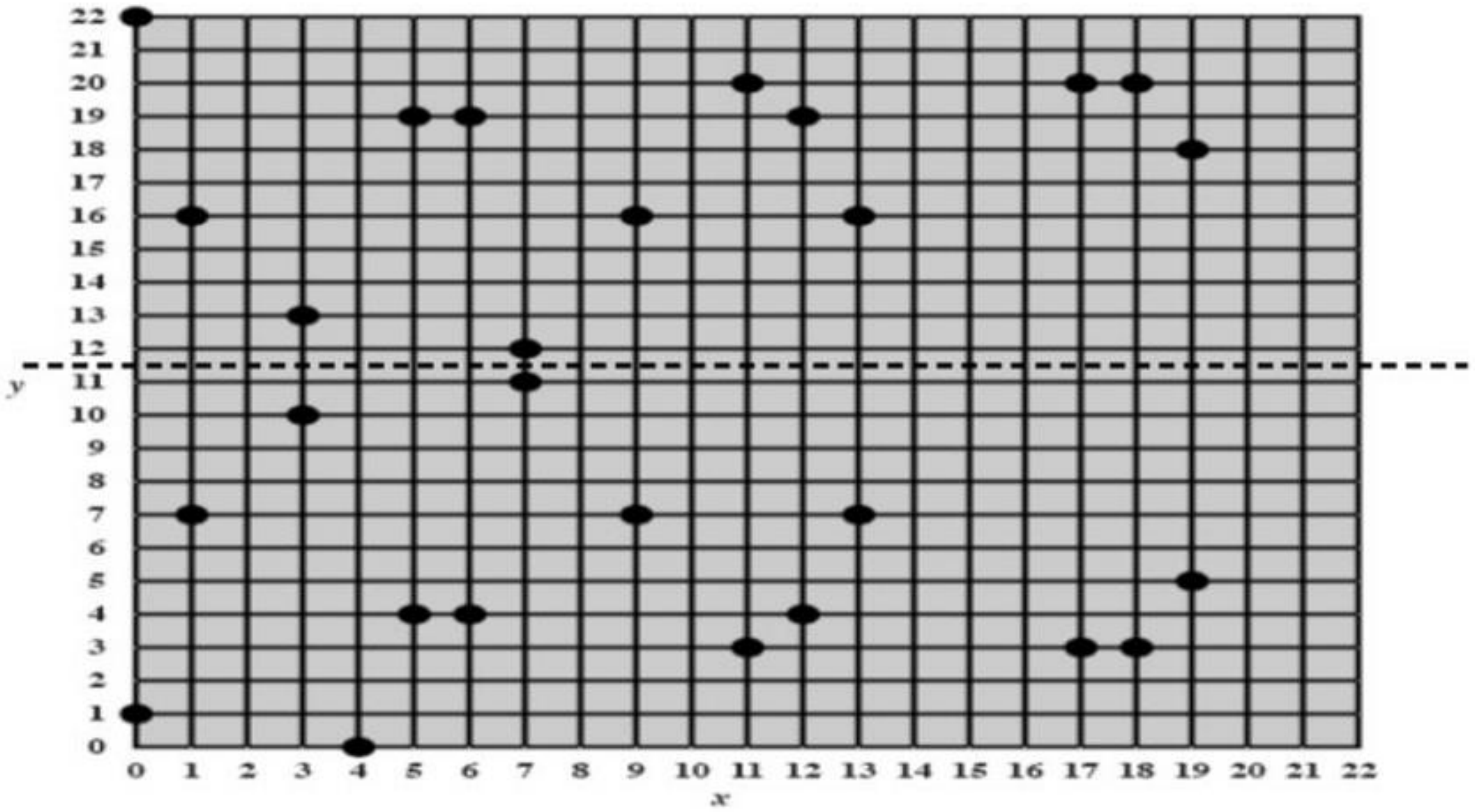
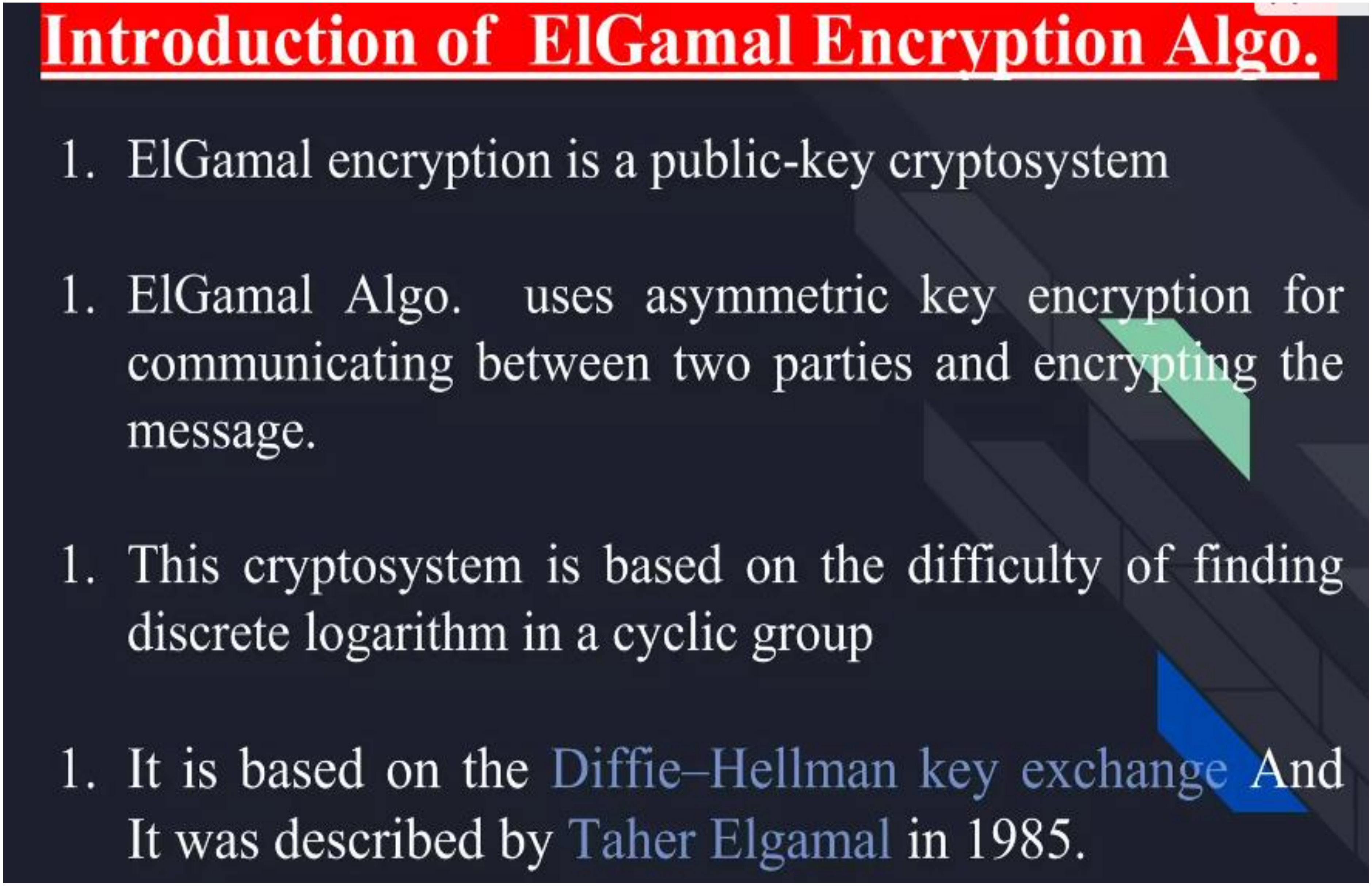


Figure 10.10 The Elliptic Curve $E_{23}(1,1)$



Introduction of ElGamal Encryption Algo.

1. ElGamal encryption is a public-key cryptosystem

1. ElGamal Algo. uses asymmetric key encryption for communicating between two parties and encrypting the message.

1. This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group

1. It is based on the Diffie–Hellman key exchange And It was described by Taher Elgamal in 1985.

Components of ElGamal Encryption Algo.

ElGamal encryption consists of three components

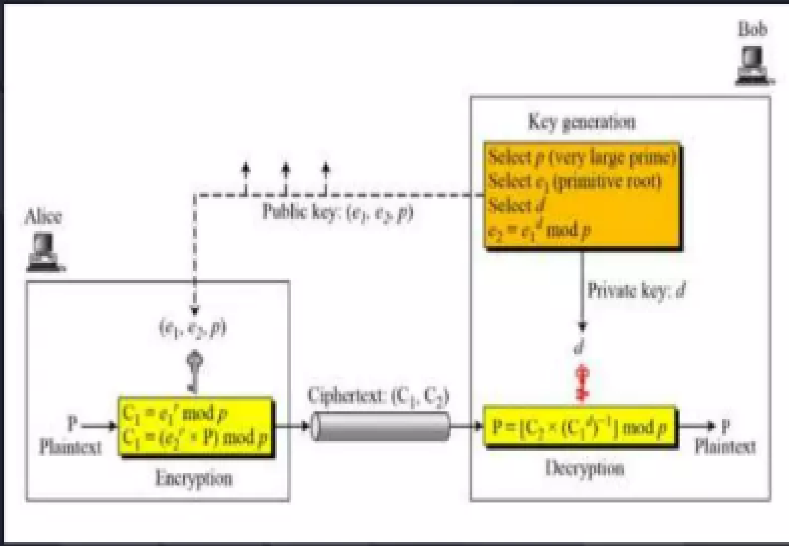
1. Key Generation
2. Encryption
3. Decryption

Step 1 : Key Generation

Receiver Generates public and private keys.

- Select Large Prime No. (P)
- Select Decryption key/ private Key (D) gcd(D,P)=1
- Select Second part of Encryption key or public key (E1) & gcd(E1,P)=1
- Third part of the encryption key or public key (E2)

  $E2 = E1^{D} \bmod P$
- Public Key=(E1, E2, P) & Private key=D



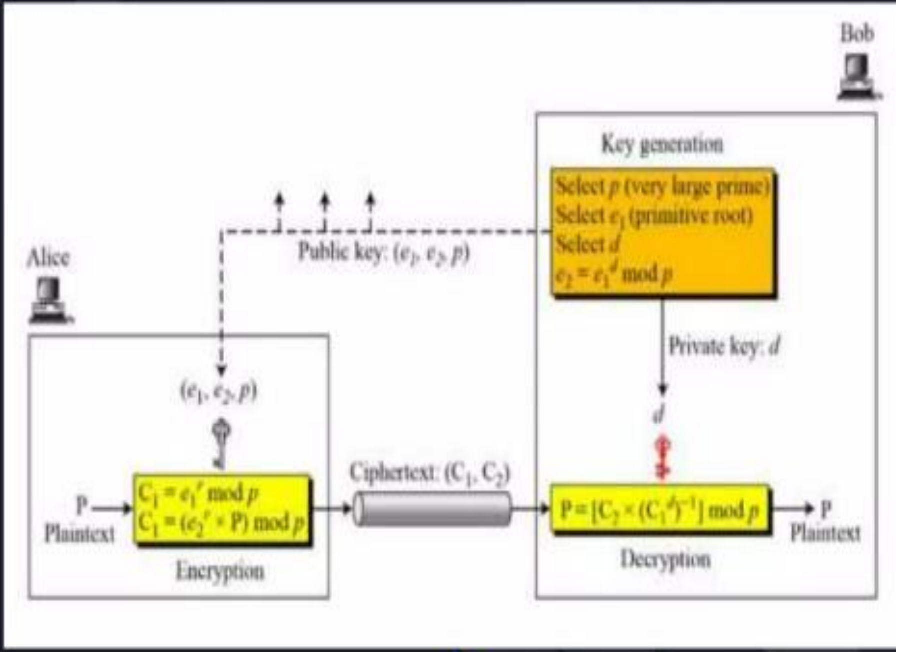
Suppose :

1. P=11 , D=3, E1=2
2. Then $E2= 2^{3}$ mod 11=8
3. Public key=(2, 8, 11) & Private key= 3.

Step 2 : Encryption

Sender Encrypts Data (PT) Using Receiver's Public Key

- Select Random Integer (R)
- C1= $E1^{R}$ mod P
- C2 =(PT X $E2^{R}$) mod P
- C. T. =(C1,C2)



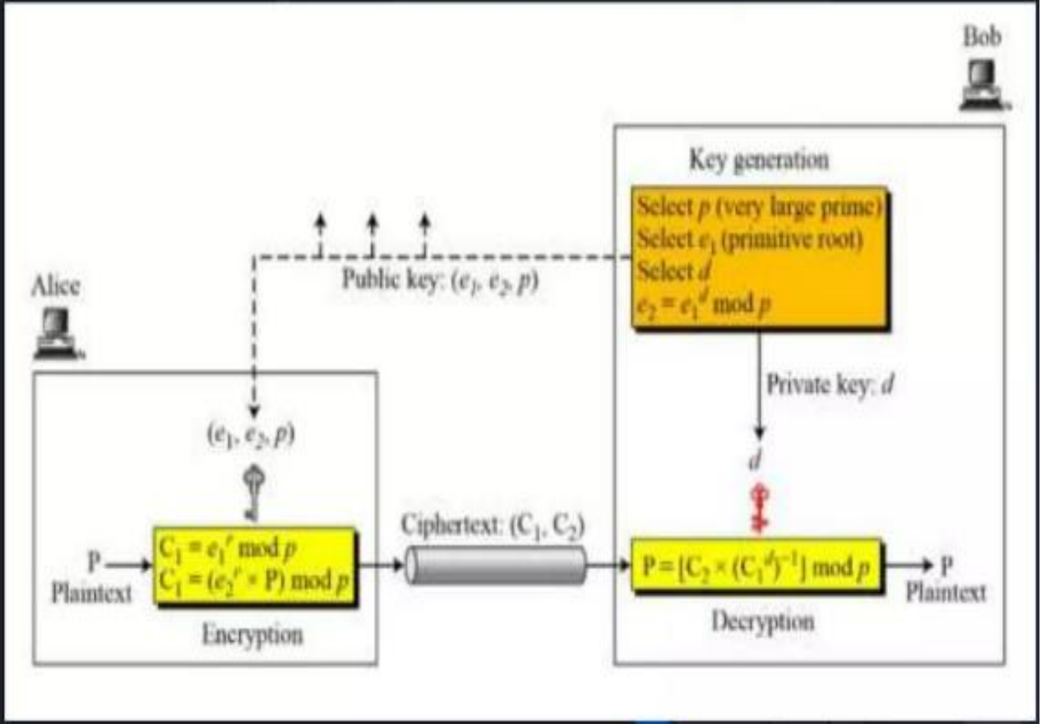
Continuous :-

1. R=4, C1=2^4 mod 11=5, PT=7
2. C2=(7 x 8^4) mod 11=6
3. C.T. = (5,6)

Step 3 : Decryption

Receiver End Decrypts the Message

- PT=[C2 X $(C1^{D})^{-1}$] mod P



Continuous :-

- PT= $(6 \times (5^3)^{-1}) \bmod 11$
  = 18 mod 11 = 7

Message Authentication

- Authentication Requirement
  - Possible attacks on the network
    - Disclosure
    - Traffic analysis
    - Masquerade
    - Content modification
    - Sequence modification
    - Timing modification
    - Source repudiation
    - Destination repudiation

Authentication Functions

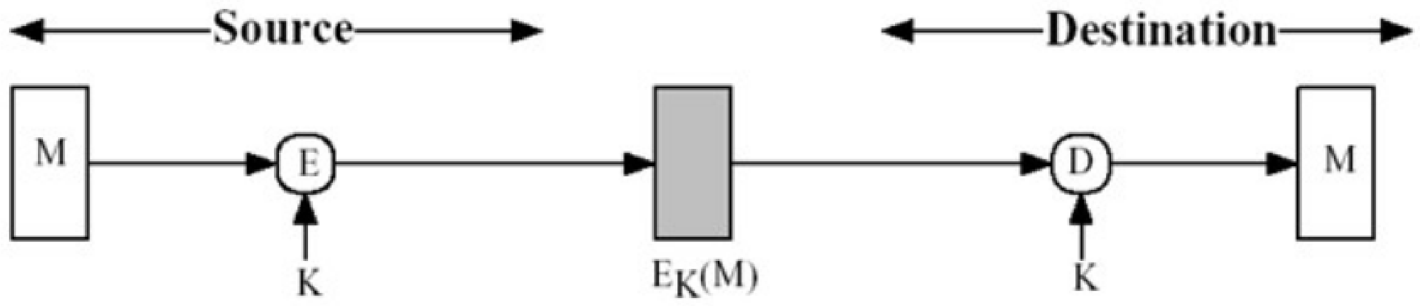
- Message encryption
  - The ciphertext of the entire message serves as its authenticator
- Message authentication code (MAC)
  - A public function of the message and a secret key that produces a fix-length value that serves as the authenticator
- Hash Function
  - A public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

Message Encryption

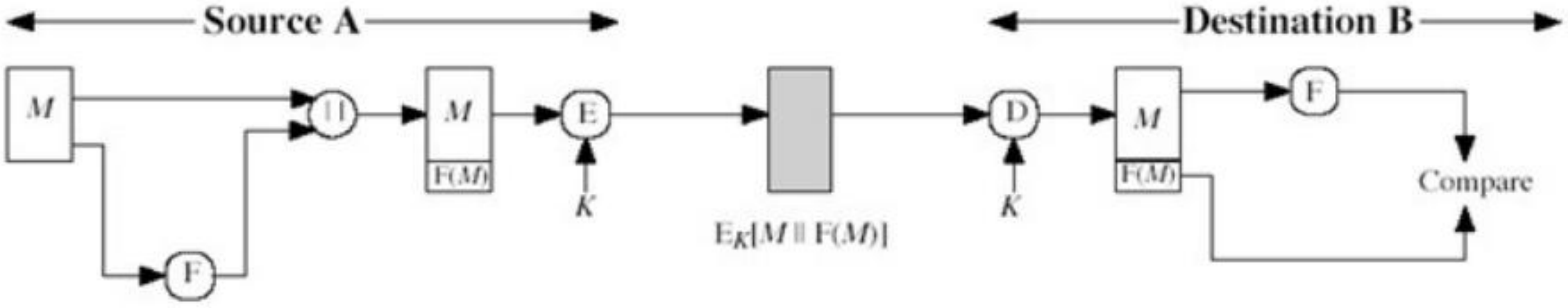
- message encryption by itself also provides a measure of authentication
- if symmetric encryption is used then:
  - since only sender and receiver now key used
  - know content cannot of been altered
  - It does not work if the plaintext is an arbitrary bit pattern
  - if message has suitable structure, redundancy or a checksum to detect any changes

Message Encryption

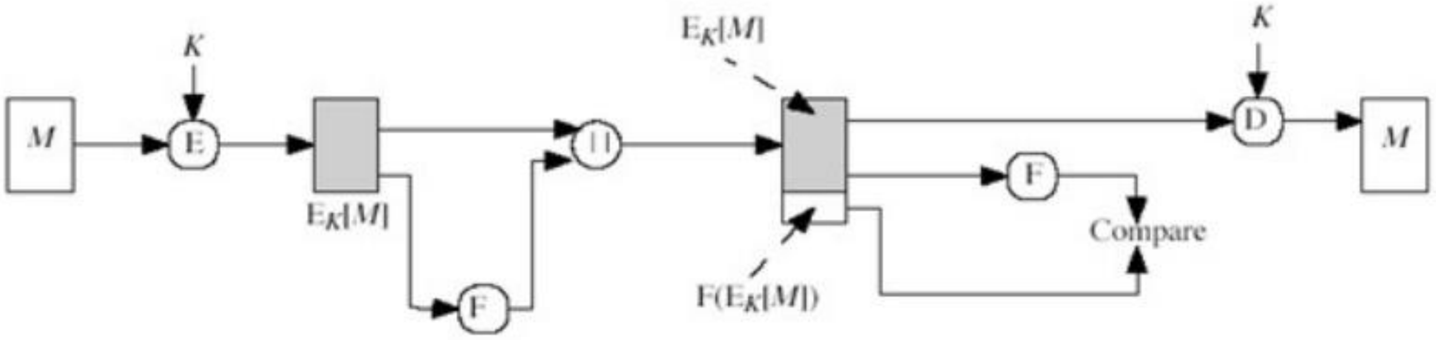
(A)



Conventional encryption: confidentiality and authentication



(a) Internal error control

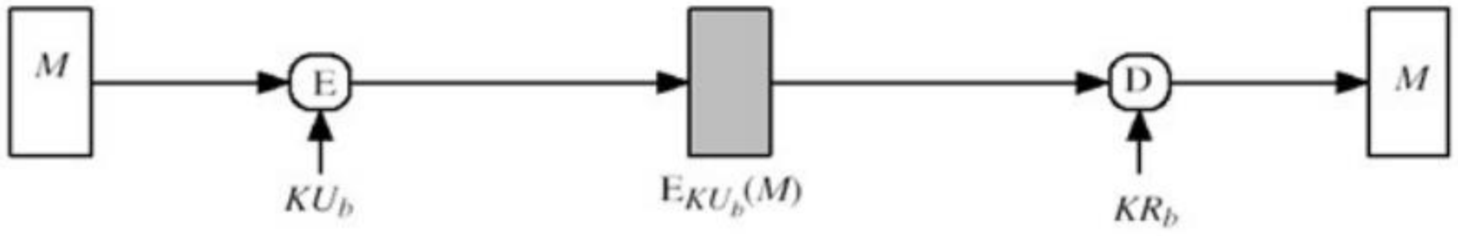


(b) External error control

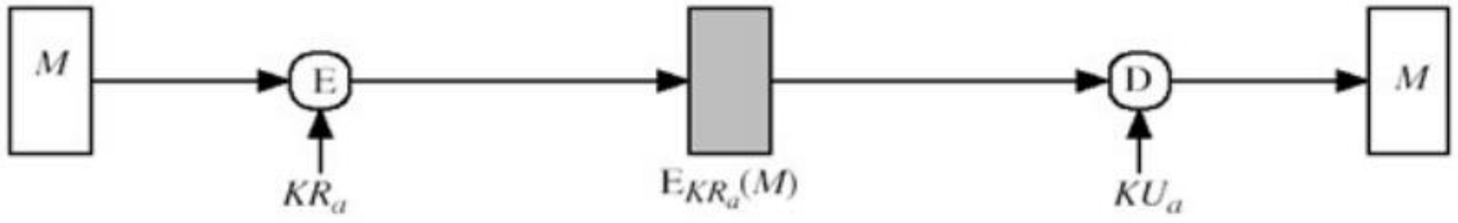
Figure 11.2 Internal and External Error Control

Message Encryption

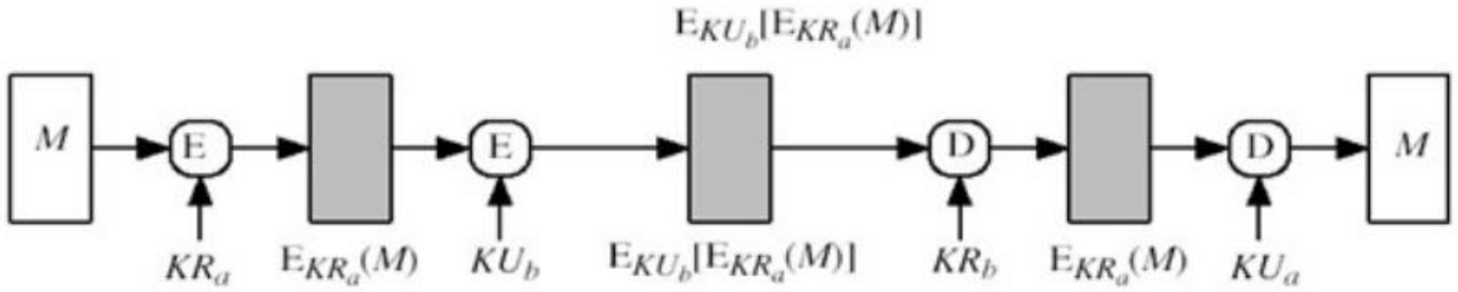
- if public-key encryption is used:
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message



(b) Public-key encryption: confidentiality



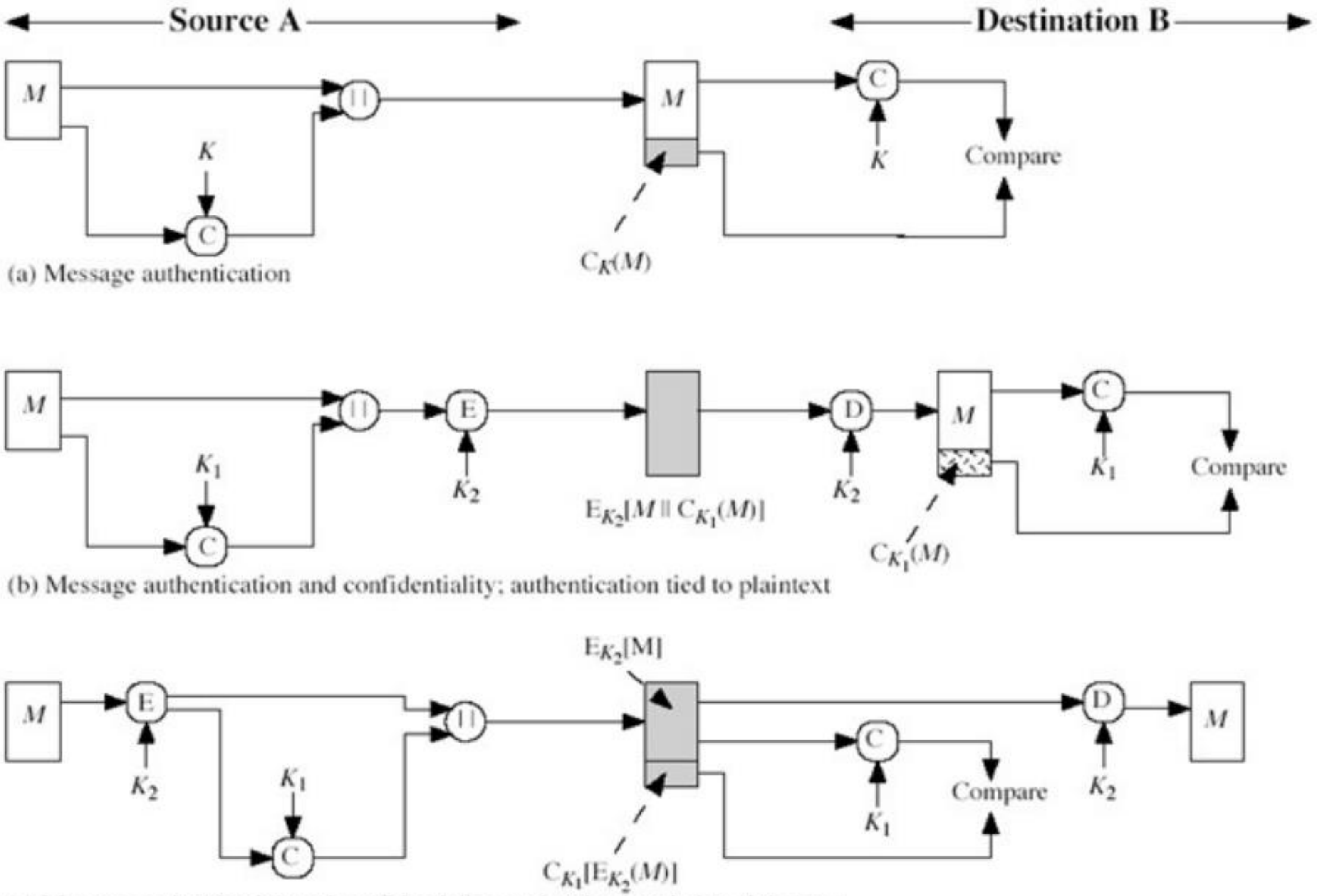
(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key
  - like encryption though need not be reversible
- appended to message as a **signature**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender



Source A
Destination B
M
||
K
C
M
C
K
Compare
$C_K(M)$
(a) Message authentication
M
||
E
K_1
C
K_2
$E_{K_2}[M \| C_{K_1}(M)]$
D
K_2
M
C
K_1
Compare
$C_{K_1}(M)$
(b) Message authentication and confidentiality; authentication tied to plaintext
$E_{K_2}[M]$
M
E
K_2
||
K_1
C
$C_{K_1}[E_{K_2}(M)]$
C
K_1
Compare
D
K_2
M
(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure 11.4 Basic Uses of Message Authentication Code (MAC)

Message Authentication Codes

- the MAC provides integrity & authentication
- can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- why use a MAC?
  - sometimes only integrity is needed
- note that a MAC is not a digital signature

MAC Properties

- a MAC is a cryptographic checksum

  $\mathtt{MAC} = \mathtt{C}_{\mathtt{K}}(\mathtt{M})$

  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- is a many-to-one function
  - potentially many messages have same MAC
  - but finding k to be very difficult

Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
  1. knowing a message and MAC, is infeasible to find another message with same MAC
  2. MACs should be uniformly distributed
  3. MAC should depend equally on all bits of the message

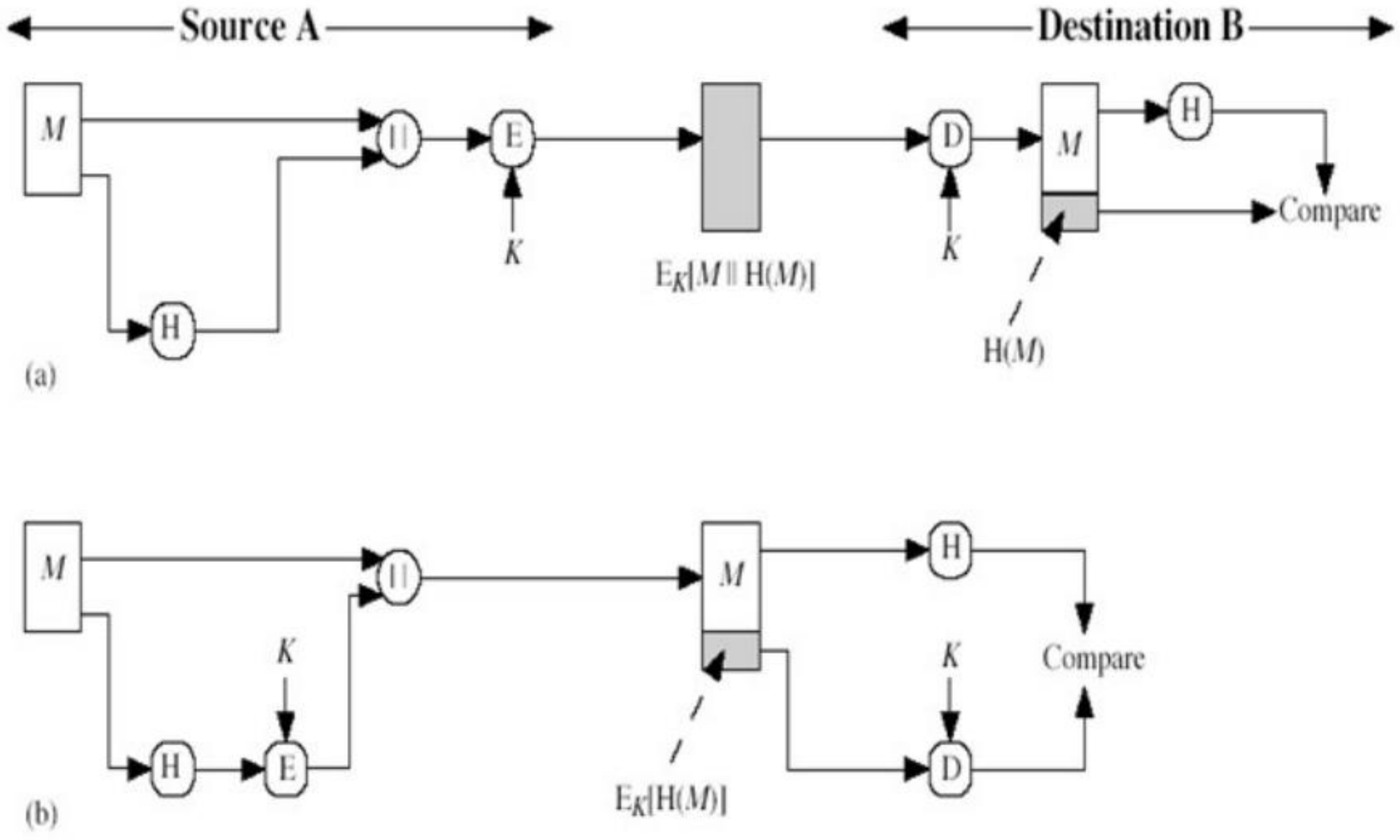
Using Symmetric Ciphers for MACs

- can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    - or the leftmost M bits ($16 \le M \le 64$) of final block
- but final MAC is now too small for security

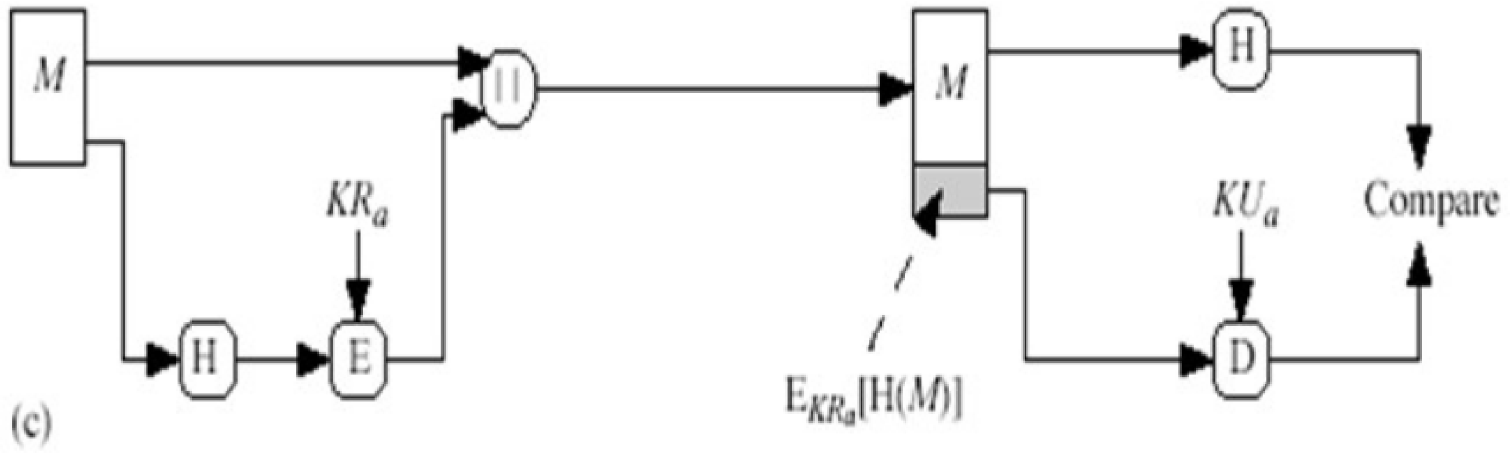
Hash Functions

- condenses arbitrary message to fixed size
- usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create a digital signature

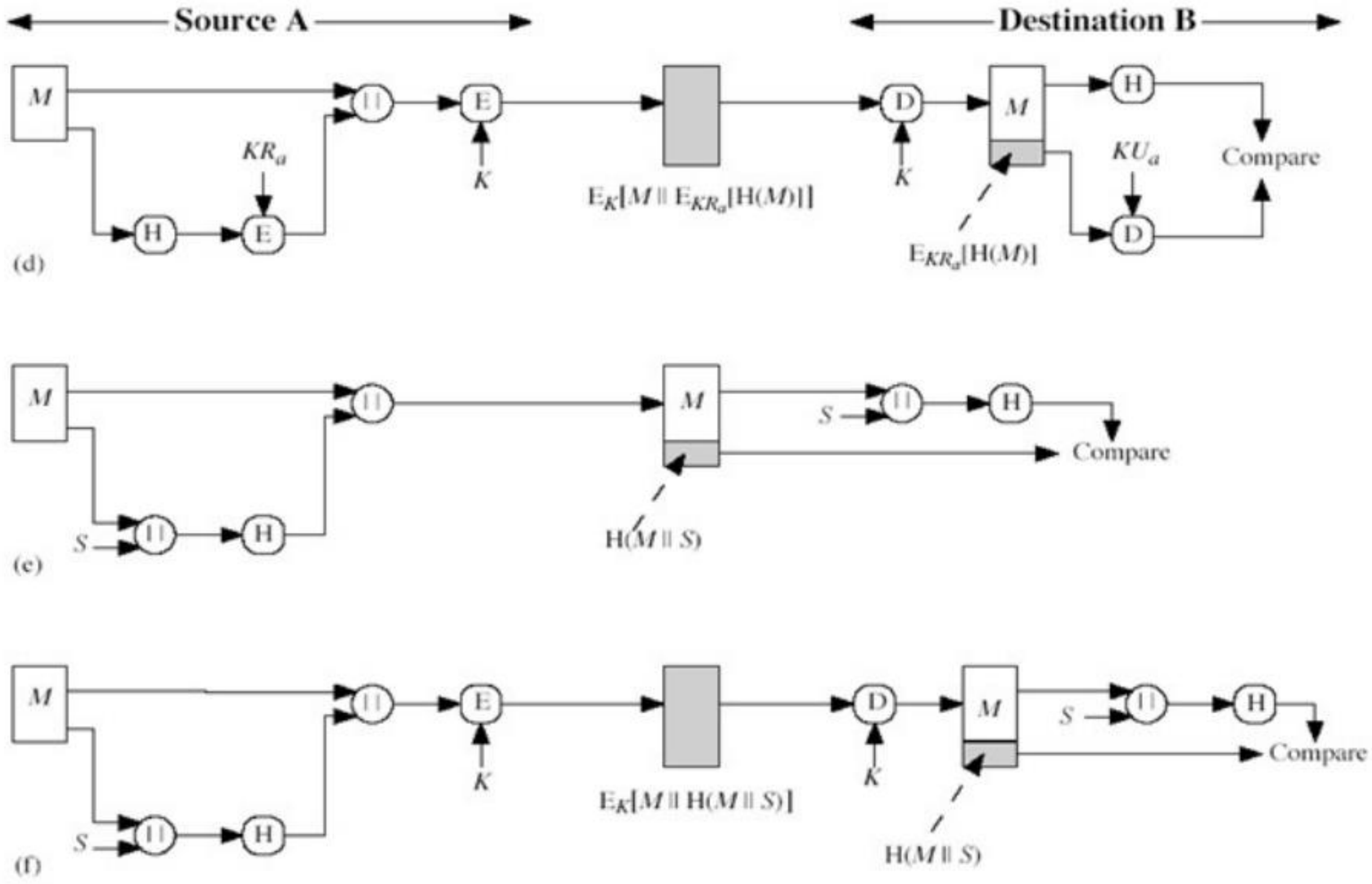
Basic Uses of Hash Function



Hash Functions & Digital Signatures



Basic Uses of Hash Function



Hash Function Properties

- a Hash Function produces a fingerprint of some file/message/data

  `h = H(M)`

  - condenses a variable-length message M
  - to a fixed-sized fingerprint
- assumed to be public

Requirements for Hash Functions

1. can be applied to any sized message `M`
2. produces fixed-length output `h`
3. is easy to compute `h=H(M)` for any message `M`
4. given `h` is infeasible to find `x` s.t. `H(x)=h`
   - one-way property
5. given `x` is infeasible to find `y` s.t. `H(y)=H(x)`
   - weak collision resistance
6. is infeasible to find any `x,y` s.t. `H(y)=H(x)`
   - strong collision resistance

Simple Hash Functions

- are several proposals for simple functions
- based on XOR of message blocks
- not secure since can manipulate any message and either not change hash or change hash also
- need a stronger cryptographic function (next chapter)

Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MACs

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using $H_0=0$ and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to “meet-in-the-middle” attack
- other variants also susceptible to attack

Hash Functions & MAC Security

- like block ciphers have:
- **brute-force** attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security

Hash Functions & MAC Security

- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- have a number of analytic attacks on iterated hash functions
  - $CV_i = f[CV_{i-1}, M_i]; H(M)=CV_N$
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions