# Review Questions Instructions

# Conventional Encryption Principles

- An encryption scheme has five ingredients:
  - Plaintext
  - Encryption algorithm
  - Secret Key
  - Ciphertext
  - Decryption algorithm
- Security depends on the secrecy of the key, not the secrecy of the algorithm

# Conventional Encryption Principles

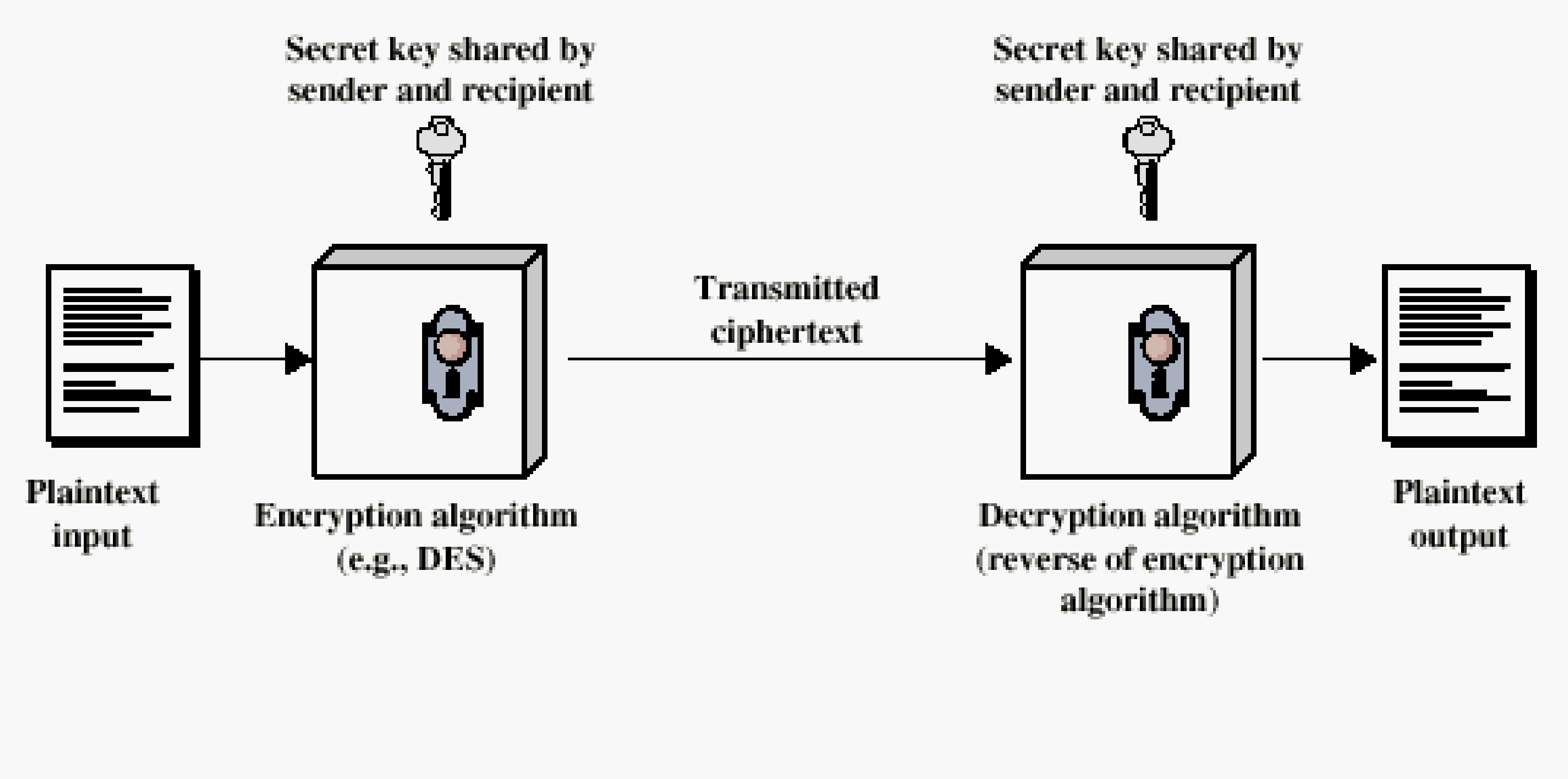

Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 2.1 Simplified Model of Conventional Encryption

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations be reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.
2. **The number of keys used.** If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver each uses a different key, the system is referred to as asymmetric, two-key, or public-key encryption.
3. **The way in which the plaintext is processed.** A *block cipher* processes the input one block of elements at a time, producing an output block for each input block. A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along.

# Triple DEA

- Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)

$$\boxed{C = E_{K3}[D_{K2}[E_{K1}[P]]]}$$

  - C = ciphertext
  - P = Plaintext
  - EK[X] = encryption of X using key K
  - DK[Y] = decryption of Y using key K

- Effective key length of 168 bits

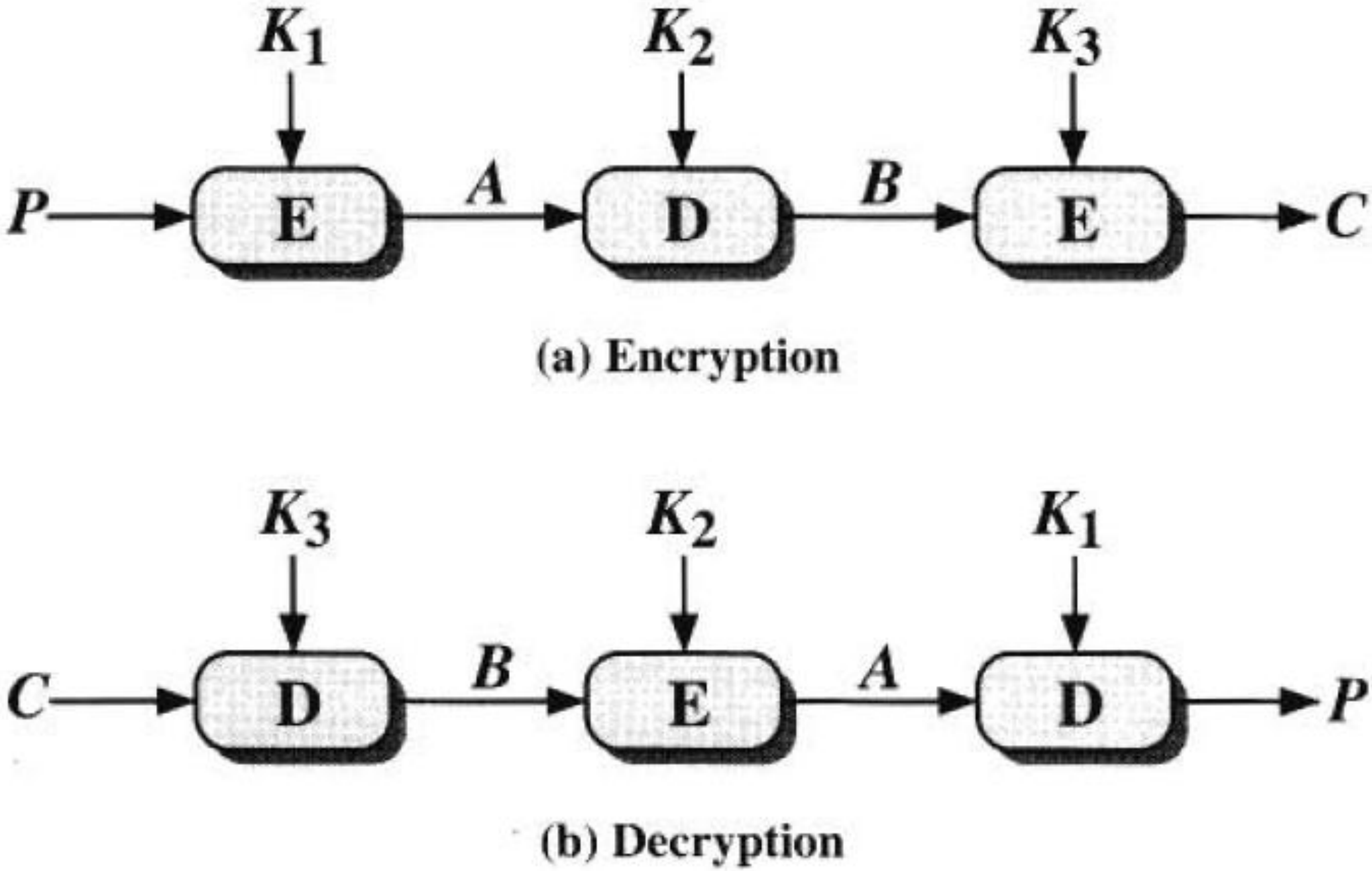


**Figure 2.4** Triple DES

# Cipher Block Modes of Operation

Cipher Block Chaining Mode (CBC)

The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block.

Repeating pattern of 64-bits are not exposed

$$C_i = E_k[C_{i-1} \oplus P_i]$$

$$D_K[C_i] = D_K[E_K(C_{i-1} \oplus P_i)]$$

$$D_K[C_i] = (C_{i-1} \oplus P_i)$$

$$C_{i-1} \oplus D_K[C_i] = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

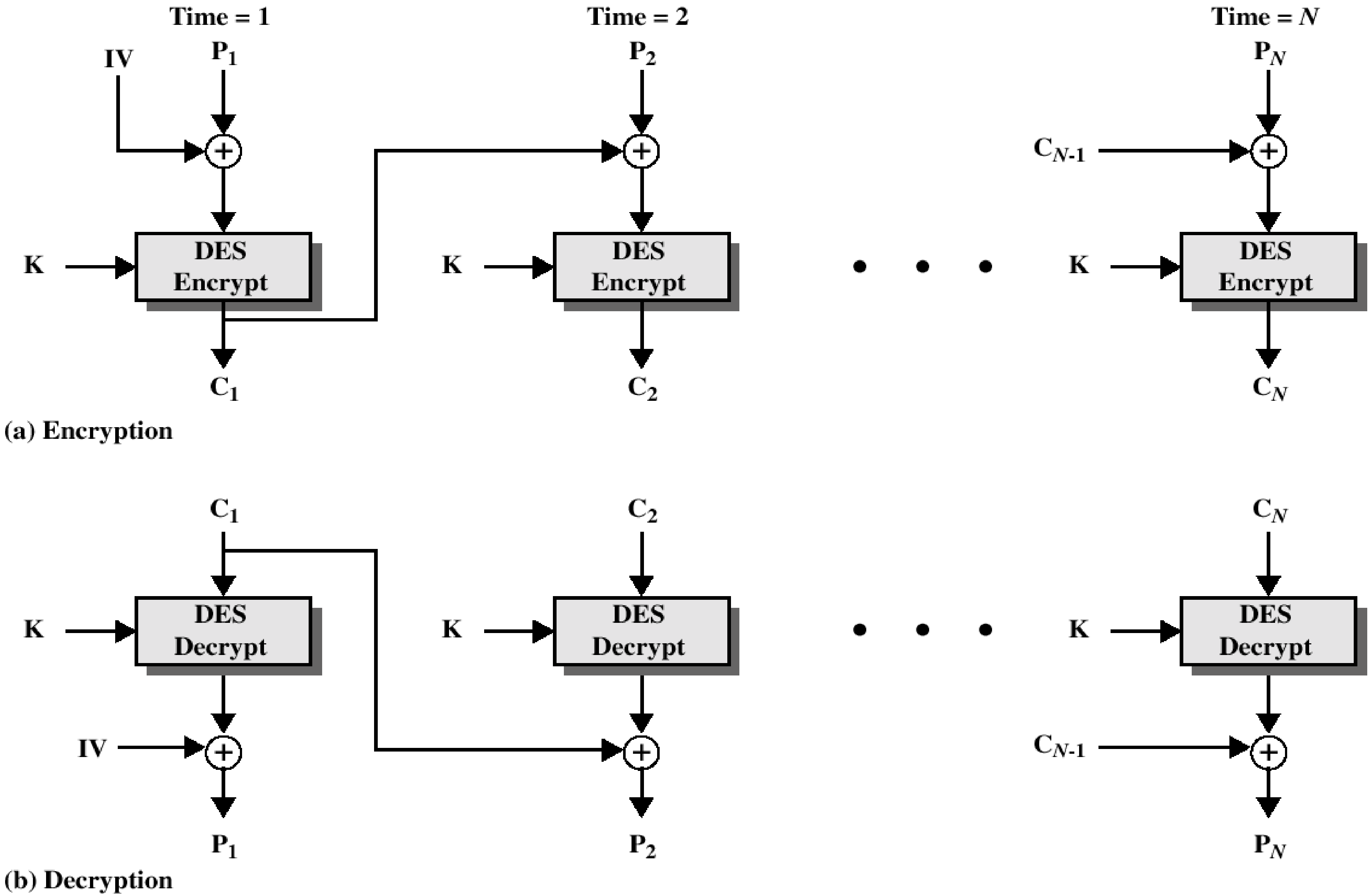


Figure 2.7 Cipher Block Chaining (CBC) Mode

# Key Distribution

1. A key could be selected by A and physically delivered to B.
2. A third party could select the key and physically deliver it to A and B.
3. If A and B have previously used a key, one party could transmit the new key to the other, encrypted using the old key.
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

# Key Distribution (See Figure 2.10)

- **Session key:**
  - Data encrypted with a one-time session key.At the conclusion of the session the key is destroyed
- **Permanent key:**
  - Used between entities for the purpose of distributing session keys

1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted

FEP = front end processor
KDC = key distribution center

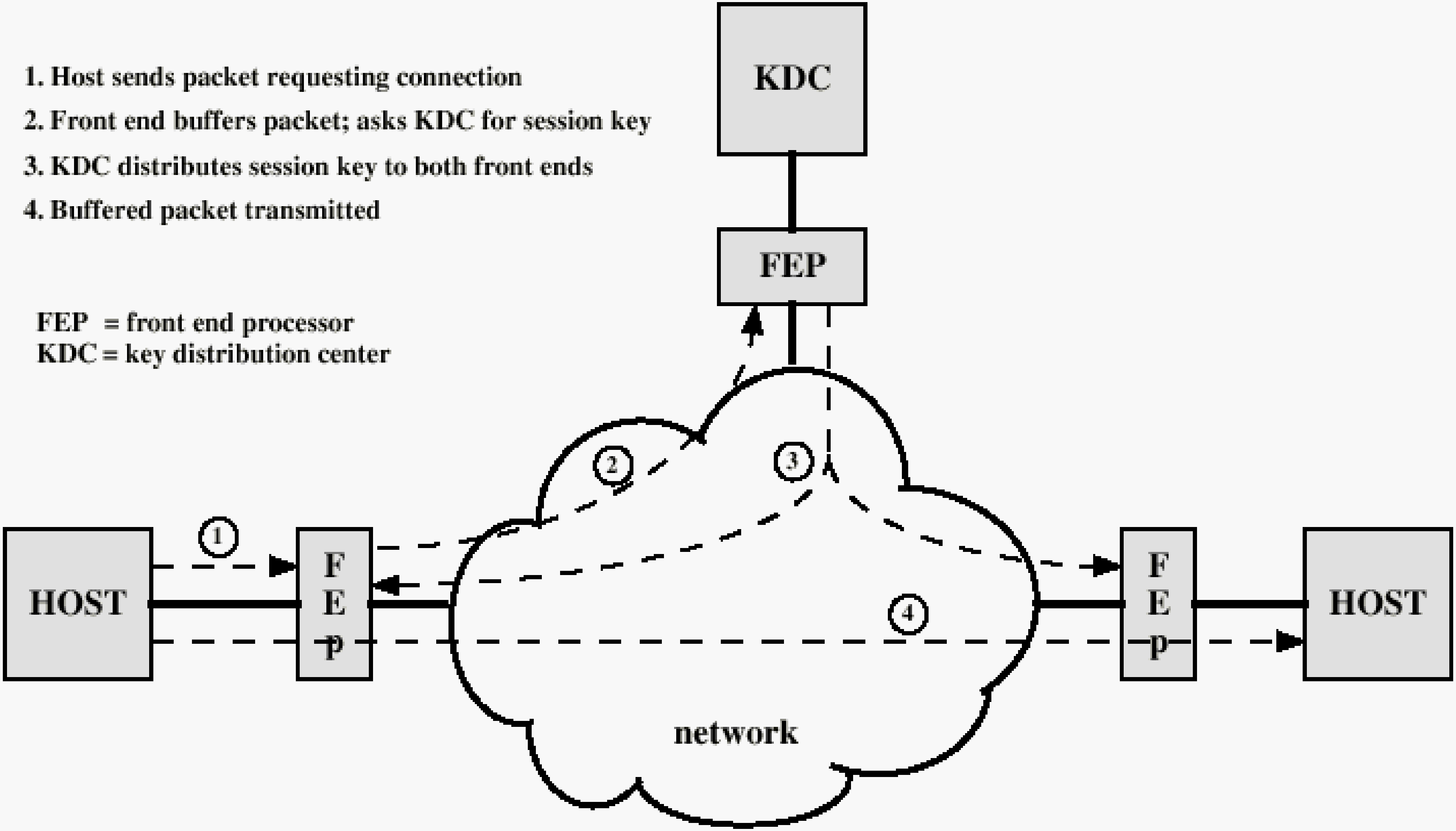


Figure 2.10 Automatic Key Distribution for Connection-Oriented Protocol

The configuration consists of the following elements:

- **Key distribution center:** The key distribution center (KDC) determines which systems are allowed to communicate with each other. When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.
- **Front-end processor:** The front-end processor (FEP) performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

# Authentication

Requirements - must be able to verify that:

1. Message came from apparent source or author,
2. Contents have not been altered,
3. Sometimes, it was sent at a certain time or sequence.

Protection against active attack (falsification of data and transactions)

# Approaches to Message Authentication

- Authentication Using Conventional Encryption
  - Only the sender and receiver should share a key
- Message Authentication without Message Encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code
  - Calculate the MAC as a function of the message and the key. MAC = F(K, M)

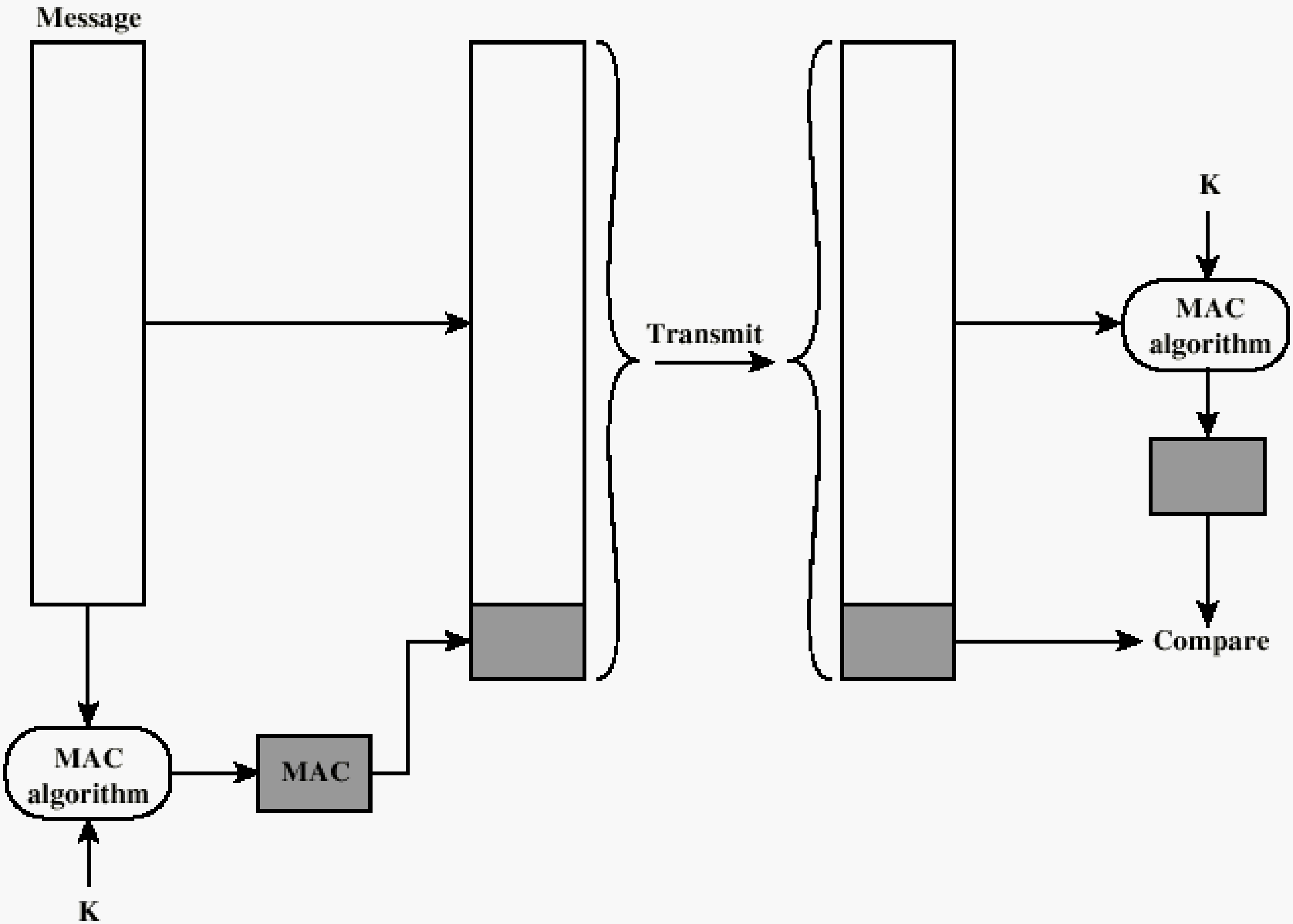
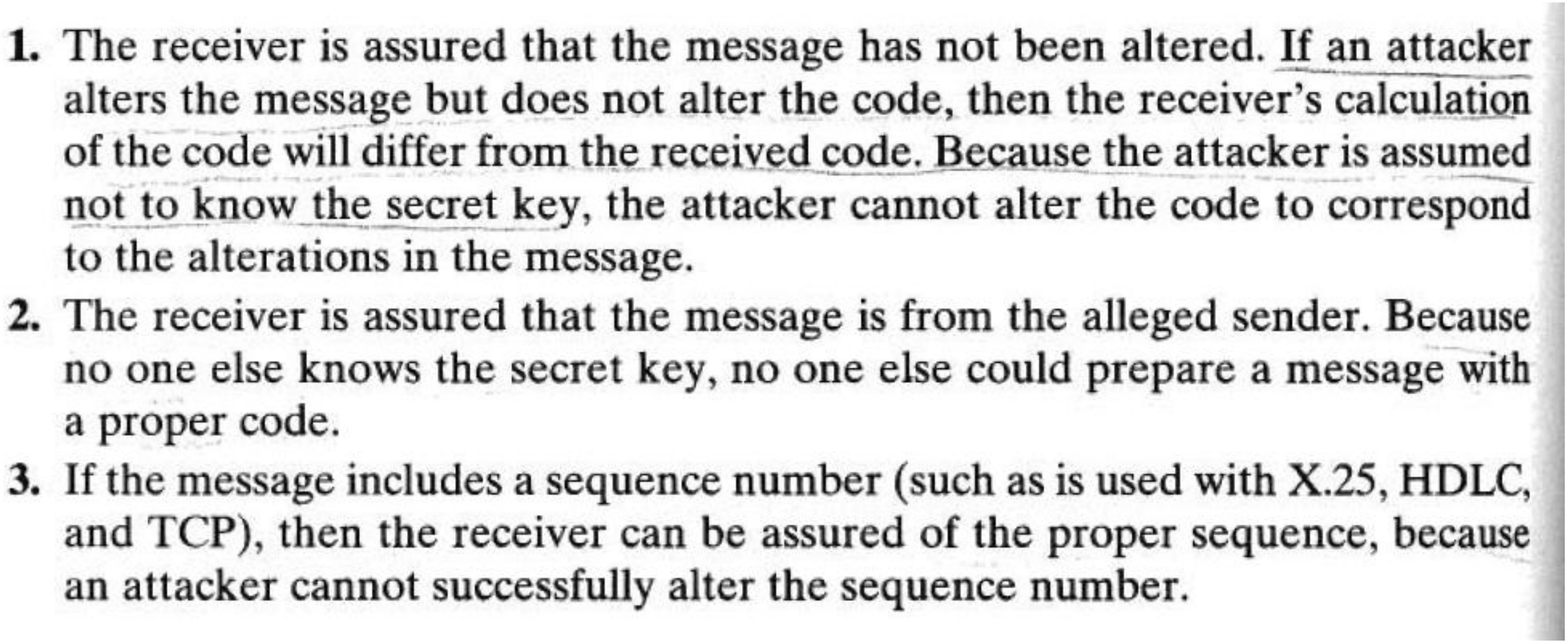
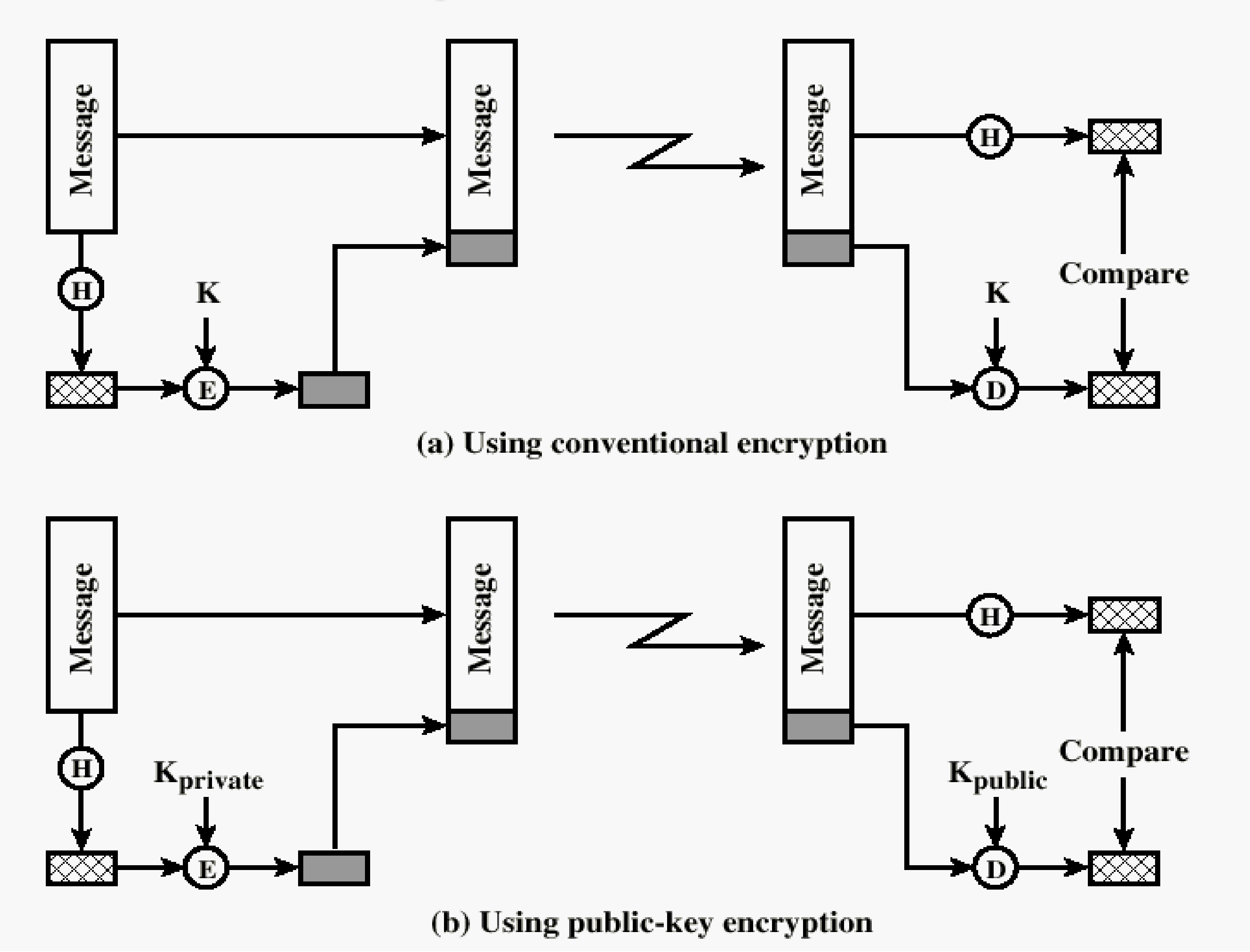


**Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)**



1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

# One-way HASH function

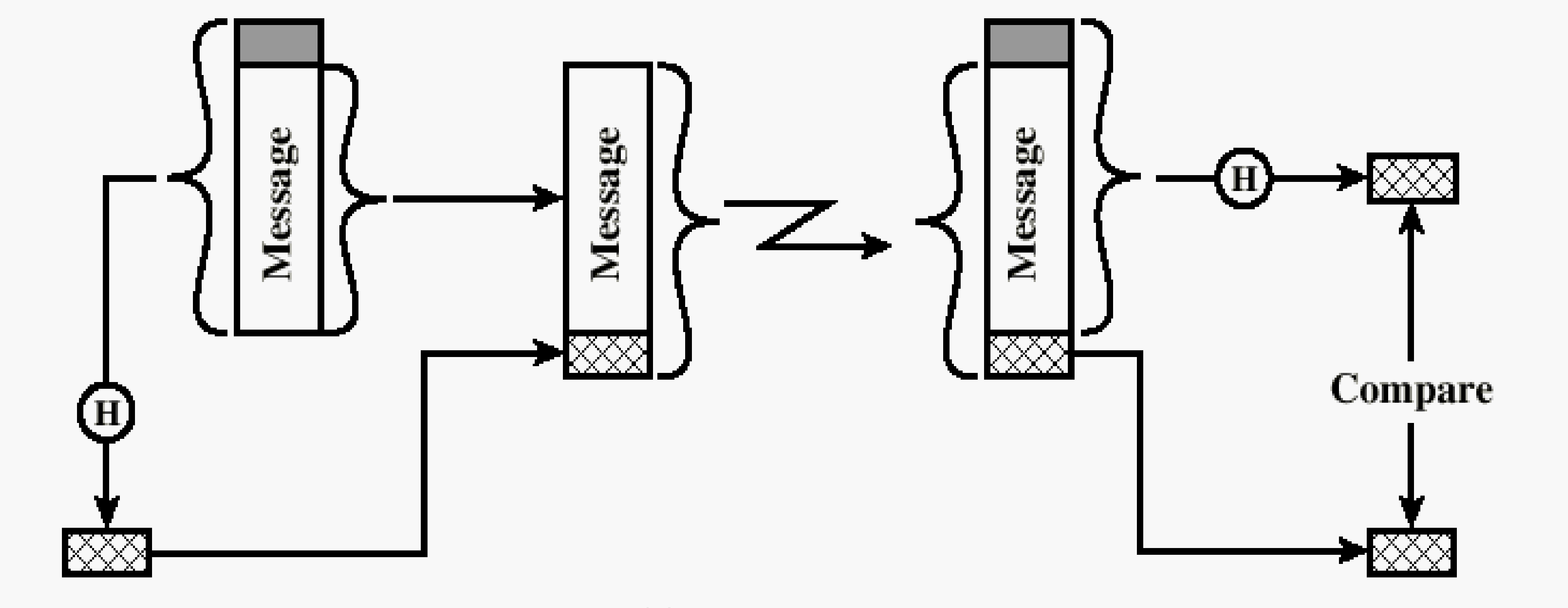


(a) Using conventional encryption

(b) Using public-key encryption

# One-way HASH function

Secret value is added before the hash and removed before transmission.



(c) Using secret value

# Secure HASH Functions

Purpose of the HASH function is to produce a "fingerprint.

Properties of a HASH function H :

1. H can be applied to a block of data at any size
2. H produces a fixed length output
3. H(x) is easy to compute for any given x.
4. For any given block x, it is computationally infeasible to find x such that H(x) = h
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

# Public-Key Cryptography Principles

The use of two keys has consequences in: key distribution, confidentiality and authentication.

The scheme has six ingredients (see Figure 3.7)

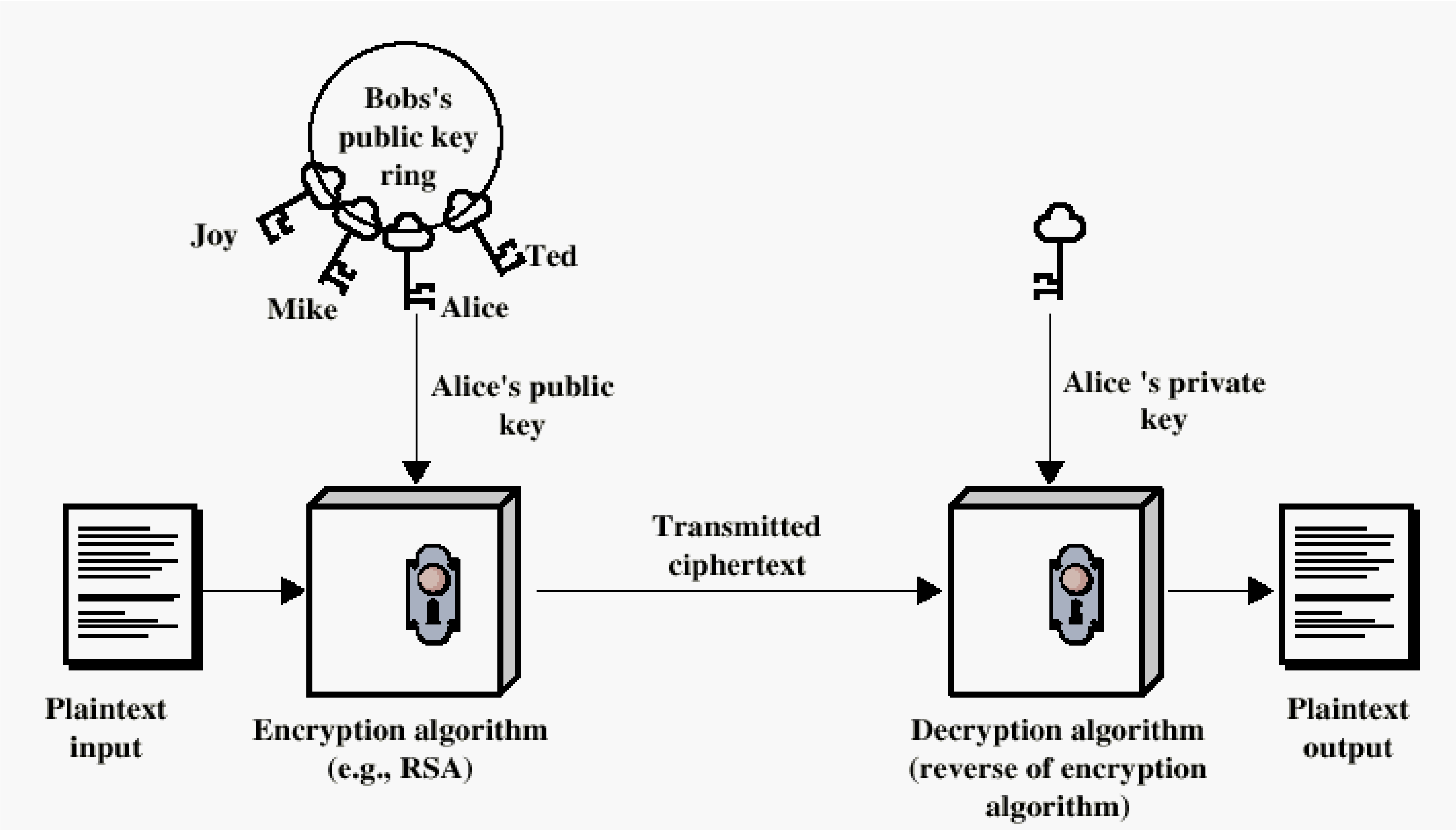
- Plaintext
- Encryption algorithm
- Public and private key
- Ciphertext
- Decryption algorithm

# Applications for Public-Key Cryptosystems

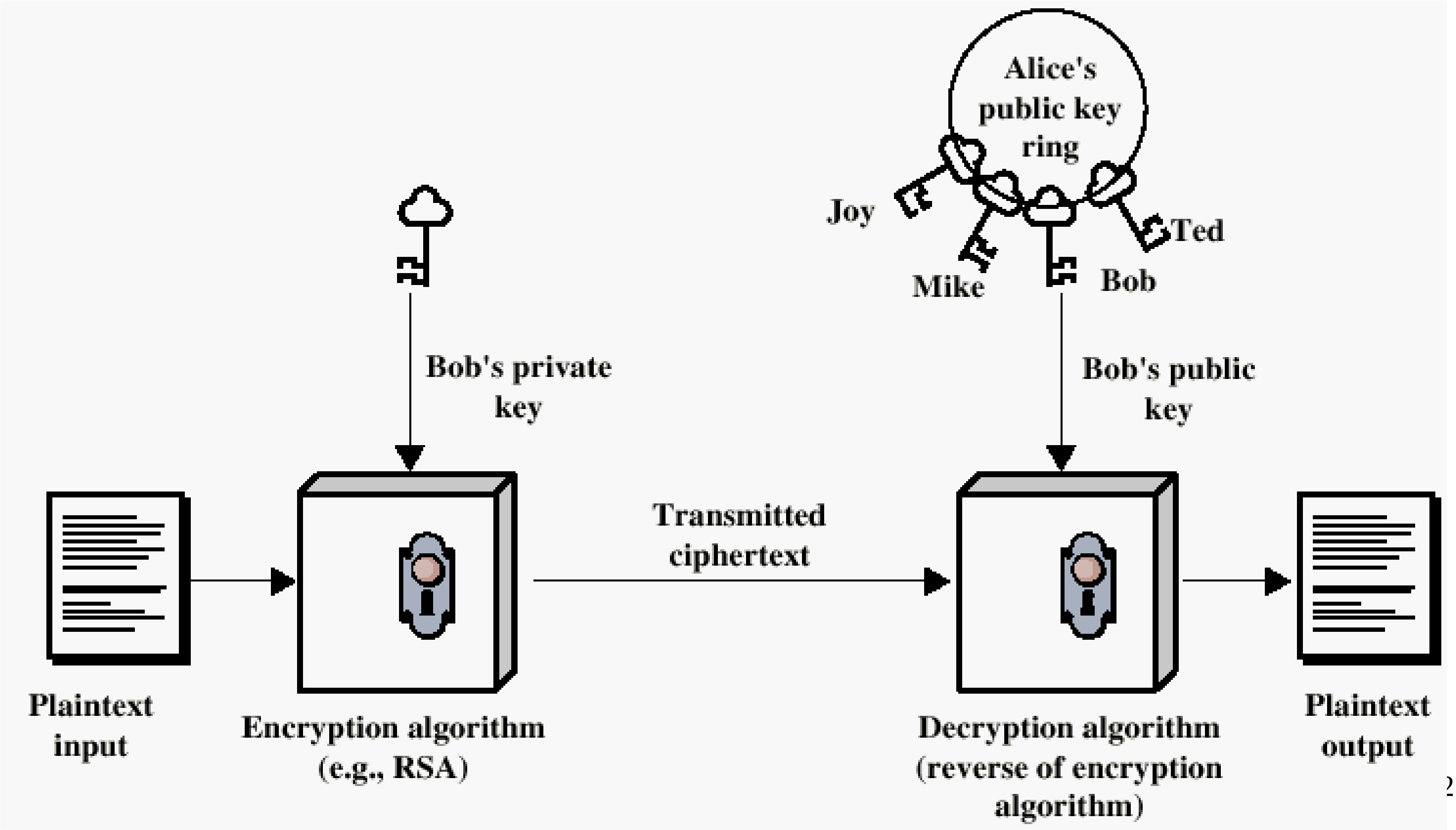
Three categories:

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key.
- **Key echange:** Two sides cooperate two exhange a session key.

# Encryption using Public-Key system



# Authentication using Public-Key System



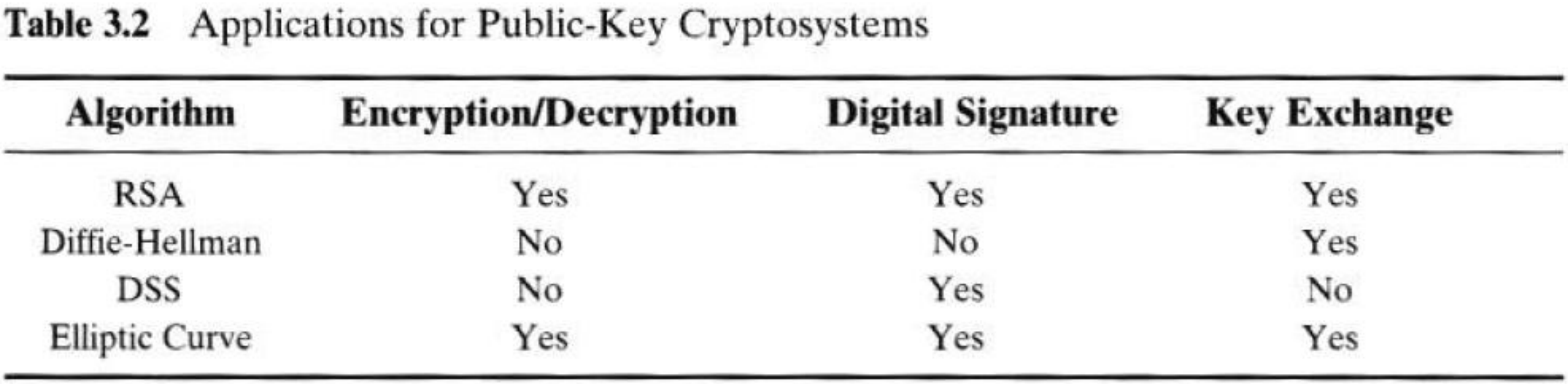
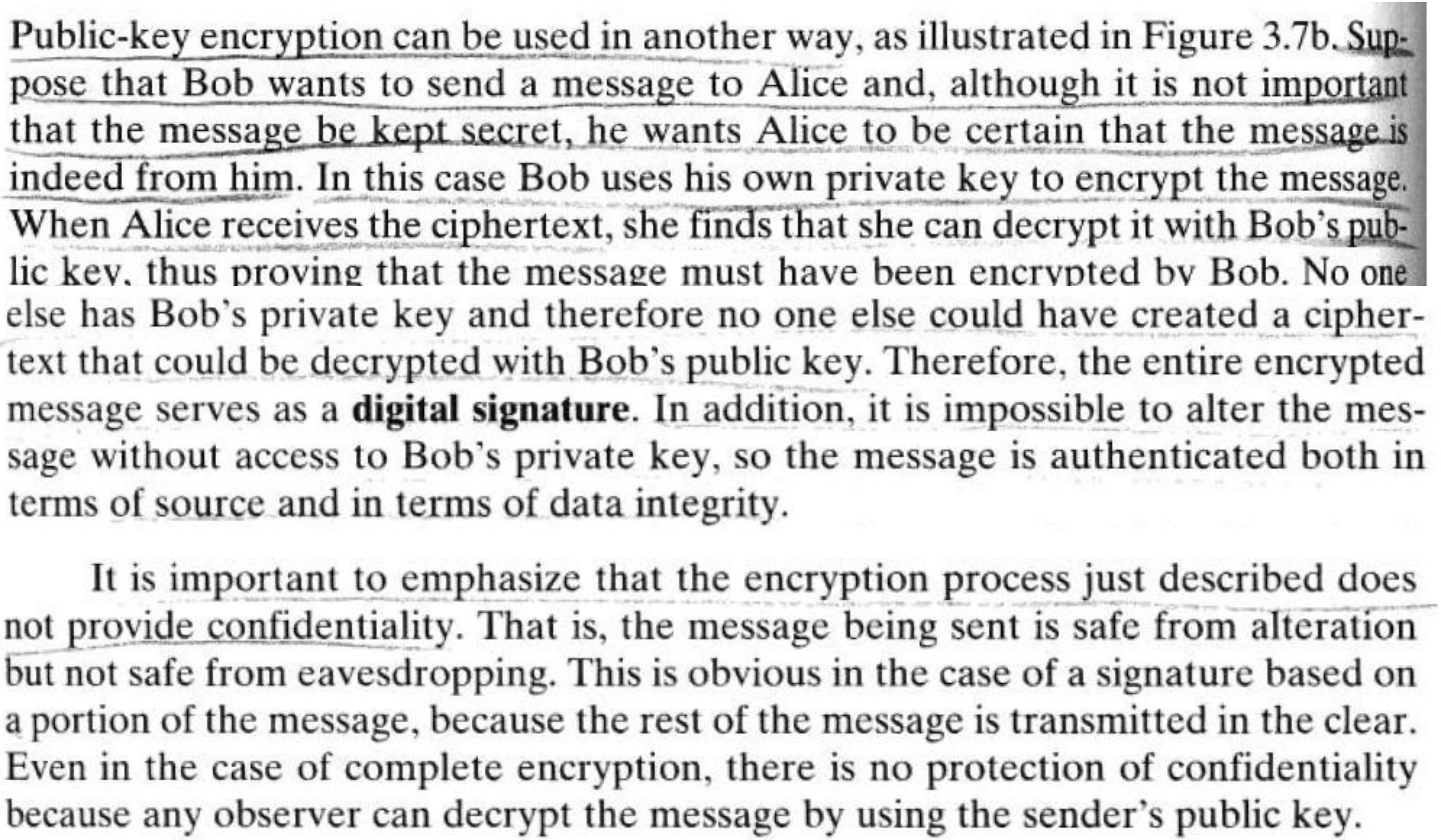


**Table 3.2** Applications for Public-Key Cryptosystems

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

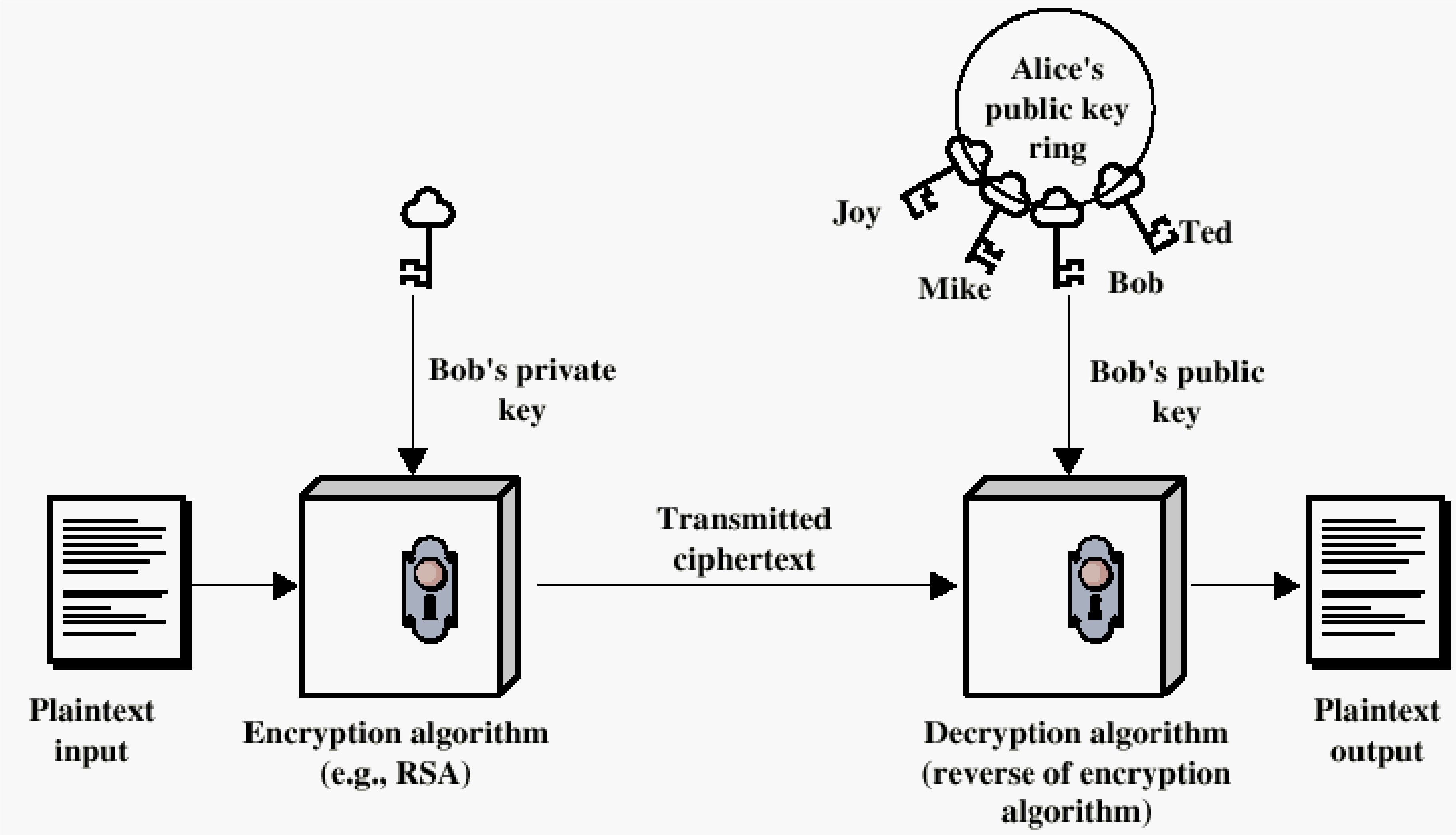
# Digital Signature

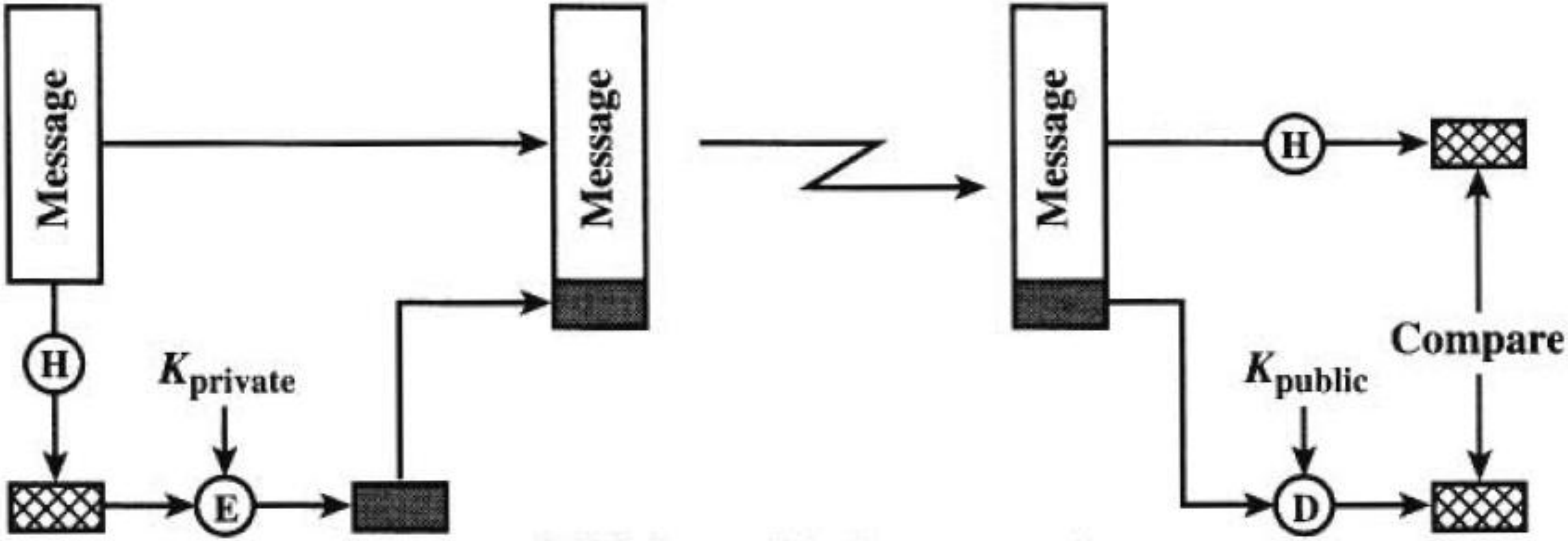


Public-key encryption can be used in another way, as illustrated in Figure 3.7b. Suppose that Bob wants to send a message to Alice and, although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In this case Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob's public key, thus proving that the message must have been encrypted by Bob. No one else has Bob's private key and therefore no one else could have created a ciphertext that could be decrypted with Bob's public key. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

It is important to emphasize that the encryption process just described does not provide confidentiality. That is, the message being sent is safe from alteration but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

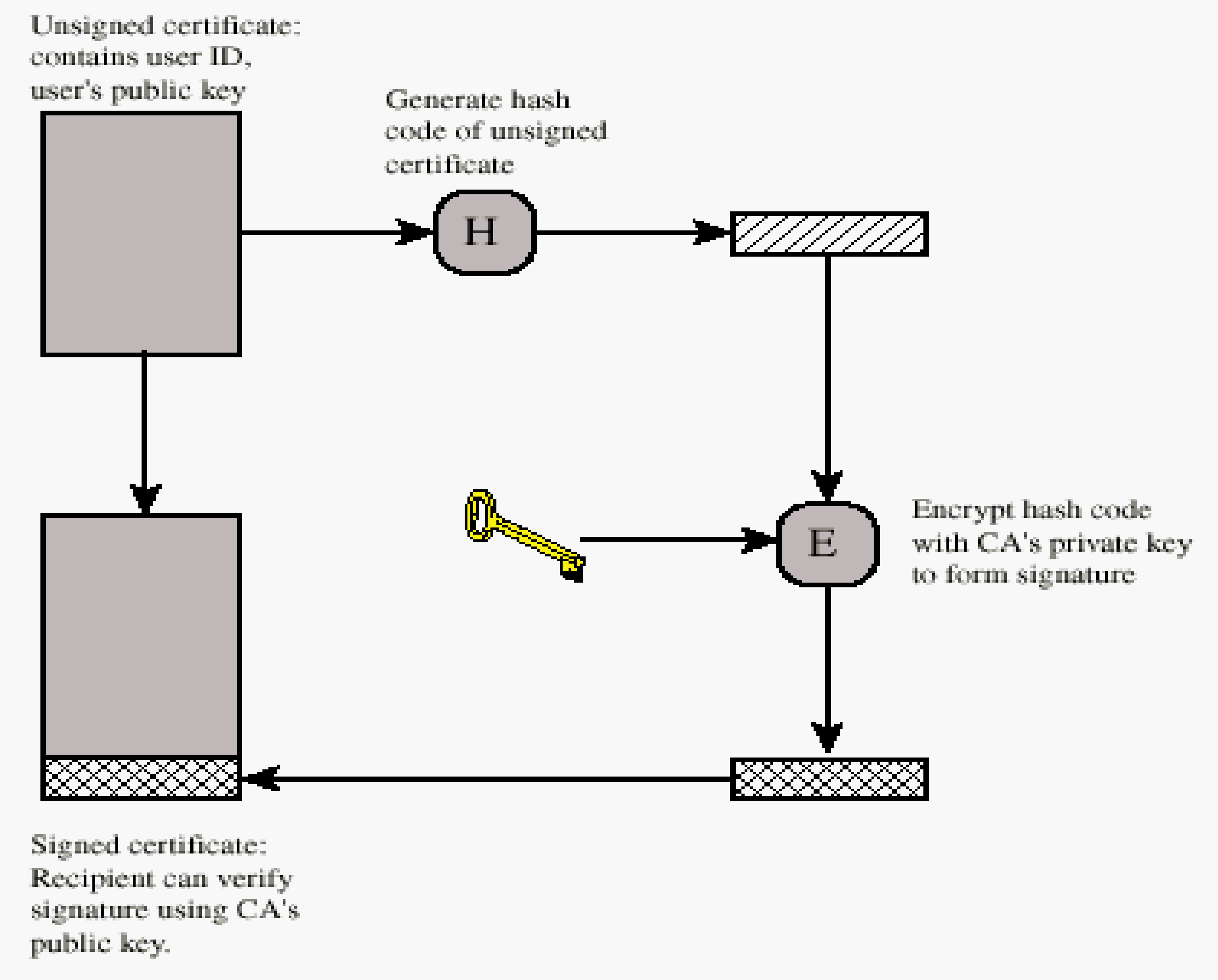
# Authentication using Public-Key System

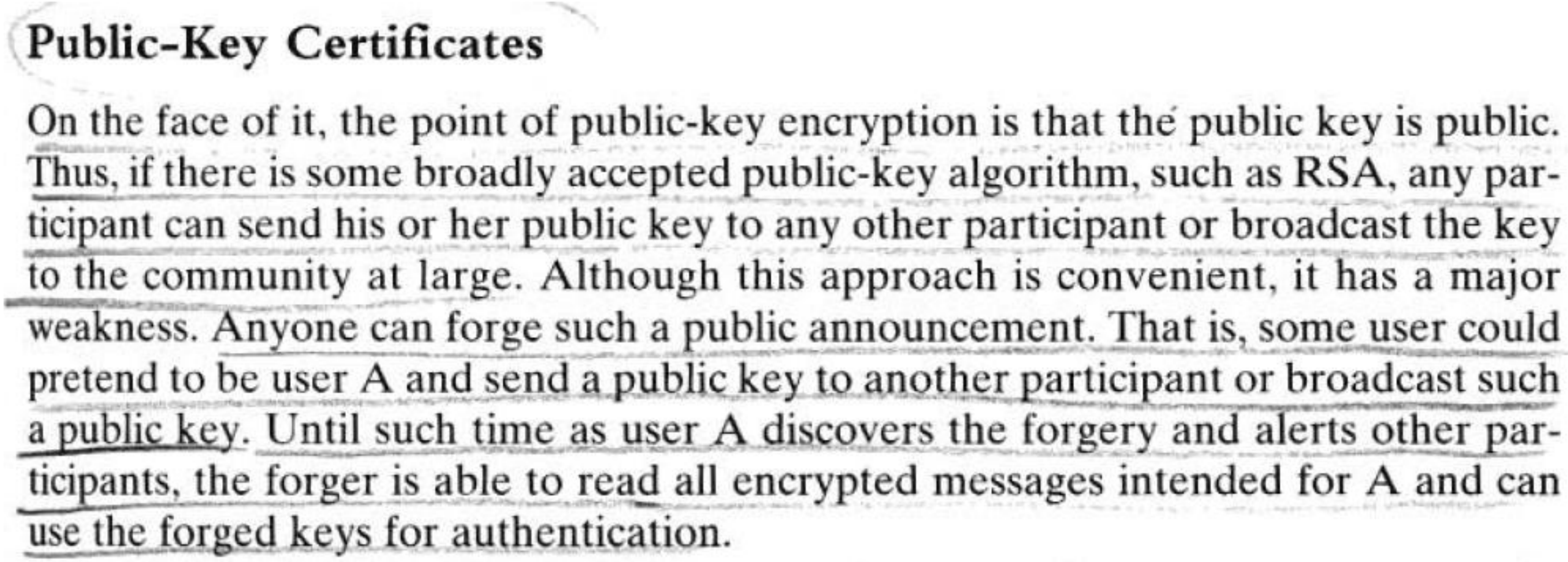




(b) Using public-key encryption

# Key Management
# Public-Key Certificate Use





## Public-Key Certificates

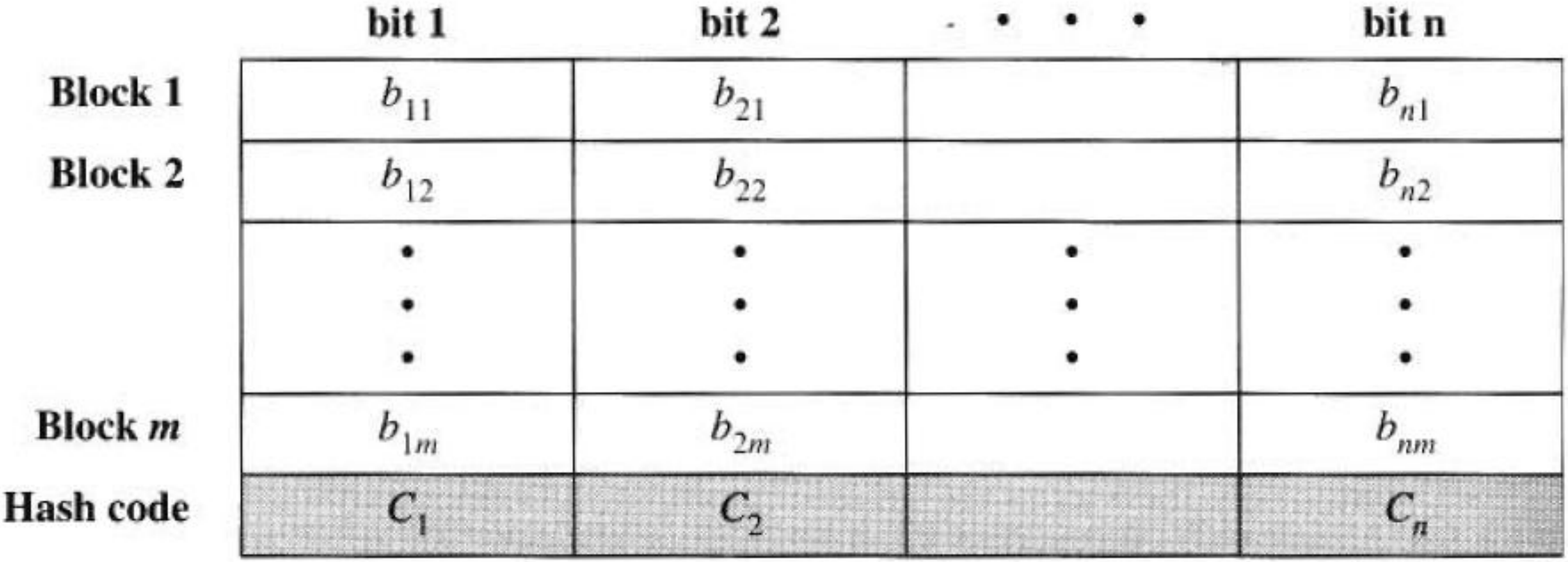
On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large. Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

One approach is the use of Diffie-Hellman key exchange. This approach is indeed widely used. However, it suffers the drawback that, in its simplest form, Diffie-Hellman provides no authentication of the two communicating partners.

A powerful alternative is the use of public-key certificates. When Bob wishes to communicate with Alice, Bob can do the following:

1. Prepare a message.
2. Encrypt that message using conventional encryption with a one-time conventional session key.
3. Encrypt the session key using public-key encryption with Alice's public key.
4. Attach the encrypted session key to the message and send it to Alice.

Only Alice is capable of decrypting the session key and therefore of recovering the original message. If Bob obtained Alice's public key by means of Alice's public-key certificate, then Bob is assured that it is a valid key.



|  | bit 1 | bit 2 | · · · · | bit n |
|---|---|---|---|---|
| **Block 1** | $b_{11}$ | $b_{21}$ |  | $b_{n1}$ |
| **Block 2** | $b_{12}$ | $b_{22}$ |  | $b_{n2}$ |
|  | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| **Block *m*** | $b_{1m}$ | $b_{2m}$ |  | $b_{nm}$ |
| **Hash code** | $C_1$ | $C_2$ |  | $C_n$ |

**Figure 3.3** Simple Hash Function Using Bitwise XOR

# The RSA Algorithm – Key Generation

1. Select *p,q* — *p* and *q* both prime
2. Calculate $n = p \times q$
3. Calculate $\Phi(n) = (p-1)(q-1)$
4. Select integer *e* — $\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$
5. Calculate *d* — $d = e^{-1} \bmod \Phi(n)$
6. Public Key — KU = {e,n}
7. Private key — KR = {d,n}

# Example of RSA Algorithm

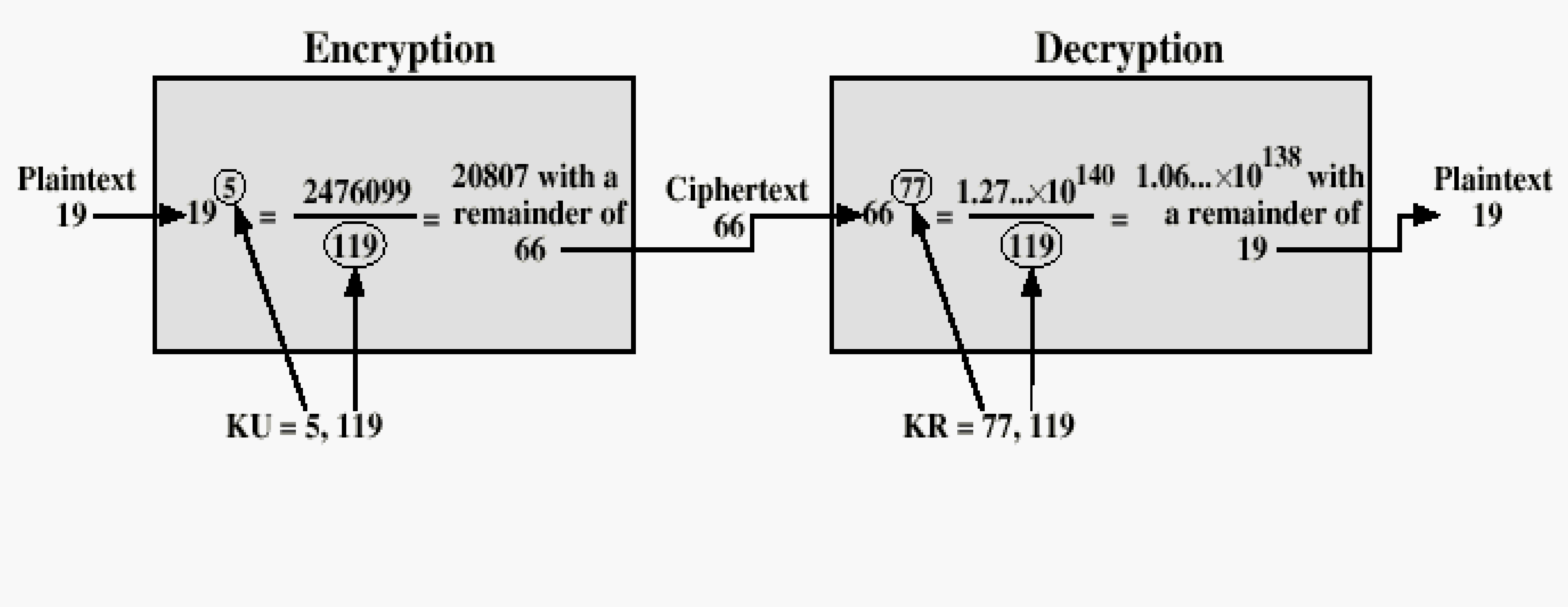
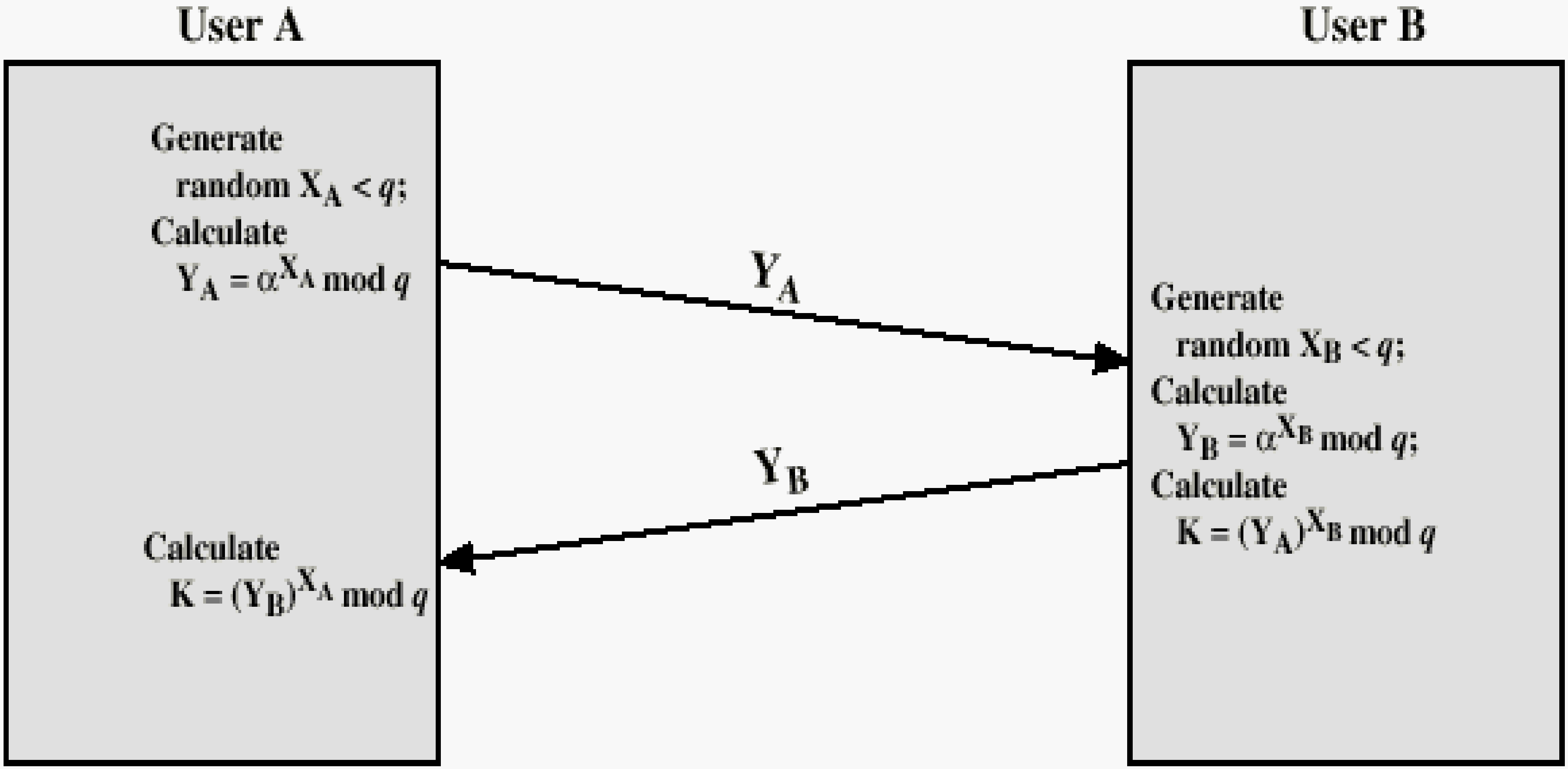

Encryption
Decryption
Plaintext 19
$19^{5} = \frac{2476099}{119} =$ 20807 with a remainder of 66
Ciphertext 66
$66^{77} = \frac{1.27\ldots\times 10^{140}}{119} = 1.06\ldots\times 10^{138}$ with a remainder of 19
Plaintext 19
KU = 5, 119
KR = 77, 119

**Figure 3.9 Example of RSA Algorithm**

# Diffie-Hellman Key Echange



User A:

Generate random $X_A < q$;

Calculate $Y_A = \alpha^{X_A} \bmod q$

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

User B:

Generate random $X_B < q$;

Calculate $Y_B = \alpha^{X_B} \bmod q$;

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

# KERBEROS

Provides a centralized authentication server to authenticate users to servers and servers to users.

Relies on conventional encryption, making no use of public-key encryption

Two versions: version 4 and 5

Version 4 makes use of DES

# KERBEROS

Users wish to access services on servers.

Three threats exist:

- User pretend to be another user.
- User alter the network address of a workstation.
- User eavesdrop on exchanges and use a replay attack.

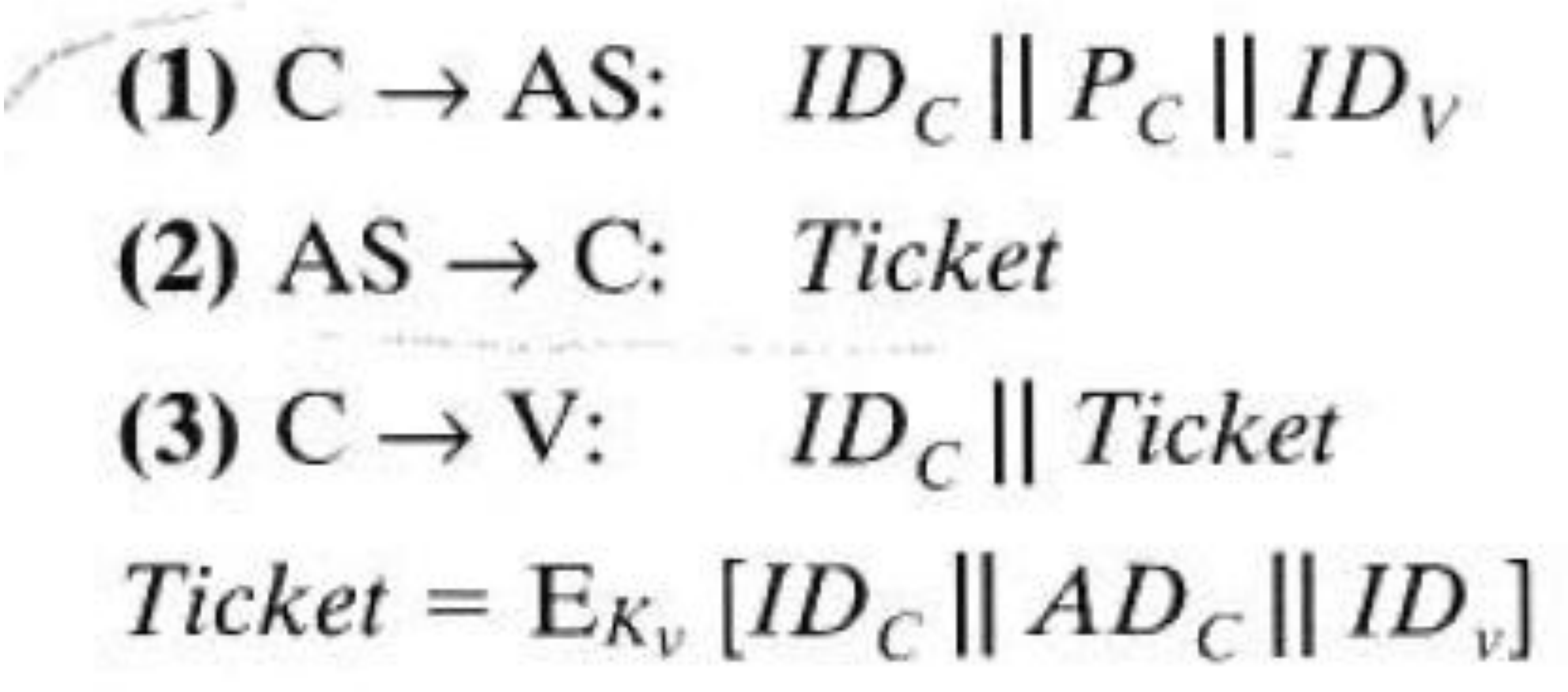
- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

# Kerberos Version 4

Terms:

- C = Client
- AS = authentication server
- V = server
- $ID_c$ = identifier of user on C
- $ID_v$ = identifier of V
- $P_c$ = password of user on C
- $AD_c$ = network address of C
- $K_v$ = secret encryption key shared by AS an V
- TS = timestamp
- || = concatenation

# A Simple Authentication Dialogue

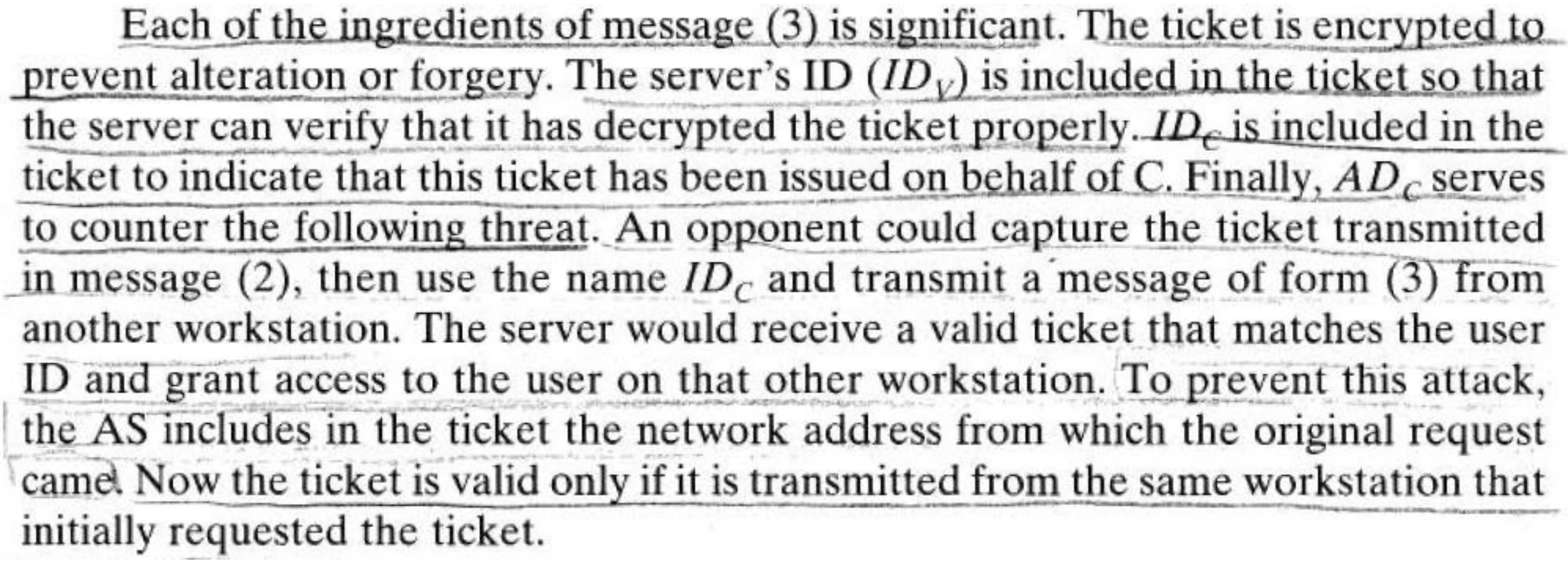


(1) C → AS: $ID_C \,\|\, P_C \,\|\, ID_V$

(2) AS → C: $Ticket$

(3) C → V: $ID_C \,\|\, Ticket$

$Ticket = \mathrm{E}_{K_v}\,[ID_C \,\|\, AD_C \,\|\, ID_v]$



Each of the ingredients of message (3) is significant. The ticket is encrypted to prevent alteration or forgery. The server's ID ($ID_V$) is included in the ticket so that the server can verify that it has decrypted the ticket properly. $ID_C$ is included in the ticket to indicate that this ticket has been issued on behalf of C. Finally, $AD_C$ serves to counter the following threat. An opponent could capture the ticket transmitted in message (2), then use the name $ID_C$ and transmit a message of form (3) from another workstation. The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation. To prevent this attack, the AS includes in the ticket the network address from which the original request came. Now the ticket is valid only if it is transmitted from the same workstation that initially requested the ticket.

# X.509 Authentication Service

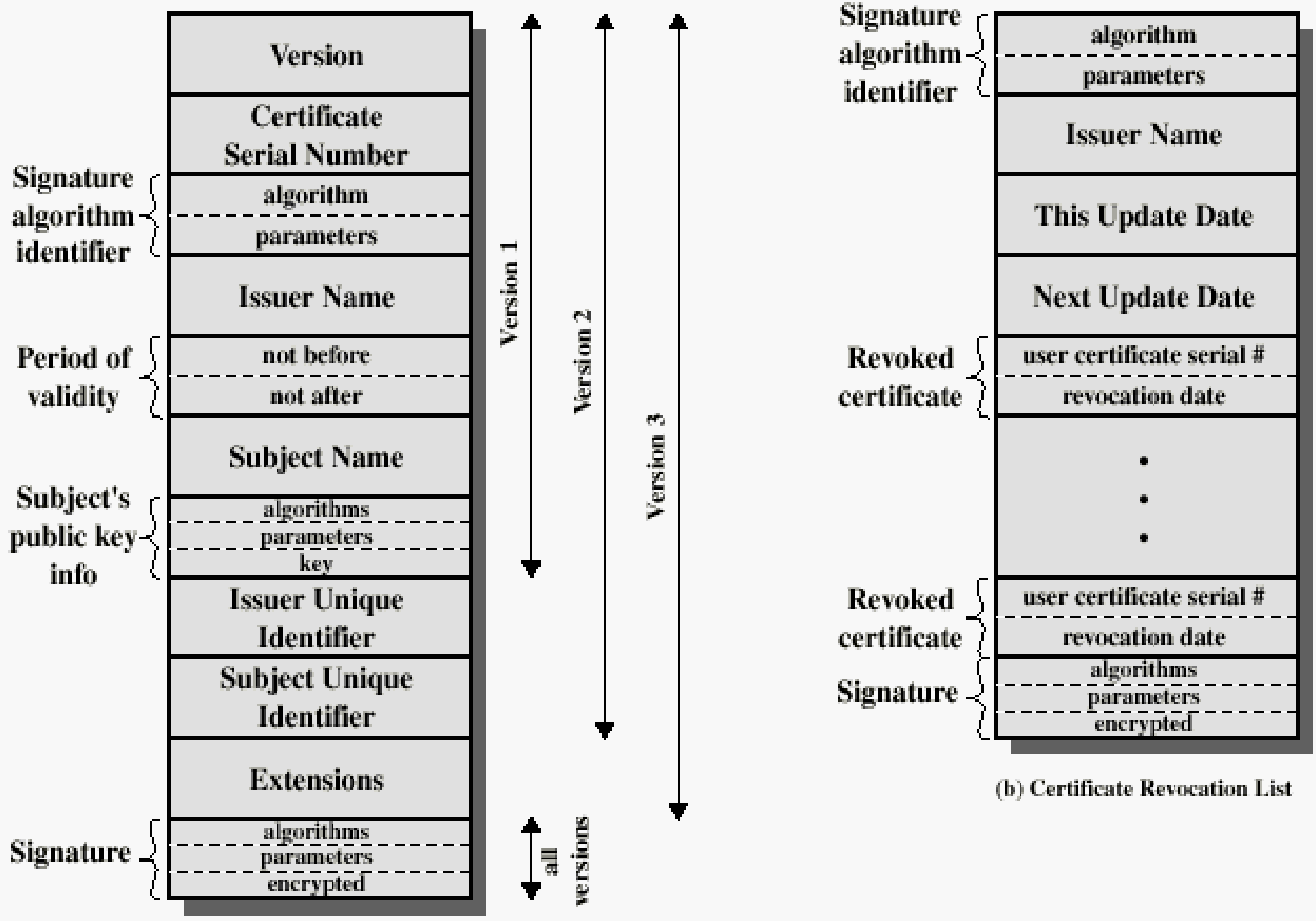
Distributed set of servers that maintains a database about users.

Each certificate contains the public key of a user and is signed with the private key of a CA.

Is used in S/MIME, IP Security, SSL/TLS and SET.

RSA is recommended to use.

# X.509 Formats



(b) Certificate Revocation List

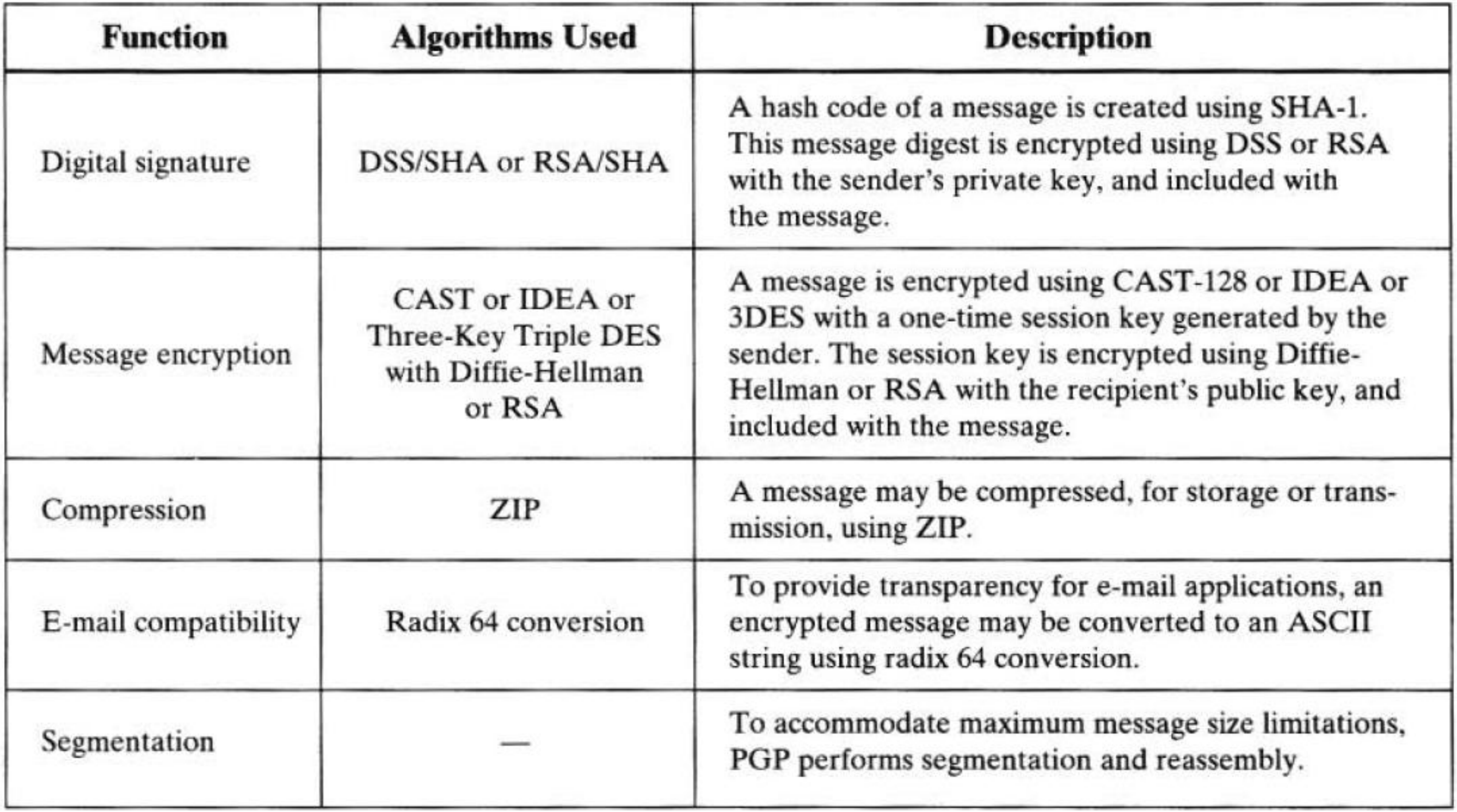
(a) X.509 Certificate

# Operational Description

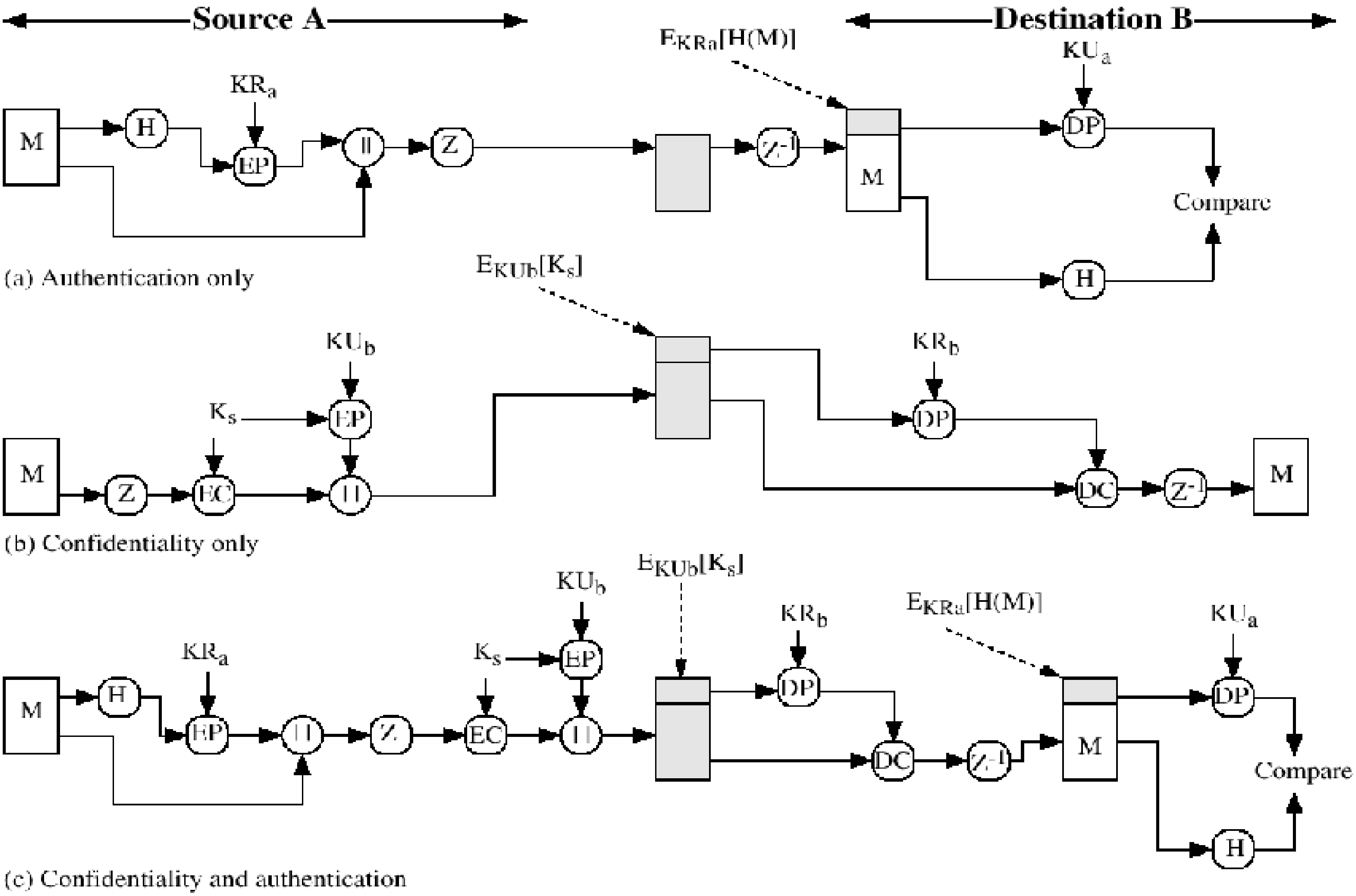
Consist of five services:

- Authentication
- Confidentiality
- Compression
- E-mail compatibility
- Segmentation

**Table 5.1** Summary of PGP Services



| Function | Algorithms Used | Description |
|---|---|---|
| Digital signature | DSS/SHA or RSA/SHA | A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key, and included with the message. |
| Message encryption | CAST or IDEA or Three-Key Triple DES with Diffie-Hellman or RSA | A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key, and included with the message. |
| Compression | ZIP | A message may be compressed, for storage or transmission, using ZIP. |
| E-mail compatibility | Radix 64 conversion | To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix 64 conversion. |
| Segmentation | — | To accommodate maximum message size limitations, PGP performs segmentation and reassembly. |



(a) Authentication only

(b) Confidentiality only

(c) Confidentiality and authentication

**Figure 5.1 PGP Cryptographic Functions**

**1.** The signature is generated before compression for two reasons:

- **a.** It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.
- **b.** Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

# E-mail Compatibility

The scheme used is radix-64 conversion (see appendix 5B).

The use of radix-64 expands the message by 33%.

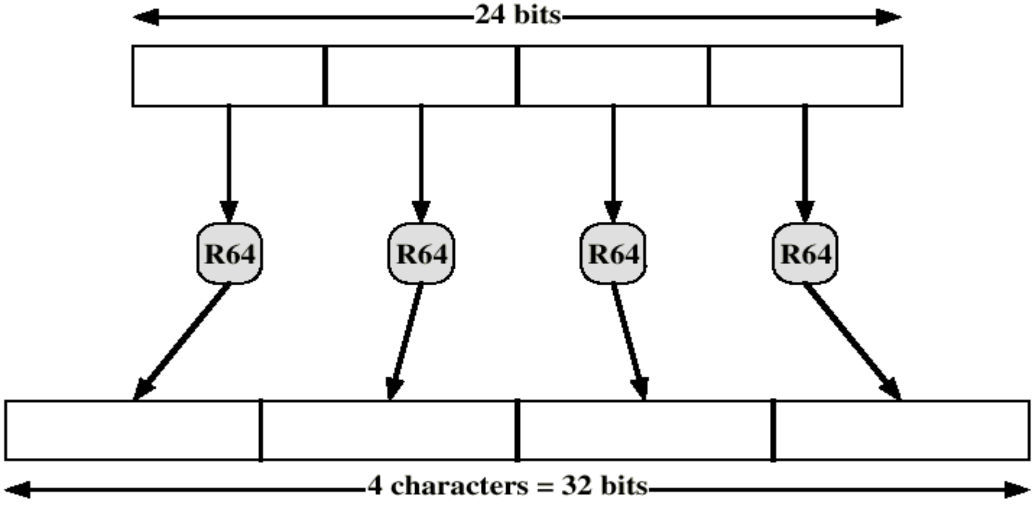
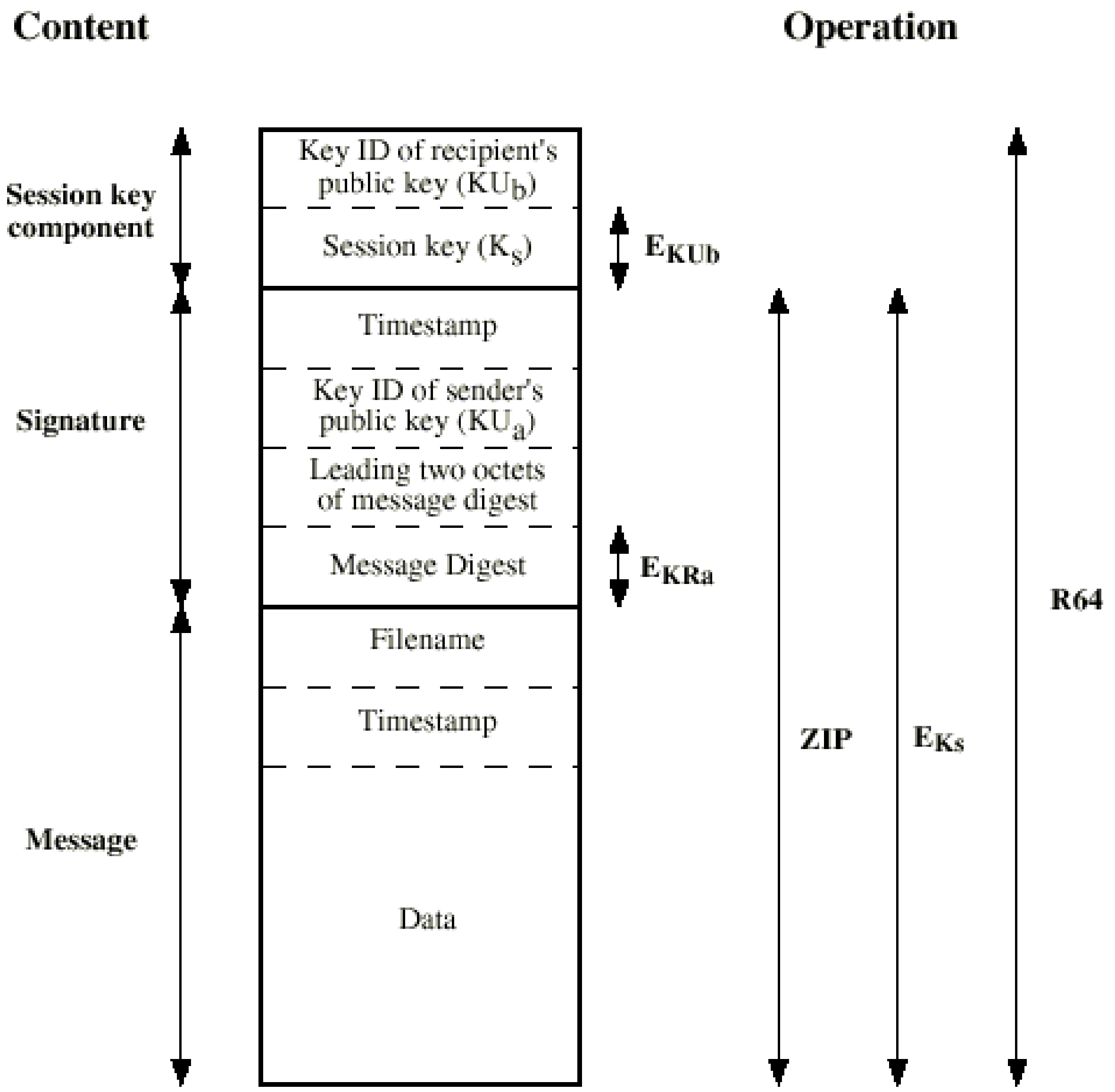


Figure 5.11 Printable Encoding of Binary Data into Radix-64 Format

# Format of PGP Message



# S/MIME

Secure/Multipurpose Internet Mail Extension

S/MIME will probably emerge as the industry standard.

PGP for personal e-mail security

# S/MIME Functions

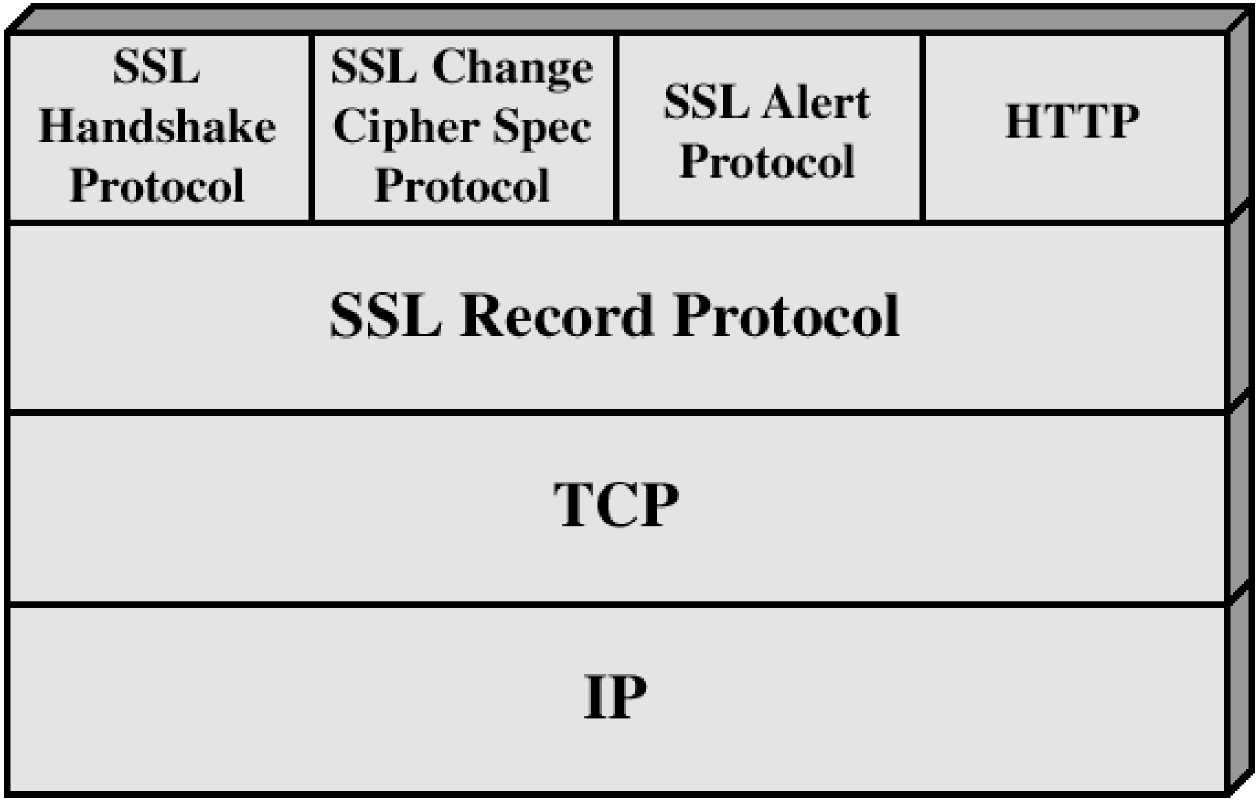
**Enveloped Data:** Encrypted content and encrypted session keys for recipients.

**Signed Data:** Message Digest encrypted with private key of “signer.”

**Clear-Signed Data:** Signed but not encrypted.

**Signed and Enveloped Data:** Various orderings for encrypting and signing.

# SSL Architecture



SSL Handshake Protocol

SSL Change Cipher Spec Protocol

SSL Alert Protocol

HTTP

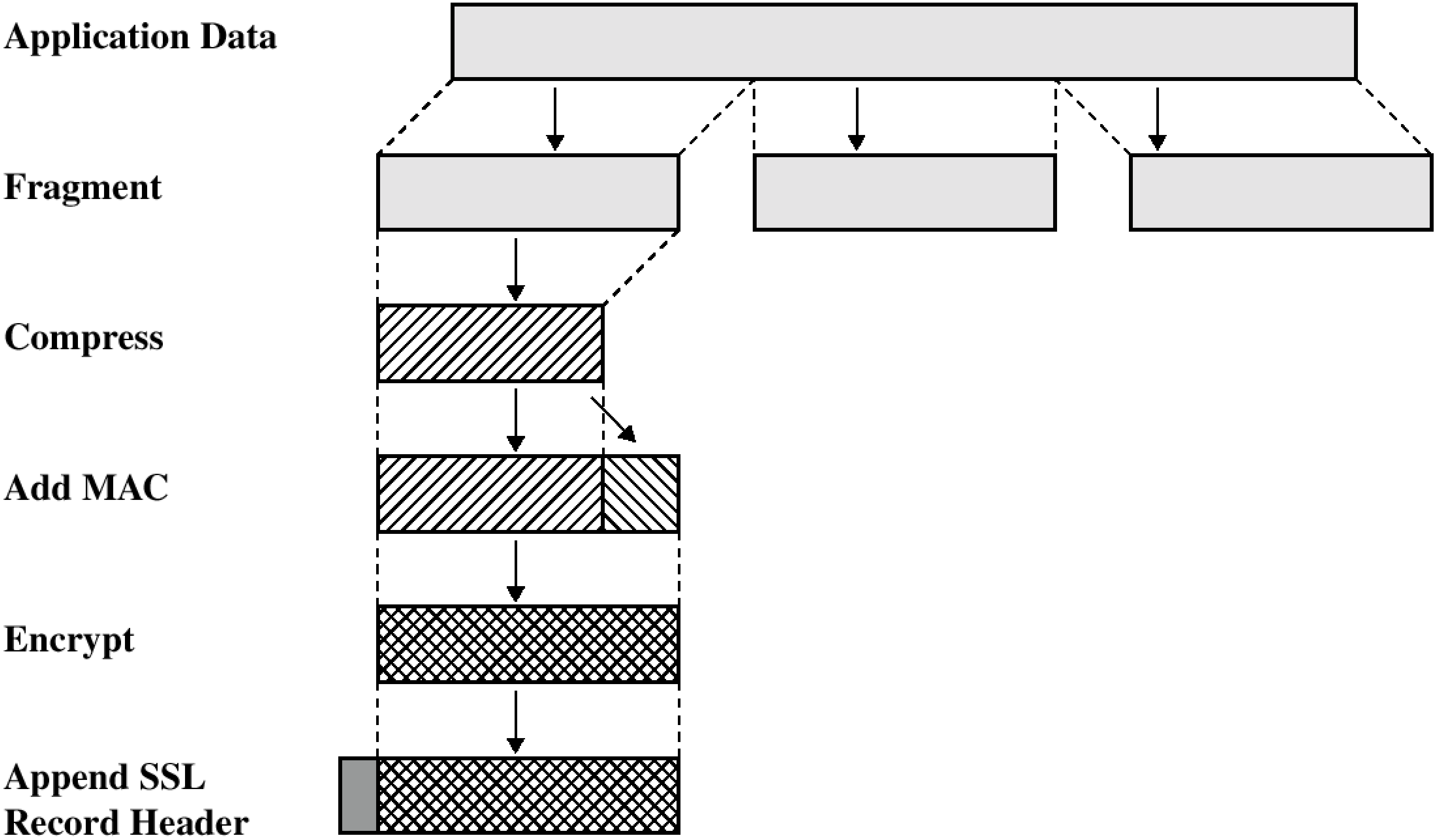
SSL Record Protocol

TCP

IP

Figure 7.2 SSL Protocol Stack

# SSL Record Protocol Operation



# Transport Layer Security

The same record format as the SSL record format.

Defined in RFC 2246.

Similar to SSLv3.

Differences in the:

- version number
- message authentication code
- pseudorandom function
- alert codes
- cipher suites
- client certificate types
- certificate_verify and finished message
- cryptographic computations
- padding

# SET Services

Provides a secure communication channel in a transaction.

Provides trust by the use of X.509v3 digital certificates.

Ensures privacy.

# SET Overview

Key Features of SET:

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

- **Confidentiality of information:** Cardholder account and payment information is secured as it travels across the network. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank. Conventional encryption by DES is used to provide confidentiality.
- **Integrity of data:** Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit. RSA digital signatures, using SHA-1 hash codes, provide message integrity. Certain messages are also protected by HMAC using SHA-1.
- **Cardholder account authentication:** SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.
- **Merchant authentication:** SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose.