Modern Cryptography
and Its Applications

6 Secure Communication ——Confidentiality Using Symmetric Encryption

Sec10.1 & Sec14.1 & Sec14.2 in textbook

Yanwei Yu

E-mail: ywyu@ustc.edu.cn

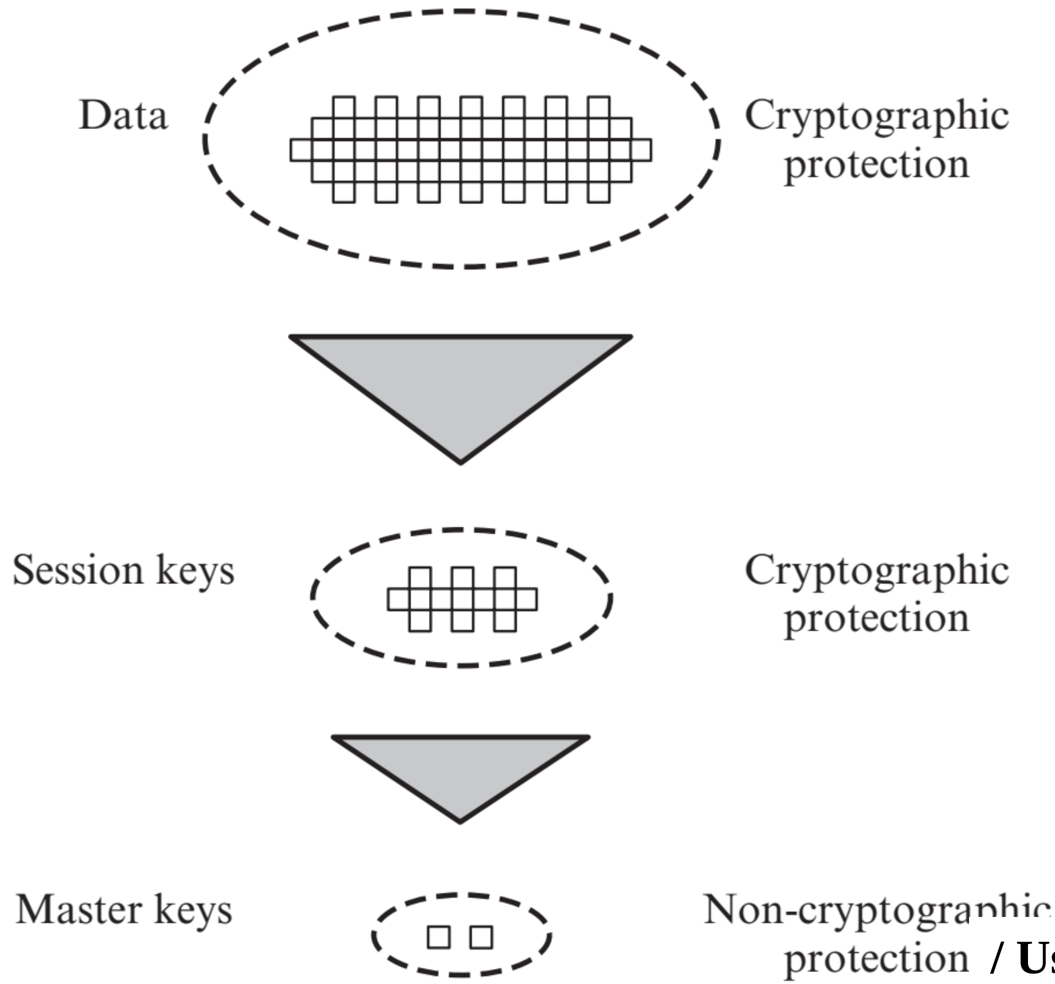
Outline

- **Key Hierarchy(层次结构)**
- **Secure Secret-key Distribution**

1.1 Key Hierarchy(层次结构)

- typically have a hierarchy of keys
- session(会话) key
  - temporary key
  - used for encryption of data between users
  - for one logical session then discarded
- master(主) key
  - used to encrypt session keys
  - shared by user & key distribution center for long time

1.1 Key Hierarchy(层次结构)



/ Using public encryption

Figure 14.2 The Use of a Key Hierarchy

Lifetime and usage of session key

- Lifetime of Session Key: temporary
  - Balance of security and efficiency
- Session Key:
  - Data-encrypting key, for general communication across a network
  - PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
  - File-encrypting key, for encrypting files stored in publicly accessible locations
- Usage of key
  - Use key tag(suitable for master key and session key)
  - control vector(suitable for session key) Fig14.6

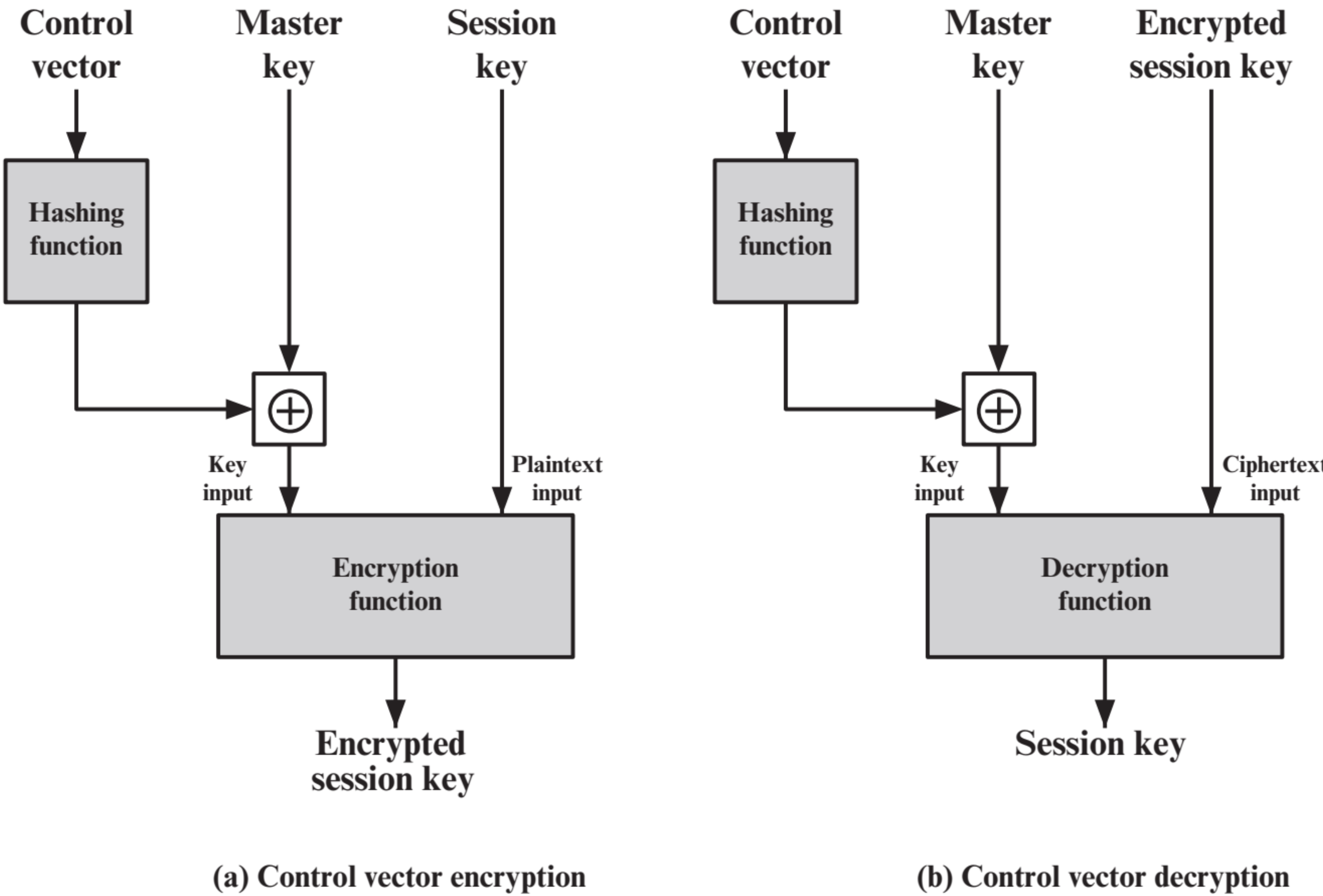


Figure 14.6 Control Vector Encryption and Decryption

1.2 Secret Key Distribution

- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- often secure system failure due to a break in the key distribution scheme

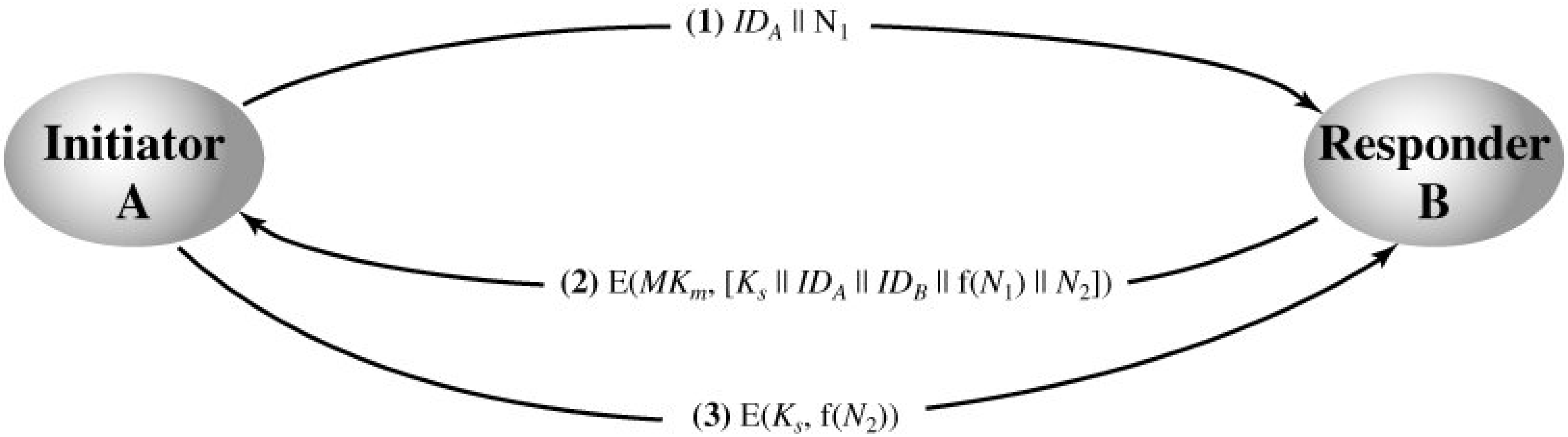
1.2 Secret Key Distribution

- **given parties A and B have various** key distribution **alternatives:**
  1. **A can select key and physically deliver to B**
  2. **third party can select & deliver key to A & B**
  3. **Decentralized Key Control:** if A & B have communicated previously, they can use previous key to encrypt a new key
  4. **Centralized Key Control:** if A & B have secure communications with a third party C, C can relay(传达) key between A & B
  5. **Use public encryption to protect secret key shared by both**
  6. **Key Pre-distribution Schemes(no key required)**

Use master key

(1) Decentralized Key Control

Assume both ends shared key *MKm*



1. A issues a request to B for a session key and includes a nonce, N1
2. B responds with a message that is encrypted using the shared master key *MKm*. The response includes the session key *Ks* selected by B, an identifier of B, the value f(N1), and another nonce, N2.
3. Using the new session key, A returns f(N2) to B.

Q: Why need N1 and N2?

Discussion——Point-to-point key update using symmetric encryption

- *key transport with one pass*:

$$A \rightarrow B : E_K(r_A) \quad (1)$$
$$A \rightarrow B : E_K(r_A, t_A{}^*, B^*) \quad (1')$$

Ks=r_A

- *key transport with challenge-response*:

$$A \leftarrow B : n_B \quad (1)$$
$$A \rightarrow B : E_K(r_A, n_B, B^*) \quad (2)$$

Ks=r_A

$$A \leftarrow B : n_B \quad (1)$$
$$A \rightarrow B : E_K(r_A, n_A, n_B, B^*) \quad (2)$$
$$A \leftarrow B : E_K(r_B, n_B, n_A, A^*) \quad (3)$$

Ks=$f(r_A, r_B)$

Point-to-point key update by key derivation and non-reversible functions

Define $T = (B, A, r_A, r_B)$.

$$A \rightarrow B: \quad r_A \quad (1)$$
$$A \leftarrow B: \quad T,\ h_K(T) \quad (2)$$
$$A \rightarrow B: \quad (A, r_B),\ h_K(A, r_B) \quad (3)$$

$$\mathbf{Ks} = h'_{K'}(r_B)$$

(a) A selects and sends to B a random number r_A.

(b) B selects a random number r_B and sends to A the values (B,A, r_A, r_B), along with a MAC over these quantities generated using h with key K.

(c) Upon receiving message (2), A checks the identities are proper, that the r_A received matches that in (1), and verifies the MAC.

(d) A then sends to B the values (A, r_B), along with a MAC.

(e) Upon receiving (3), B verifies that the MAC is correct, and that the received value r_B matches that sent earlier.

(f) Both A and B compute the session key as Ks = $h_K(r_B)$.

(2) Centralized Key Control

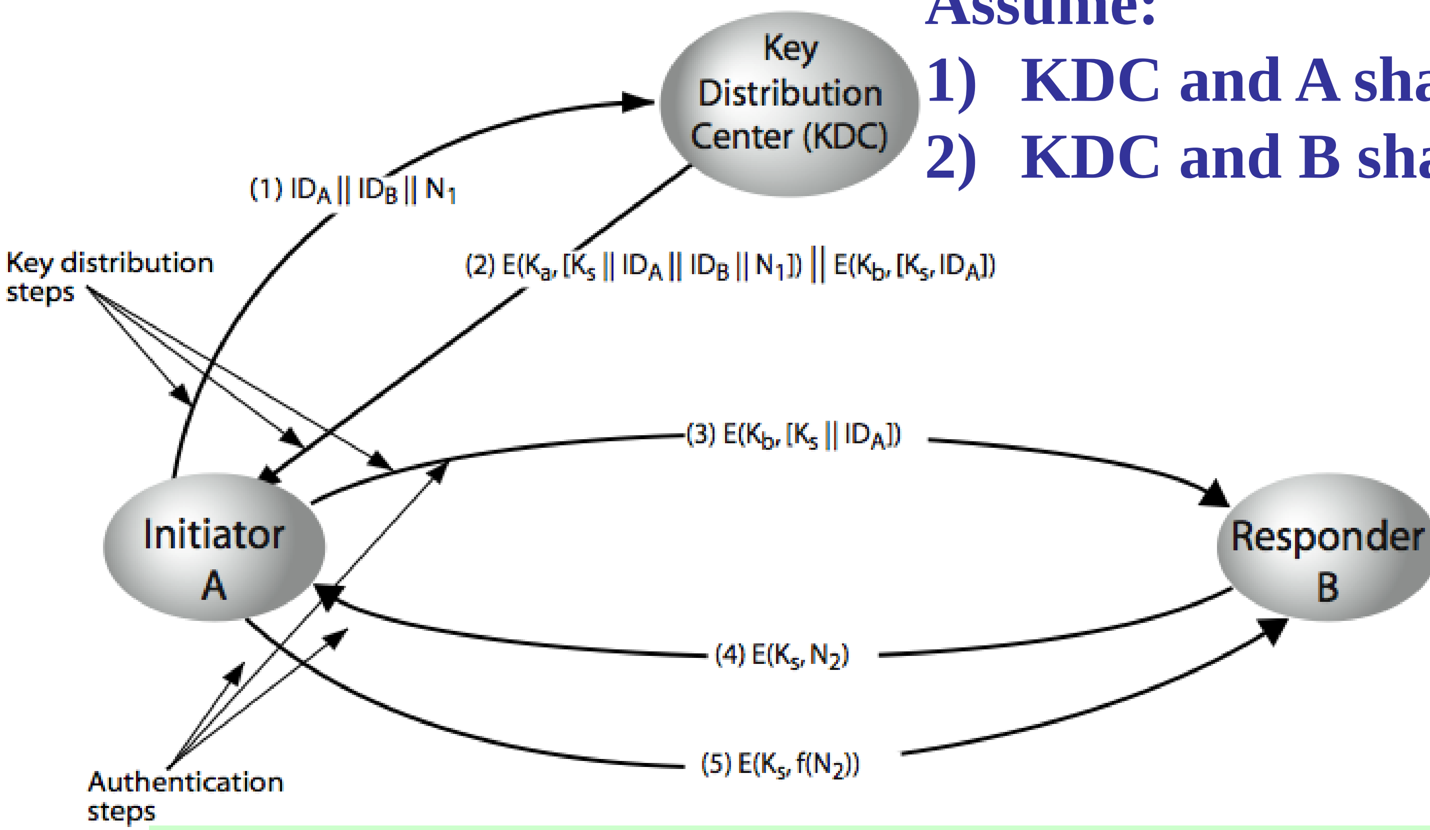
- **Example1**: Needham-Schroeder shared-key protocol
  - Basic scheme and its improvement

- **Example2**: **Kerberos**
  - Basic principles

- **Hierarchical Key Control**
  - Use a hierarchy of KDCs
  - A Transparent Key Control Scheme

Needham-Schroeder Shared-key Protocol

Assume:
1) KDC and A shared key Ka.
2) KDC and B shared key Kb.



Drawback: any party knowing an old session key Ks may both resend message (3) and compute a correct message (5) to impersonate A to B.

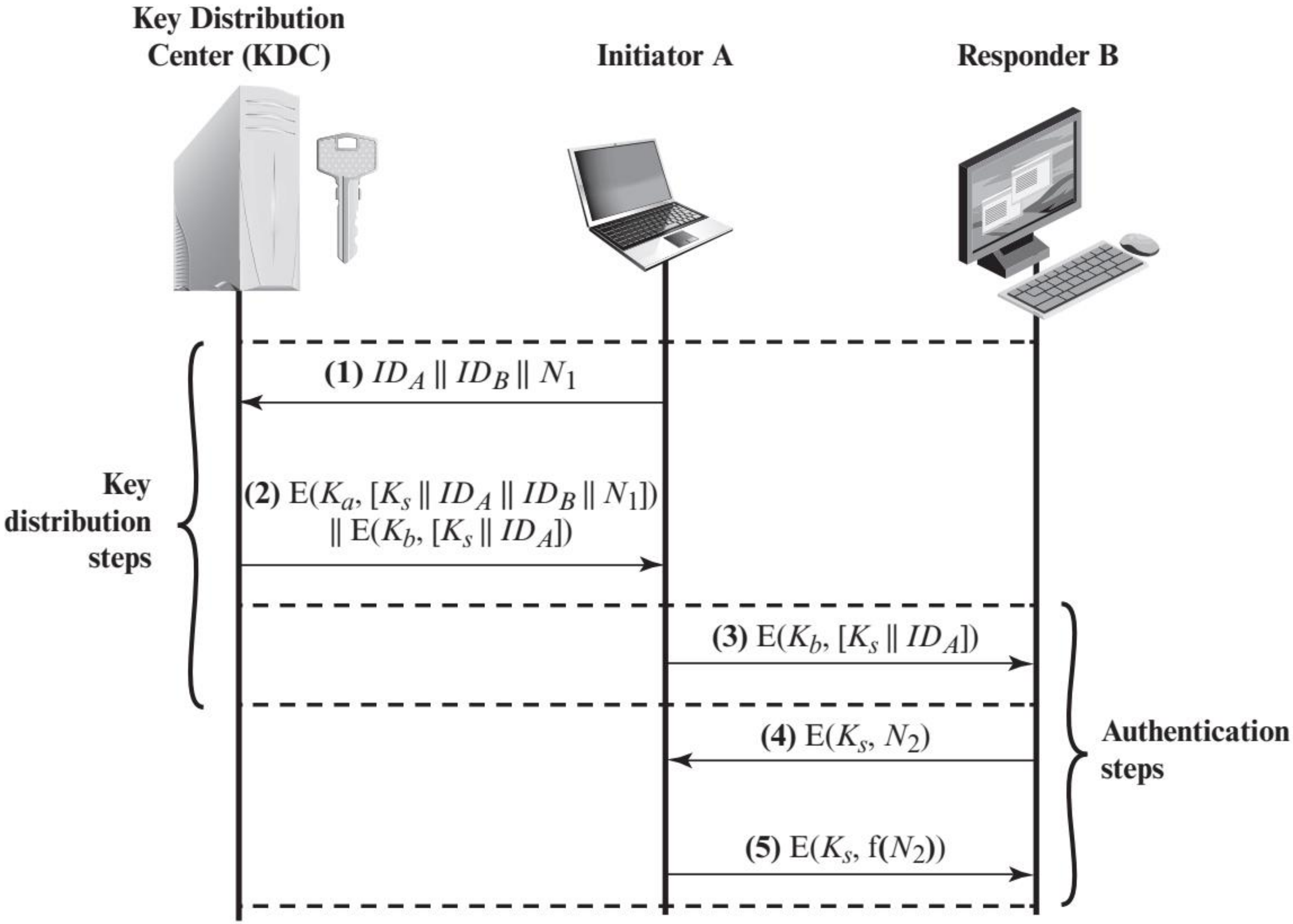
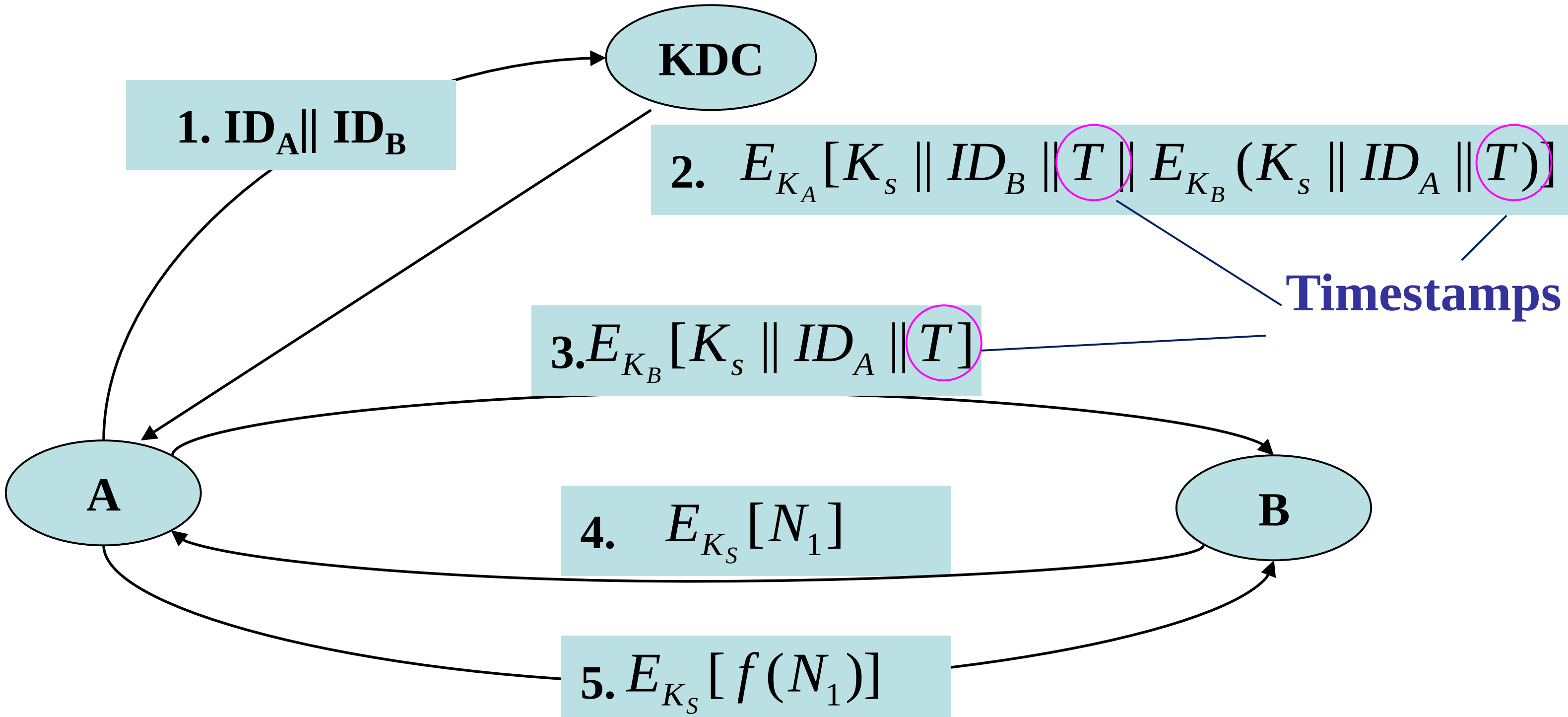


Figure 14.3 Key Distribution Scenario

Needham-Schroeder Improvement (1)



Synchronization is necessary to clocks of each party

$|Clock - T| < \Delta t_1 + \Delta t_2$

Clock: For host;

Δt_1: estimated difference between hosts and KDC;

Δt_2: network delay;

Needham-Schroeder Improvement (2)

KDC

A

B

1. $ID_A \parallel N_A$

2. $ID_B \parallel N_B \parallel E_{K_B}[ID_A \parallel N_A \parallel T_B]$

3. $E_{K_A}[ID_B \parallel N_A \parallel K_S \parallel T_B] \parallel E_{K_B}[ID_A \parallel K_S \parallel T_B] \parallel N_B$

4. $E_{K_B}[ID_A \parallel K_S \parallel T_B] \parallel E_{K_S}[N_B]$

Deadline for session key

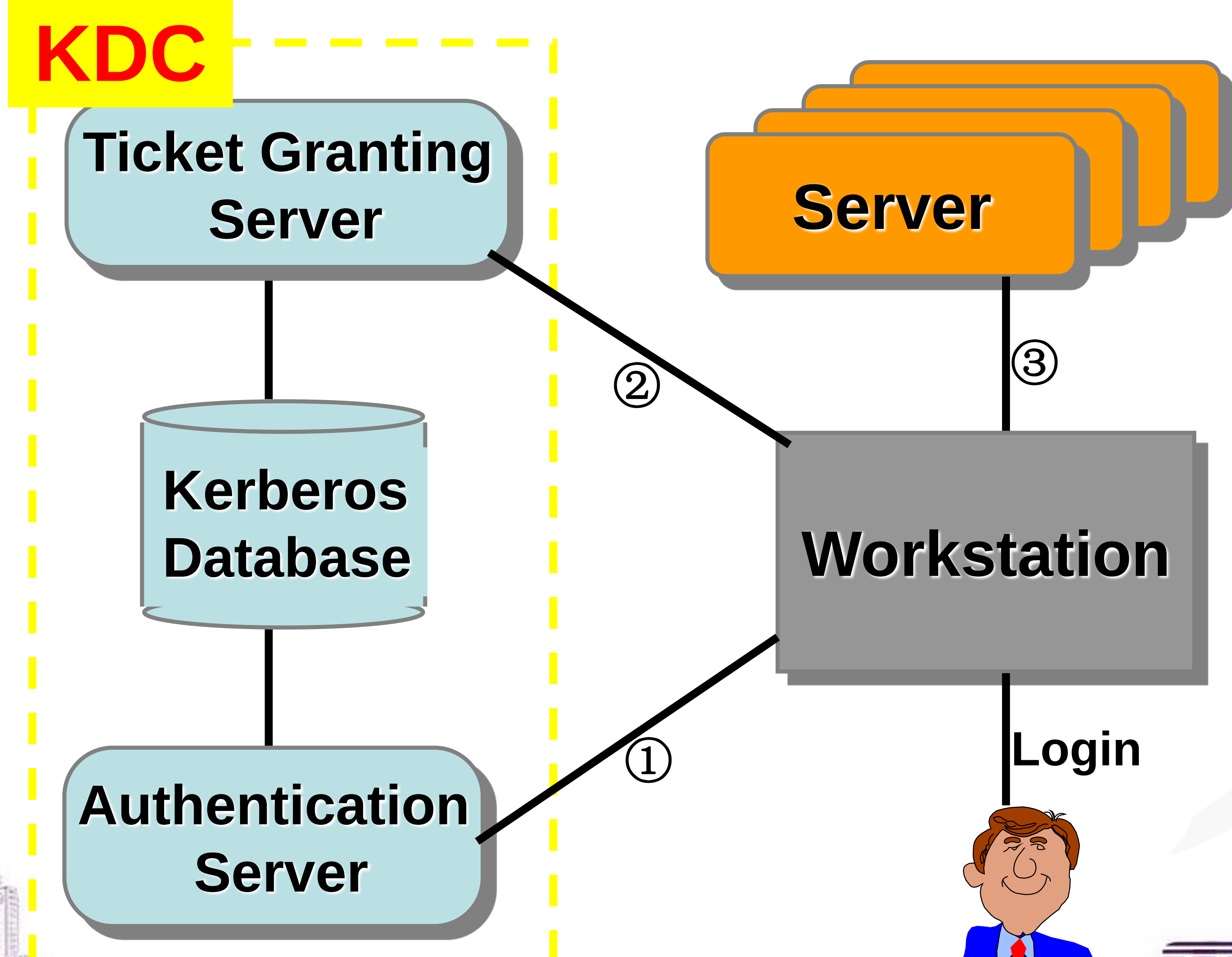
Needham-Schroeder Improvement (2)

In the valid lifetime of Ks, need not authentication of KDC

A → B: **1.** $E_{K_B}[ID_A \| K_S \| T_B], N_A^{'}$

B → A: **2.** $N_B^{'}, E_{K_S}[N_A^{'}]$

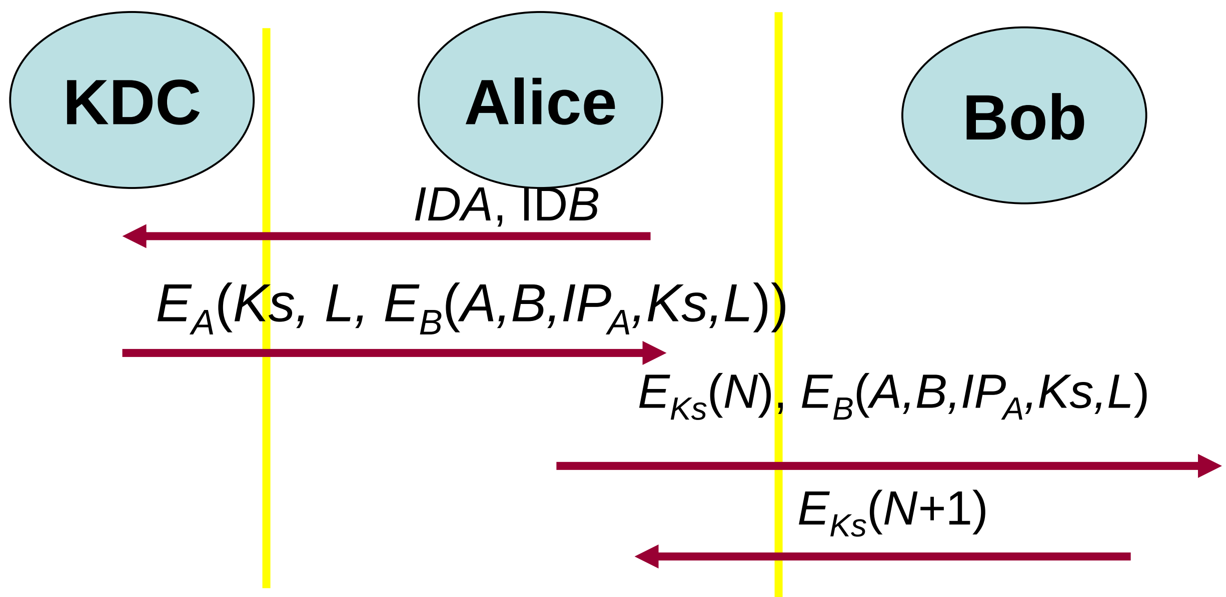
A → B: **3.** $E_{K_S}[N_B^{'}]$



Kerberos **key distribution service**

Basic Principle of Kerberos

- **Main target： entity authentication**
- **Additional results: shared secret key distribution**



- $E_A(Ks, L, E_B(A,B,IP_A,Ks,L))$
  - L: lifetime of session key K

Encryption by master key between KDC and B

session key

Ticket:
login name
TGS name
net address
TGS session key

Encryption by master key between KDC and A

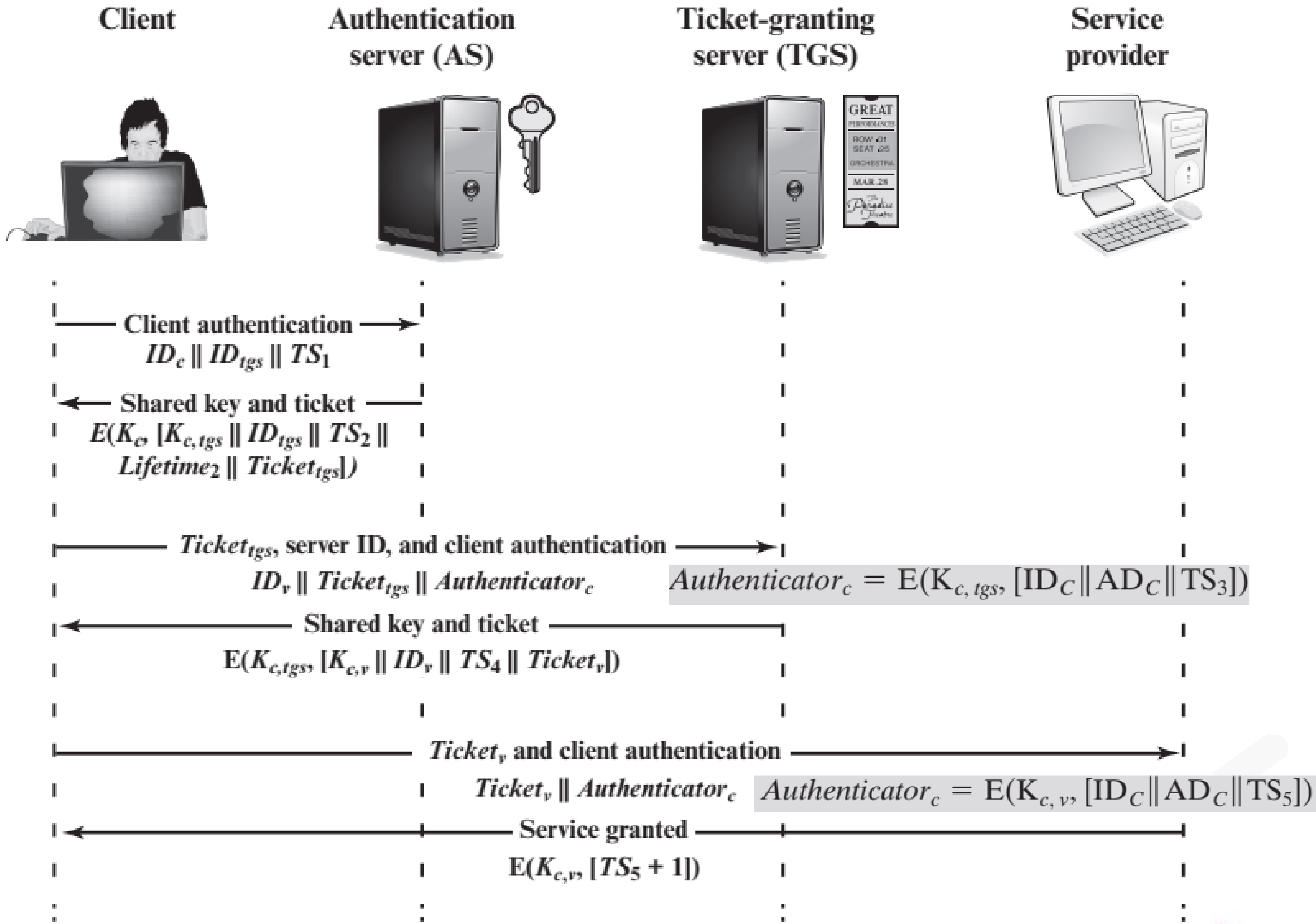
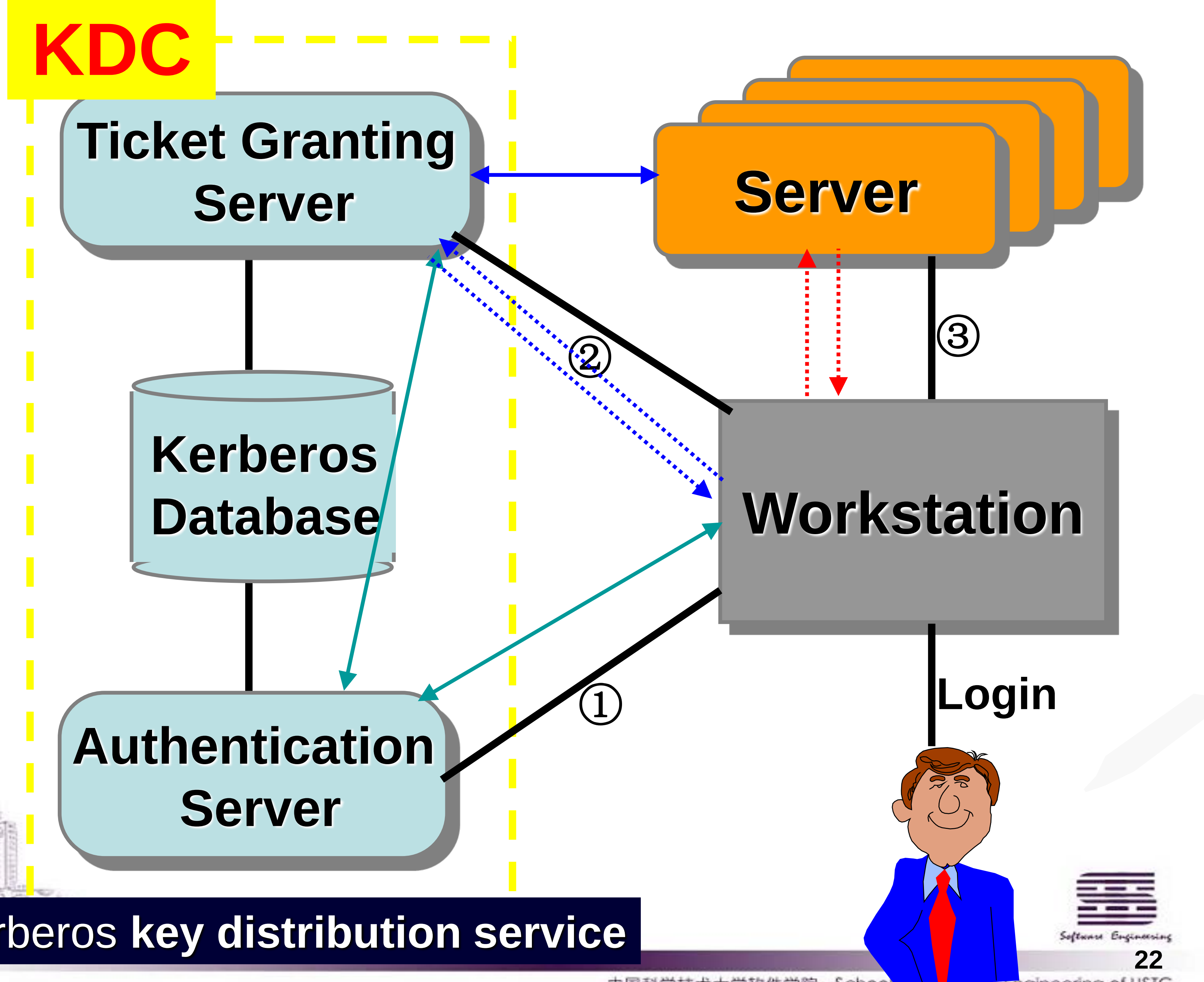


Figure 15.3 Kerberos Exchanges

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,\,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,\,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

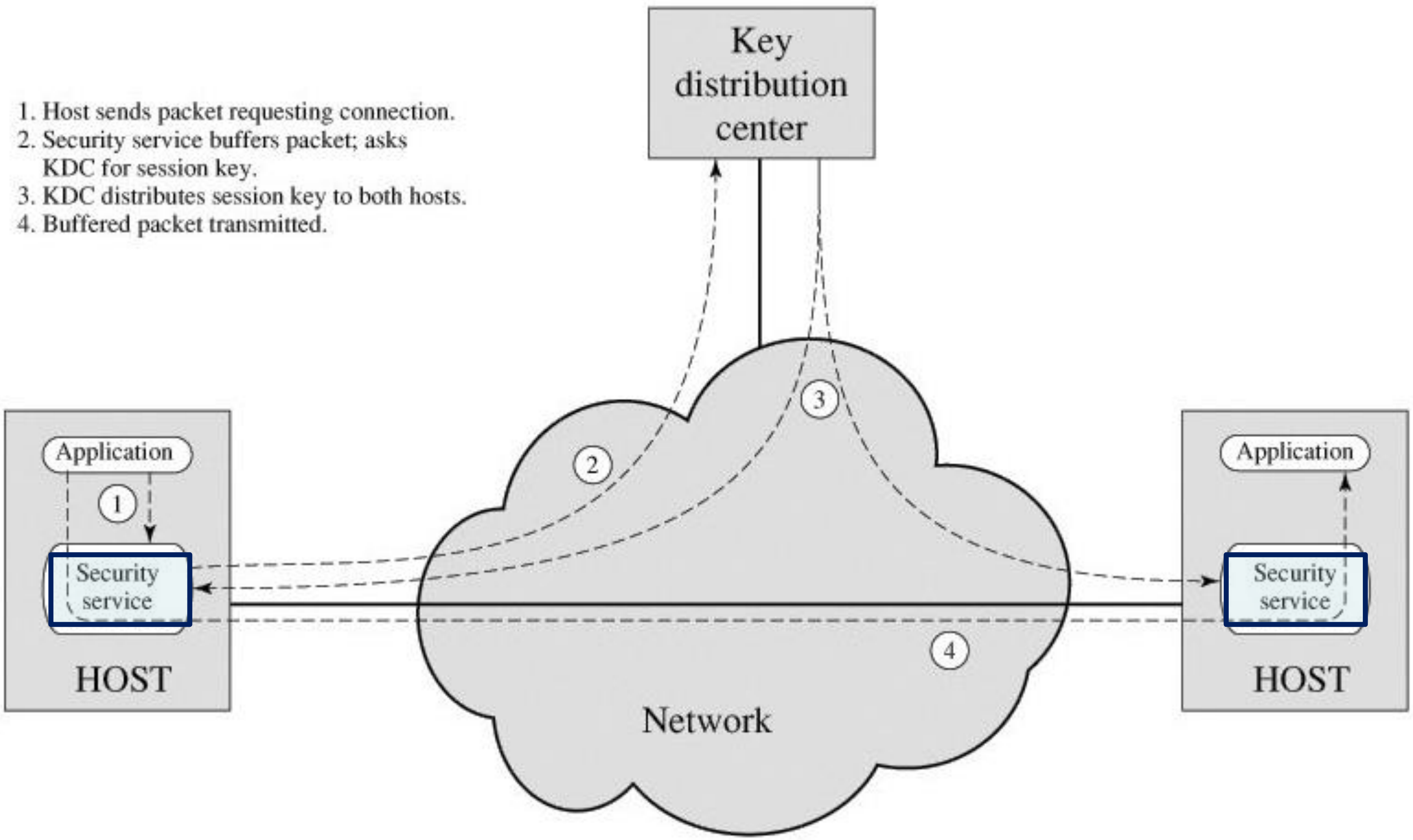


Kerberos **key distribution service**

Hierarchical Key Control

- Use a hierarchy of KDCs
  - each local KDC is responsible for a small domain of the overall internetwork, such as a single LAN or a single building
  - For communication among entities within the same local domain, the local KDC is responsible for key distribution.
  - If two entities in different domains desire a shared key, the corresponding local KDCs can communicate through a global KDC
- Suit for very large networks
  - Low cost:
    - most master keys are those shared by a local KDC with its local entities
  - Limited damage:
    - damage of a faulty or subverted(坏的) KDC to its local area only

Automatic Key Distribution for Connection-Oriented Protocol



Key Distribution Issues

- **session key lifetimes should be limited for greater security**
  - **How to determine the lifetime?**
    - **Balance security and exchange delay & network burden**
      - **more frequently session keys exchanged, more secure**
      - **But, distribution of session keys delays the start of any exchange and places a burden on network capacity**
    - **For connection-oriented protocols, use a new session key for each new session**
    - **For a connectionless protocol, use a given session key for a certain fixed period only or for a certain number of transactions**

Key Distribution Issues

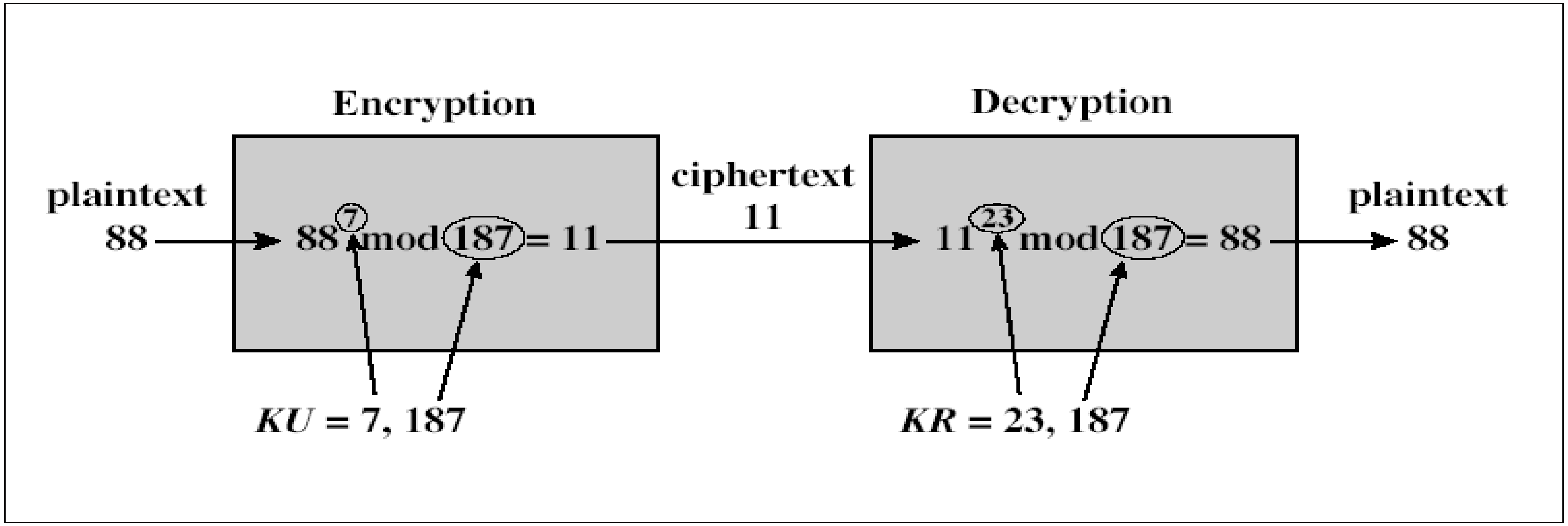
- use of automatic key distribution
  - on behalf of users
  - KDC be trusted and be protected from subversion
  - hierarchies of KDC’s required for large networks, but must trust each other
- use of decentralized key distribution
  - requires [n(n-1)]/2 master keys for a configuration with n end systems
- controlling key usage
  - Use key tag or use control vector

(3) Key Transport based on Public-key Encryption

- What is Public-key Encryption?

- Secret Key Distribution
  - Simple Secret Key Distribution
  - Secret Key Distribution with Confidentiality and Authentication

Public-key Encryption

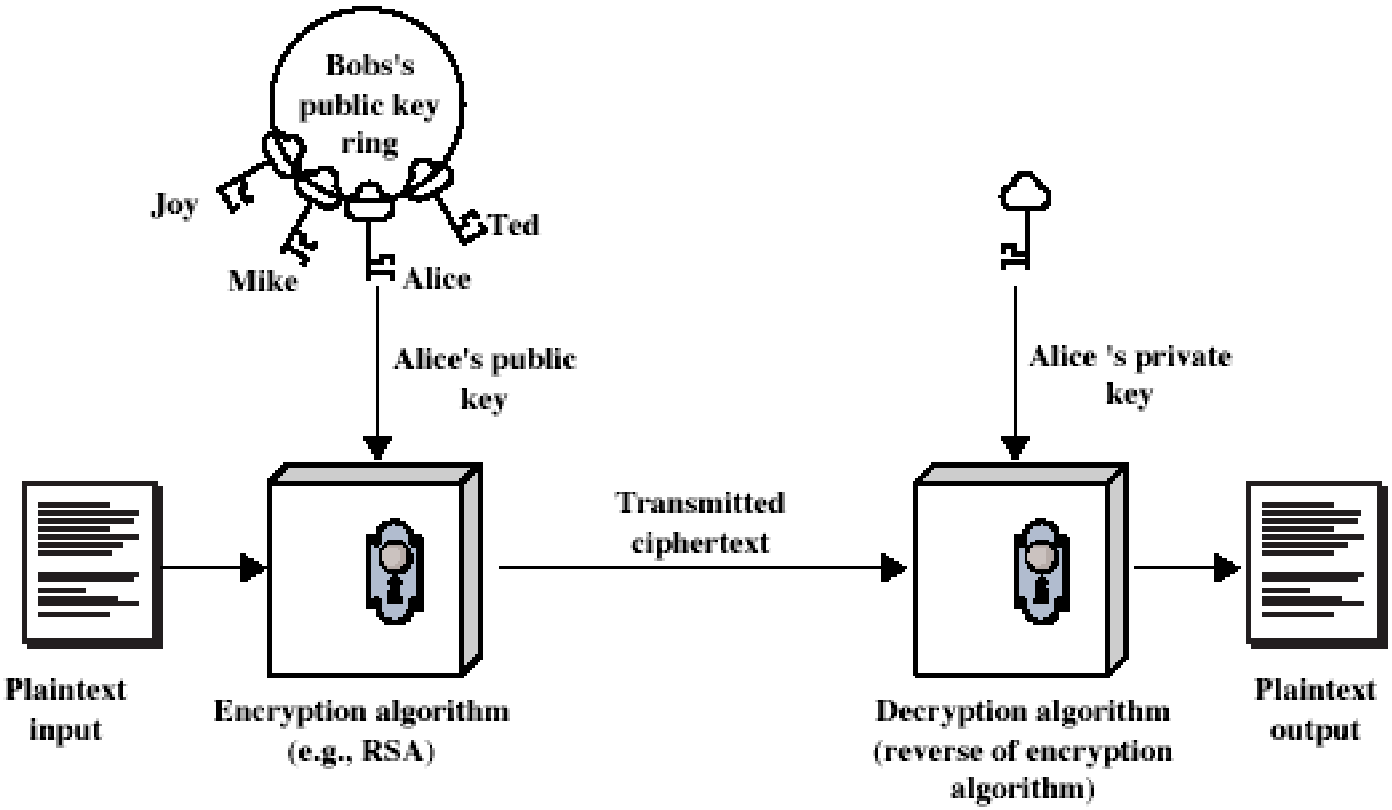


Public key: 7 and 187 , Private key: 23

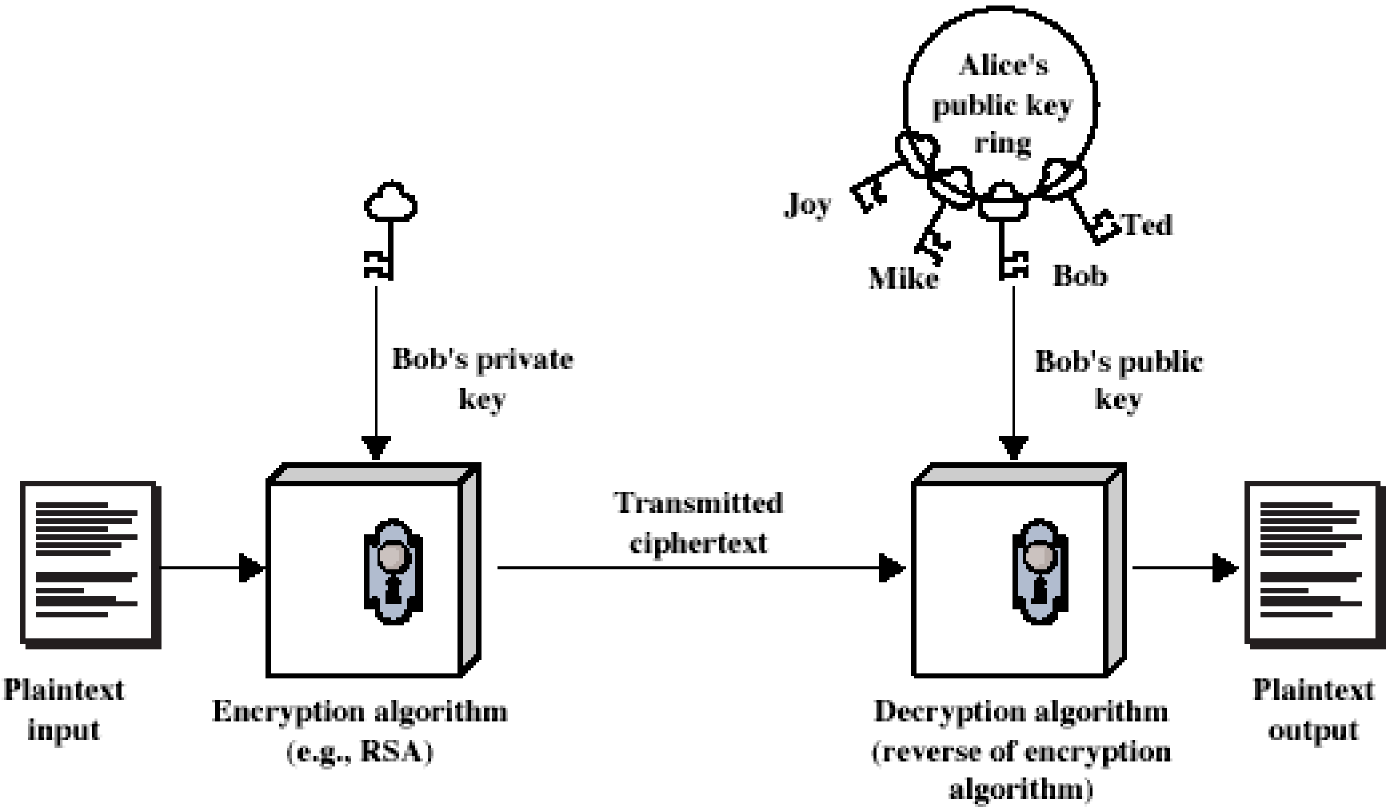
Plain-text 88 cannot be concluded from only 7, 187 and cipher-text 11

Mathematics is so wonderful!

- **public-key algorithms are slow**
- **used for data confidentiality , authentication and non-repudiation**
- **Confidentiality: usually used to protect secret key (master key or session key)**

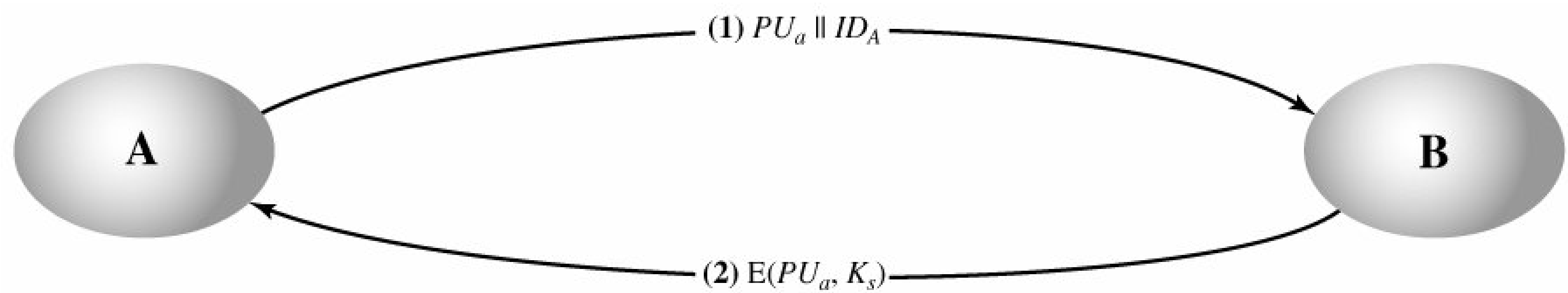


(a) Encryption



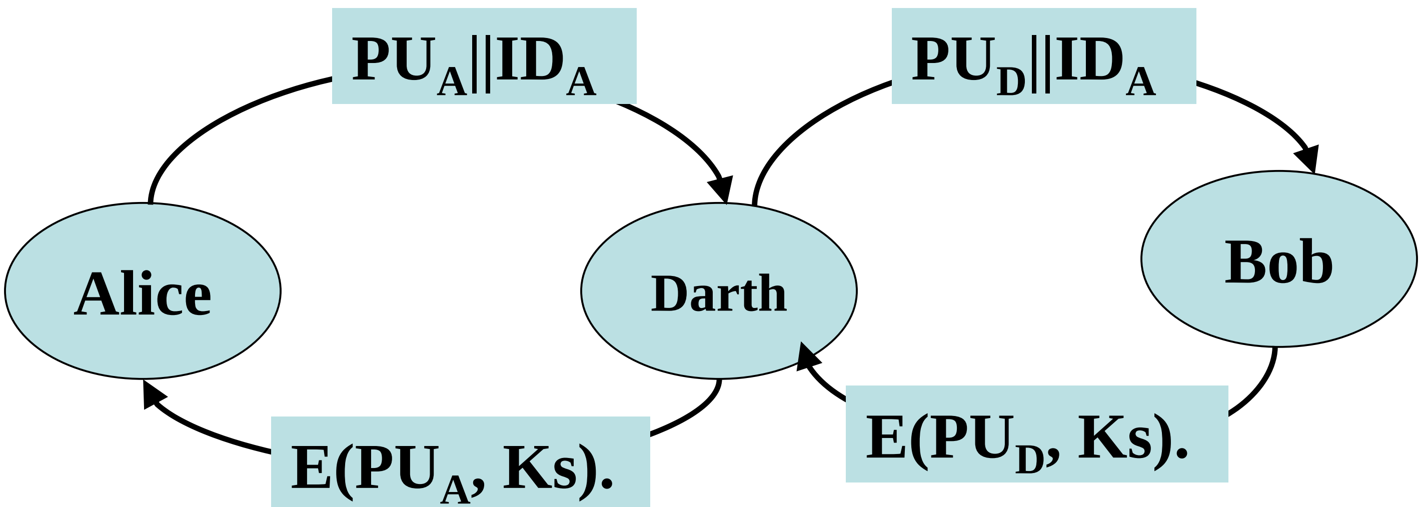
(b) Authentication

Simple Secret Key Distribution



- A generates a new **temporary** public key pair
- A sends B the public key and their identity
- B generates a session key K sends it to A encrypted using the supplied public key
- A decrypts the session key and both use

Man-in-the-middle Attack



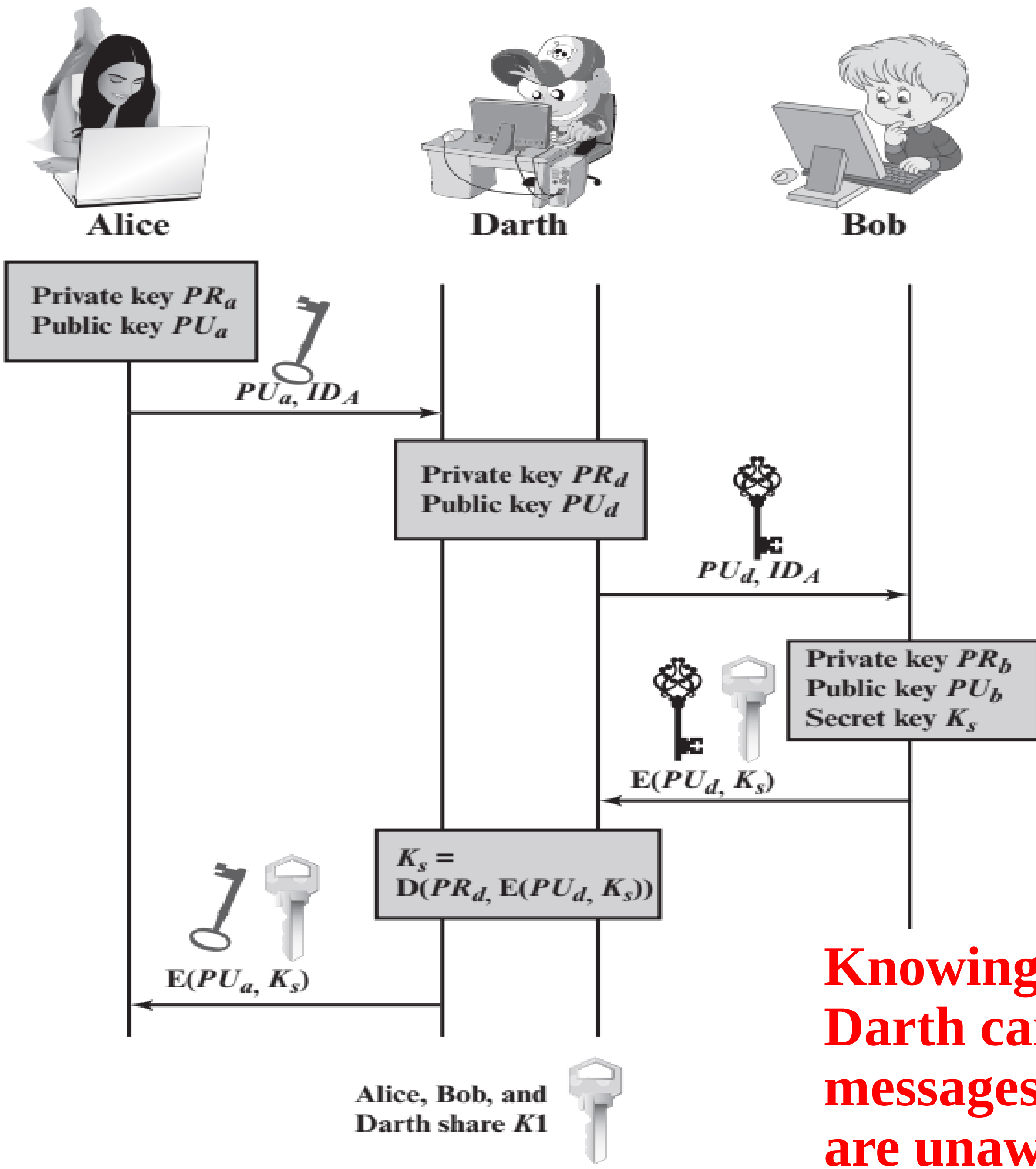
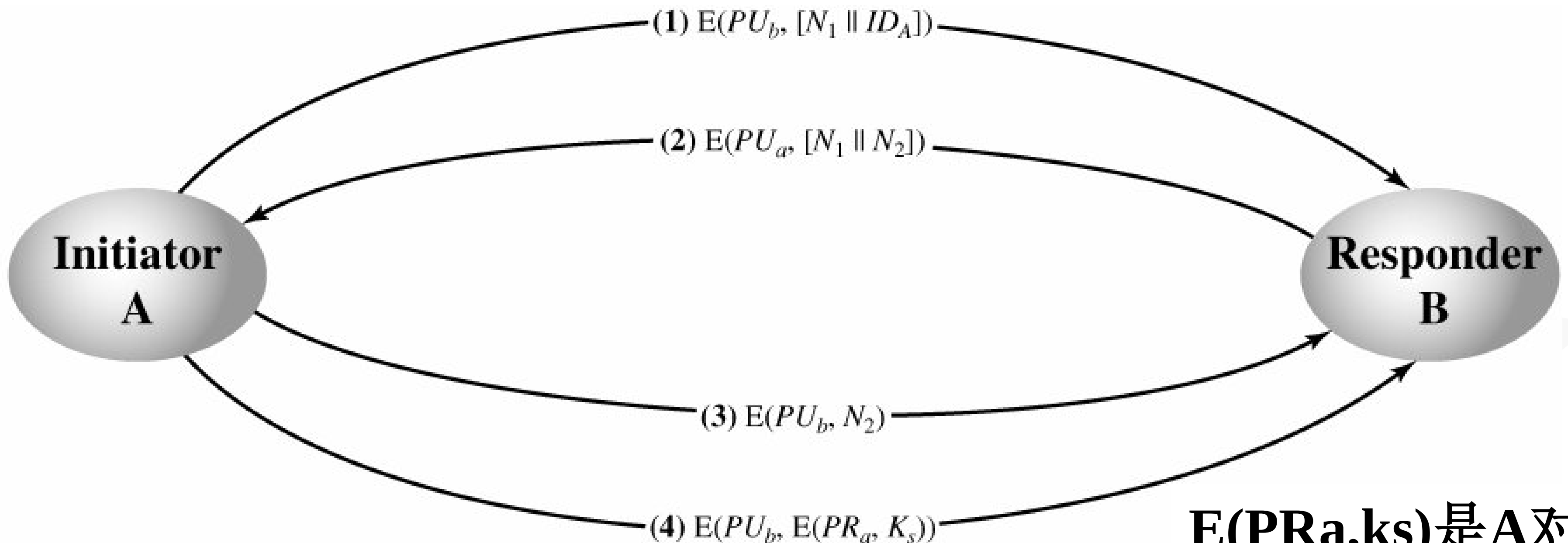


Figure 14.8 Another Man-in-the-Middle Attack

Knowing Ks, Adaversary Darth can decrypt all messages, and both A and B are unaware of the problem.

Secret Key Distribution with Confidentiality and Authentication

- **assumed that A and B have securely exchanged public keys by one of the schemes**



E(PRa,ks)是A对Ks进行数字签名

Hybrid(混合的) Key Distribution

- distributes session key using master key
  - retain use of KDC
  - KDC shares secret master key with each host
- public-key used to distribute master keys
  - especially useful with widely distributed users

(4) Key Pre-distribution Schemes

- two parties establish a shared secret key by Pre-distributed keying material and no cryptographic messages need be exchanged.
- Example: Diffie-Hellman with fixed exponentials
  - Both's Diffie-Hellman public key parameters PU_A and PU_B are contained in a certificate signed by the certificate authority (CA)
  - results in a fixed secret key between two peers for long time, based on the Diffie-Hellman calculation using the fixed public keys.
    - shared Key between A and B is $(PU_A)^{PR_B}=(PU_B)^{PR_A}$, where
      - For A, publice key:PU_A,private key: PR_A
      - For B, publice key:PU_B,private key: PR_B
  - 0 step key exchange if both certificates are known beforehand.

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Classification of Diffie-Hellman key exchange in SSL *P558*

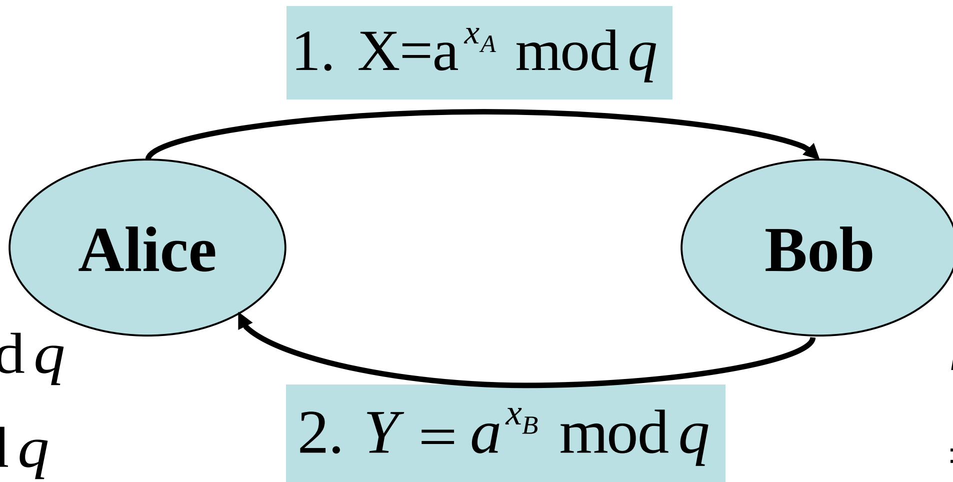
- **Fixed Diffie-Hellman:**
  - the server's Diffie-Hellman public key parametersare contained in a certificate signed by the certificate authority (CA)
  - The client provides its Diffie-Hellman public key parameters either in a certificate
  - This method results in a fixed secret key between two peers for long time, based on the Diffie-Hellman calculation using the fixed public keys.
  - 0 step key exchange if both certificates are known beforehand.

- **Anonymous Diffie-Hellman: (base Diffie-Hellman algorithm)**
  - create temporary(one-time) secret keys, with no authentication.
  - each side sends its public Diffie-Hellman parameters to the other, with no authentication.
  - vulnerable to man-in-the-middle attacks, in which the attacker conducts anonymous Diffie-Hellman with both parties.

- **Ephemeral(瞬时的) Diffie-Hellman:**
  - used to create temporary(one-time) and authenticated secret keys.
  - The Diffie-Hellman public keys are exchanged, signed using the sender's private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys.
  - This would appear to be the most secure of the three Diffie-Hellman options.

Anonymous Diffie-Hellman —— basic version

- **Security is based on difficulty of computing discrete logarithms**



$$k = Y^{x_A} \bmod q = a^{x_A x_B} \bmod q$$

$$k = X^{x_B} \bmod q = a^{x_A x_B} \bmod q$$

global public parameters: a and q
one-time private key: x_A for Alice, x_B for Bob

Diffie-Hellman Setup

- all users agree on global parameters:
  - *large* prime integer or polynomial q
  - a being a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their public key: $y_A = a^{x_A} \bmod q$
- each user makes public that key y_A

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Table 2.7 Powers of Integers, Modulo 19

Diffie-Hellman Example

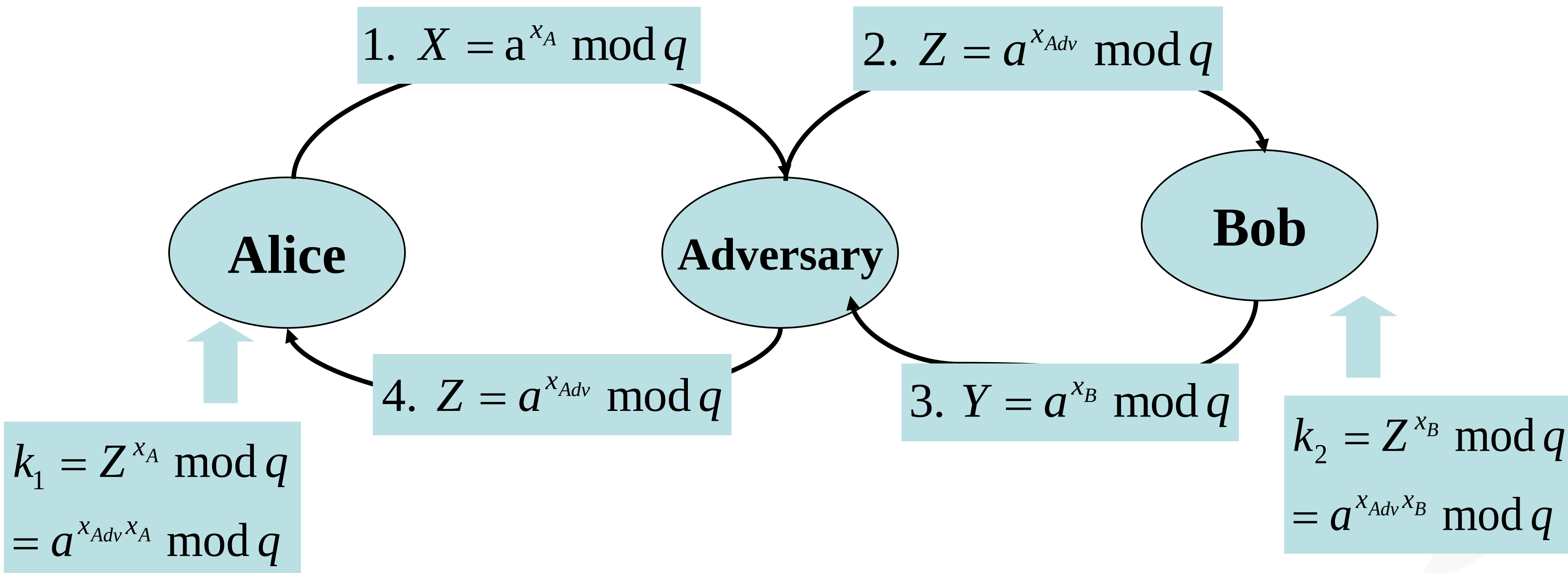
users Alice & Bob who wish to swap keys:

- agree on prime $q=353$ and $a=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:
  - $K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Issues on Anonymous D-H

- users could create random private/public D-H keys each time they communicate
- vulnerable to a **meet-in-the-Middle Attack**
- authentication of the keys is needed

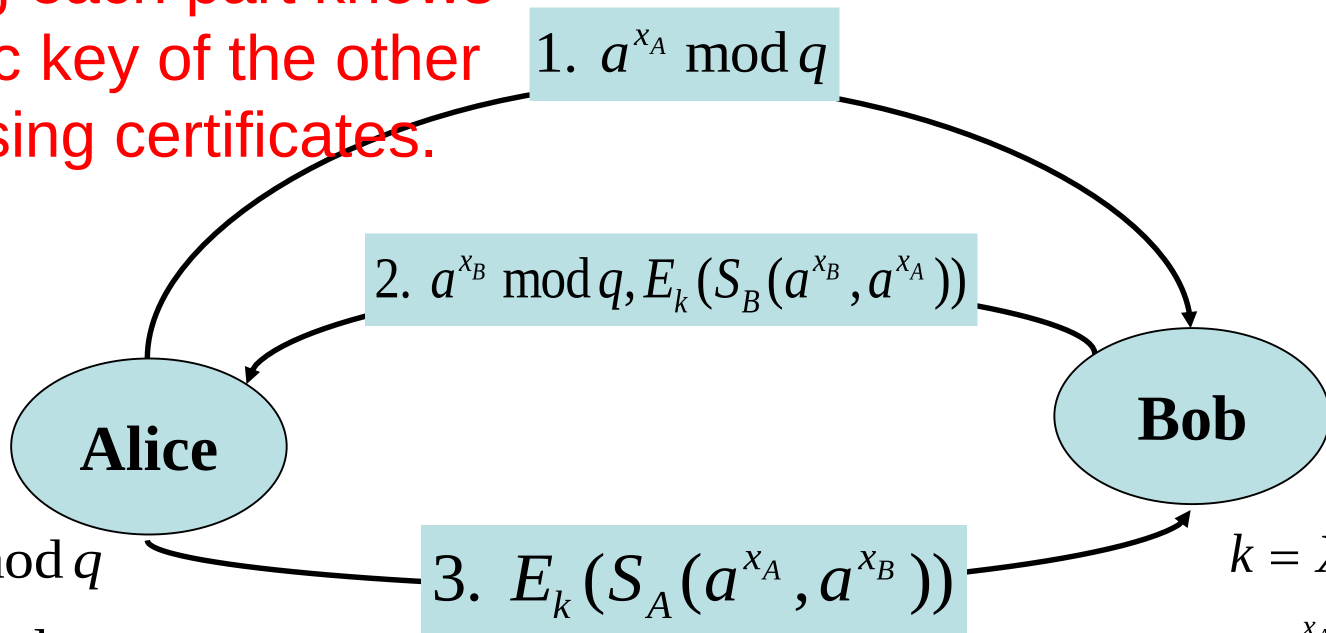
meet-in-the-Middle Attack



Especially, $x_{Adv}=p*x_A$

Station to Station (STS)

Assuming each part knows real public key of the other part by using certificates.



$$k = Y^{x_A} \bmod q = a^{x_A x_B} \bmod q$$

$$k = X^{x_B} \bmod q = a^{x_A x_B} \bmod q$$

one-time secret keys: x_A for Alice, x_B for Bob.
Private key: PR_A for Alice, PR_B for Bob
S_A denotes A’ s signature mechanism;

$$S_A(m)=E_{PR_A}(H(m))$$

Shamir's no-key protocol

Key transport withou
a priori shared keys

A Select a random session key ***k***

A

B

$E_A(k)$

$E_B[E_A(k)]$

$D_A[E_B[E_A(k)]] = E_B(k)$

$E_B(k)$

$D_B[E_B(k)] = k$

Lack of entity authentication

Encryption algorithms can be exchanged, $E_i[E_j(k)] = E_j[E_i(k)]$ **e.g. RSA. Firstly, A and B randomly select secret number a,b respectively. Then,**

$$A \to B : K^a \bmod p \quad (1)$$

$$A \leftarrow B : (K^a)^b \bmod p \quad (2)$$

$$A \to B : (K^{ab})^{a^{-1}} \bmod p \quad (3)$$

Summary of Key Distribution

- session key
  - temporary key
  - used for encryption of data between users
  - for one logical session then discarded
  - By symmetric or public encryption
- master key
  - used to encrypt session keys
  - shared by user & key distribution center for long time
  - Generated by Key Pre-distribution, public encryption, password

Summary

- **Discuss the concept of a key hierarchy.**
- **Understand how to distribute symmetric keys.**

Key Terms

- end-to-end encryption
- key distribution
- key distribution center (KDC)key management
- man-in-the-middle attackmaster key
- nonce
- Diffie–Hellman key exchangediscrete logarithm

Review Questions

- 14.1 Explain why man-in-the-middle attacks are ineffective on the secret key distribution protocol discussed in Figure 14.3.
- 14.2 What is the major issue in end to end key distribution? How does the key hierarchy concept address that issue?
- 14.3 What is a nonce?

- What's the difference between session key and master key?

Problems

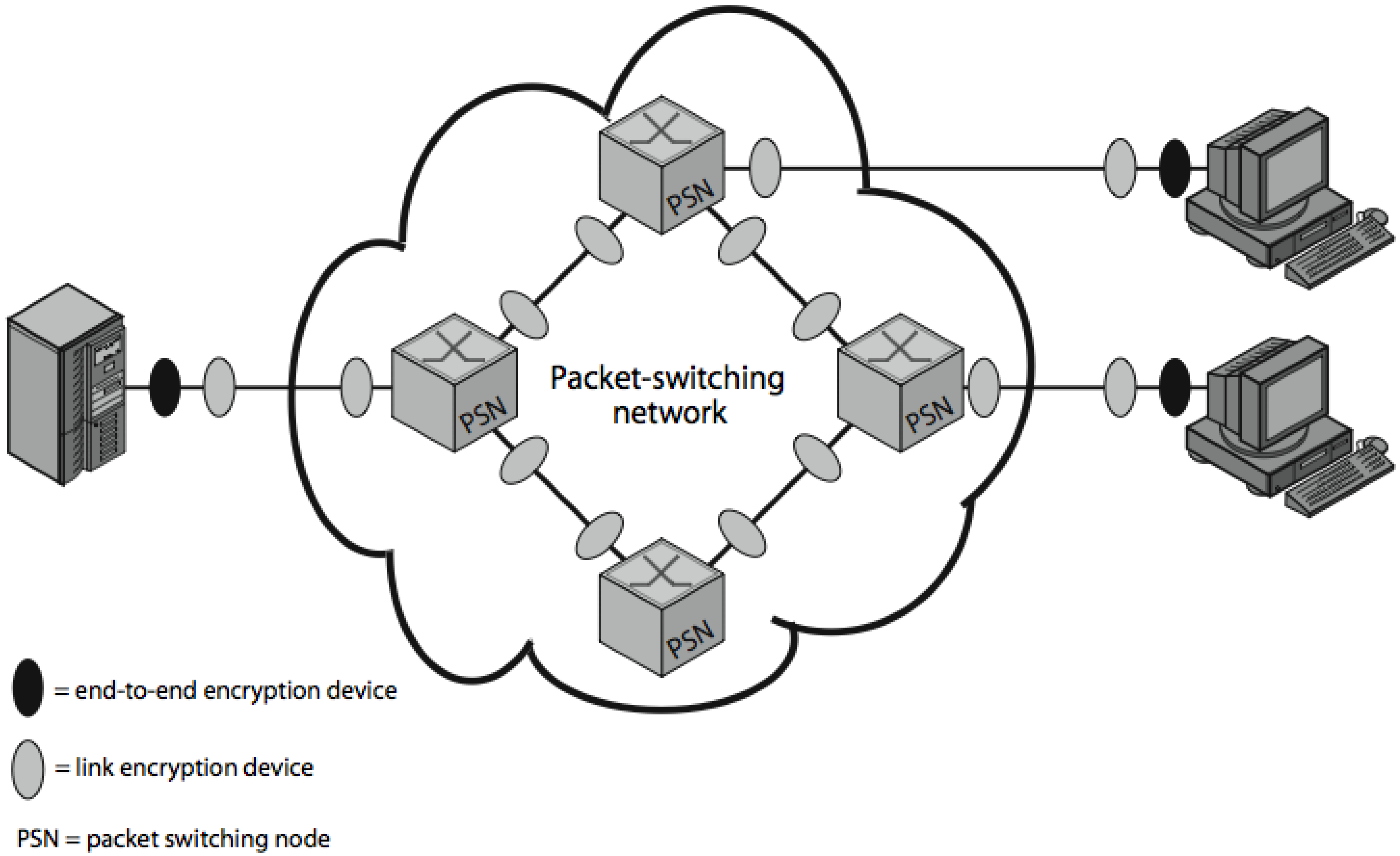
- **14.1**
- **14.2**

Thanks!

- **Key Factors of Secure Communication**
  - **Where Encrypt?**
  - **What to Encrypt?**
  - **Select Cipher and Use**
- **Secure Communication**

- Key Factors of Secure Communication
  - Where Encrypt?
  - What to Encrypt?
  - Select Cipher and Use
- Secure Communication

Where Encryption?



What to Encrypt?

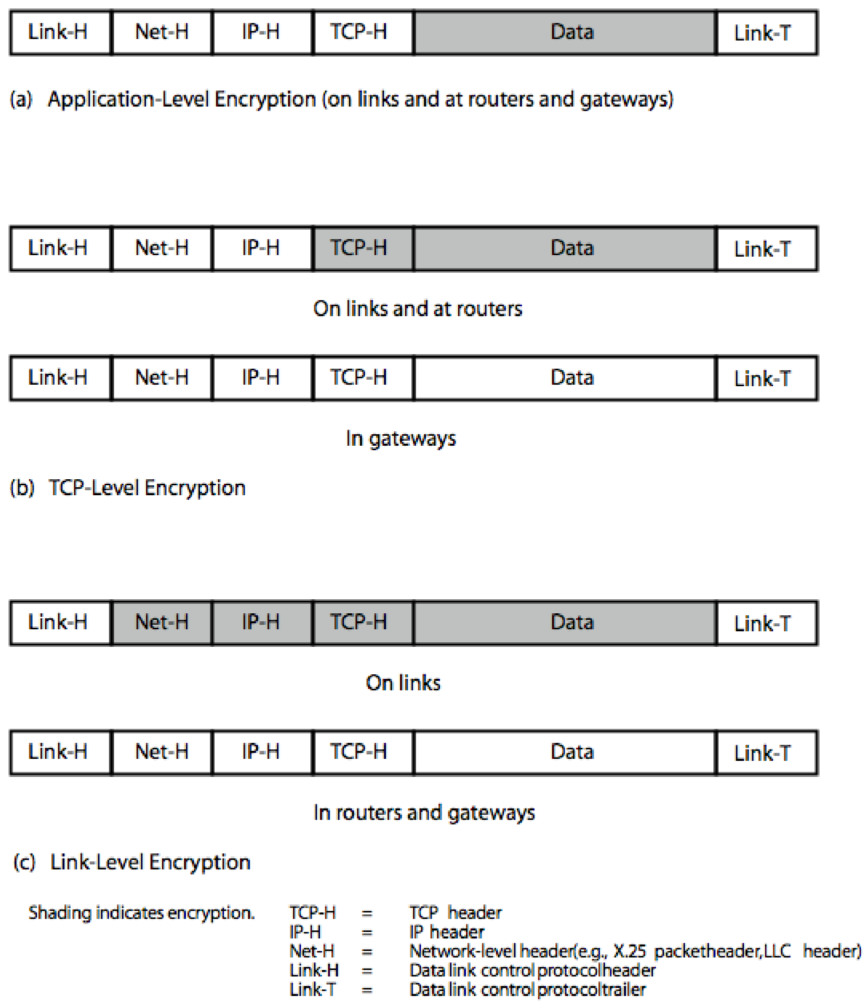
Link-H	Net-H	IP-H	TCP-H	Data	Link-T

Shading indicates encryption.

TCP-H = TCP header
IP-H = IP header
Net-H = Network-level header (e.g., X.25 packet header, LLC header)
Link-H = Data link control protocol header
Link-T = Data link control protocol trailer

- **Link Encryption：packet**
- **End-to-End Encryption:**
  - **Data:user data**
  - **TCP-H+Data**
  - **IP-H+TCP-H+Data**
  - **Net-H+IP-H+TCP-H+Data**

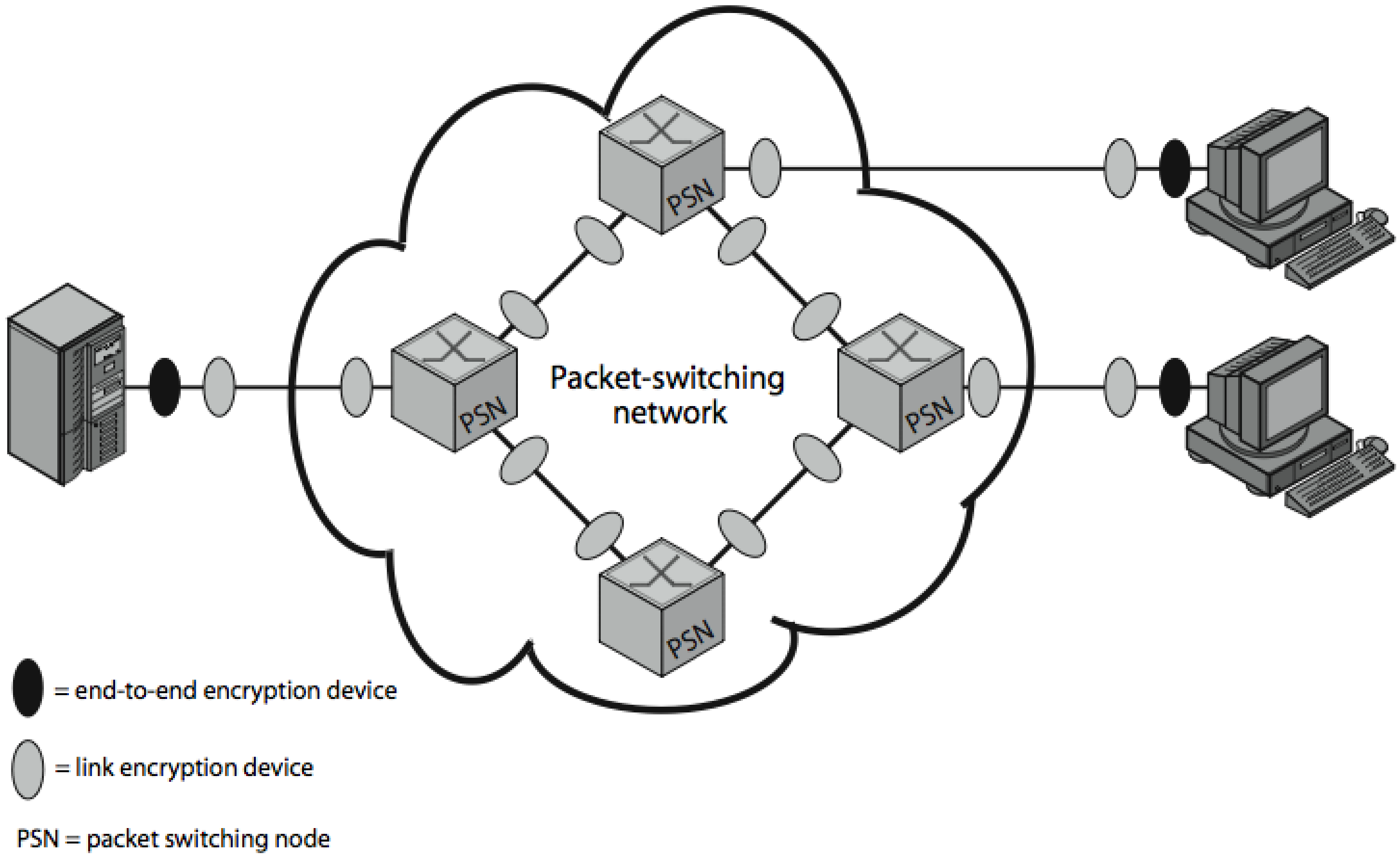
Encryption vs. Protocol Level



Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending host	Message encrypted in sending host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending host	Applied by sending process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done in hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

- **Key Factors of Secure Communication**
  - **Where Encrypt?**
  - **What to Encrypt?**
  - **Select Cipher and Use**
- **Secure Communication**

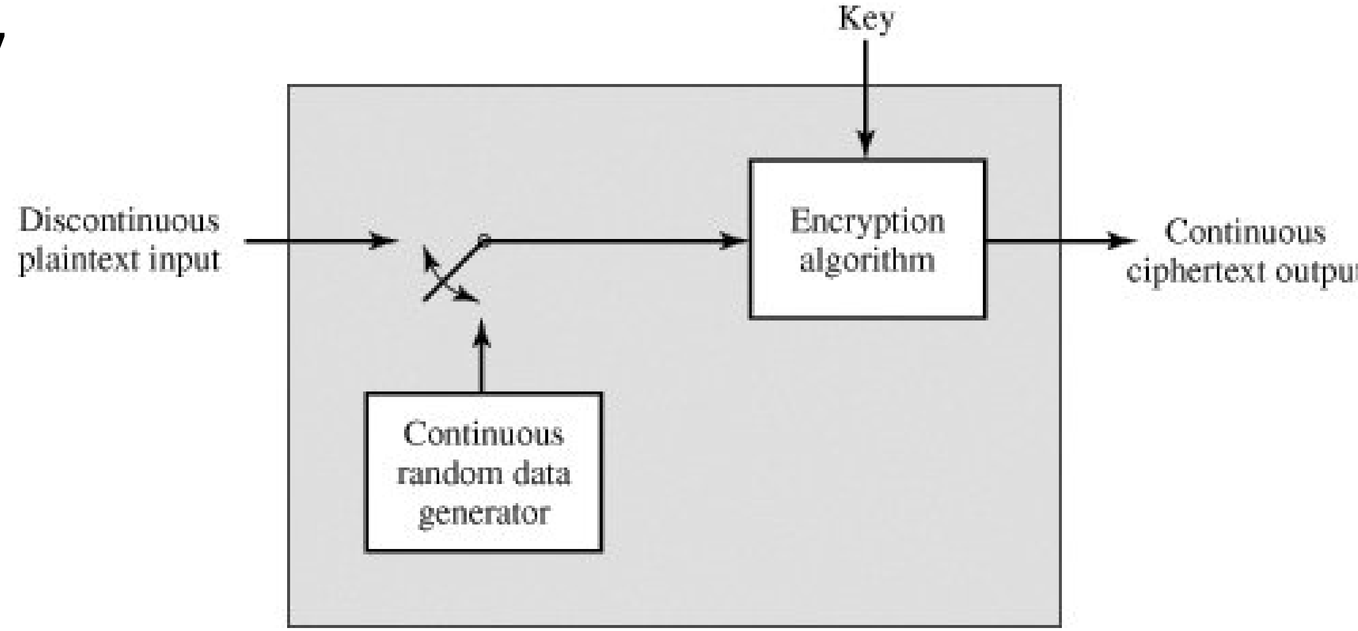
Secure communication



Measures against Traffic Analysis

- **Link Encry**

Figure 7.6. Traffic-Padding Encryption Device



- **End-to-End Encryption:**
  - pad out data units to a uniform length at either the transport or application level.
  - null messages can be inserted randomly into the stream