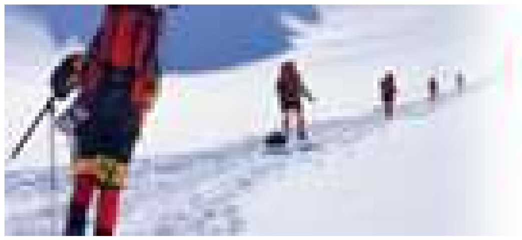


Data Communications and Networking Fourth Edition

Forouzan

Chapter 31

Network Security

31-1-1 SECURITY GOALS

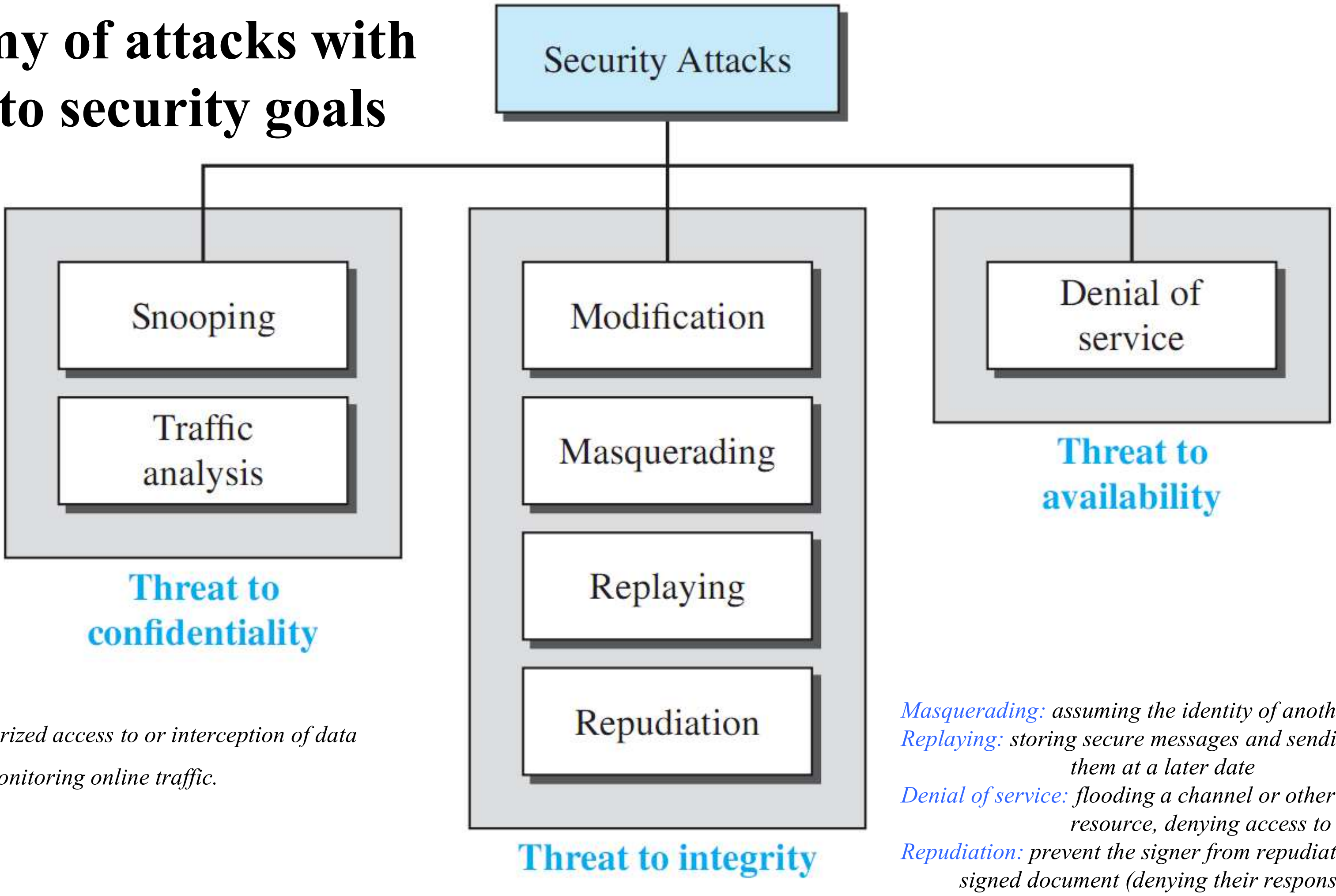
Confidentiality: ***Protect our confidential information from unauthorized access. Confidentiality not only applies to the storage of information, it also applies to the transmission of information.***

Integrity: ***changes can only be done by authorized entities and through authorized mechanisms.***

Availability: ***Information created and stored by an organization needs to be available to authorized entities.***

31-1-1 ATTACKS

Taxonomy of attacks with relation to security goals



Snooping: *unauthorized access to or interception of data*

Traffic analysis: *monitoring online traffic.*

Masquerading: assuming the identity of another user
Replaying: storing secure messages and sending them at a later date
Denial of service: flooding a channel or other resource, denying access to others
Repudiation: prevent the signer from repudiating a signed document (denying their responsibility)

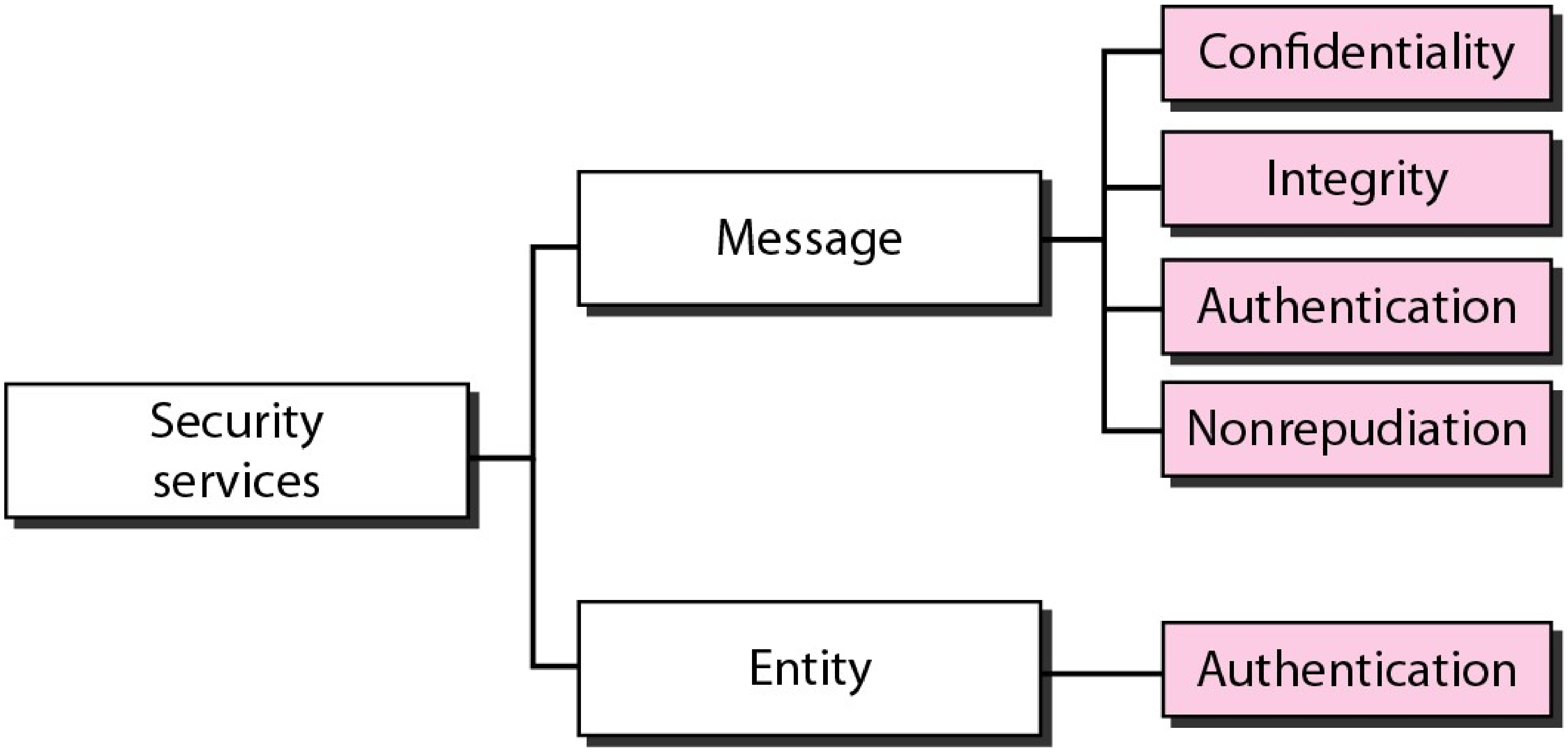
31-1-3 SECURITY SERVICES

Network security can provide five services. Four of these services are related to the message exchanged using the network. The fifth service provides entity authentication or identification.

Topics discussed in this section:

Message Confidentiality
Message Integrity
Message Authentication
Message Nonrepudiation
Entity Authentication

Figure 31.1 ***Security services related to the message or entity***



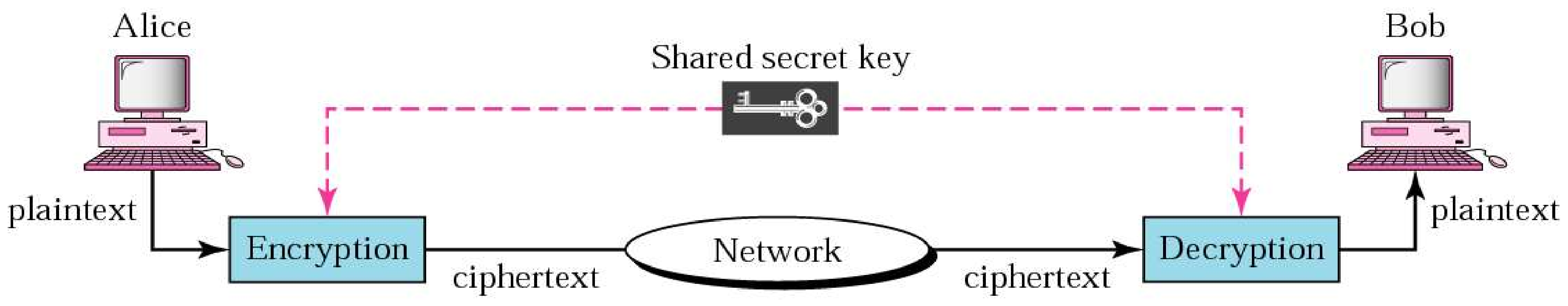
31-2 MESSAGE CONFIDENTIALITY

The concept of how to achieve message confidentiality or privacy has not changed for thousands of years. The message must be encrypted at the sender site and decrypted at the receiver site. This can be done using either symmetric-key cryptography or asymmetric-key cryptography.

Topics discussed in this section:

Confidentiality with Symmetric-Key Cryptography
Confidentiality with Asymmetric-Key Cryptography

Figure 31.2 *General idea of symmetric-key cipher*



Traditional symmetric-key ciphers:

Substitution ciphers (monoalphabetic or polyalphabetic)

Transposition ciphers

Modern symmetric-key ciphers:

DES, Triple-DES, AES, …

Figure 31.4 ***Monoalphabetic ciphers:*** *Representation of plaintext and ciphertext characters in modulo 26*

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example 31.1

Use the additive cipher with key = 15 to encrypt the message “hello”.

Solution

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

The result is “WTAAD”. Note that the cipher is monoalphabetic because two instances of the same plaintext character (*l*) are encrypted as the same character (*A*).

Example 31.2

Use the additive cipher with key = 15 to decrypt the message “WTAAD”.

Solution

We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22	Decryption: (22 – 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 – 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 – 15) mod 26	Plaintext: 14 → o

The result is “hello”. Note that the operation is in modulo 26, which means that we need to add 26 to a negative result (for example −15 becomes 11).

Additive ciphers are vulnerable to attacks using exhaustive key searches (brute-force attacks)

Polyalphabetic ciphers: *each occurrence of a character may have a different substitute*

Example:* a simple polyalphabetic cipher called *autokey cipher

$P = P_1P_2P_3 \ldots$ $C = C_1C_2C_3 \ldots$ $k = (k_1, P_1, P_2, \ldots)$

Encryption: $C_i = (P_i + k_i) \bmod 26$ Decryption: $P_i = (C_i - k_i) \bmod 26$

Example 31.4

Assume that Alice and Bob agreed to use an autokey cipher with initial key value $k_1 = 12$. Now Alice wants to send Bob the message "Attack is today".

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	*12*	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	**M**	**T**	**M**	**T**	**C**	**M**	**S**	**A**	**L**	**H**	**R**	**D**	**Y**

How to decrypt?

***Transposition Ciphers**: does not substitute one symbol for another; instead, it changes the location of the symbols.*

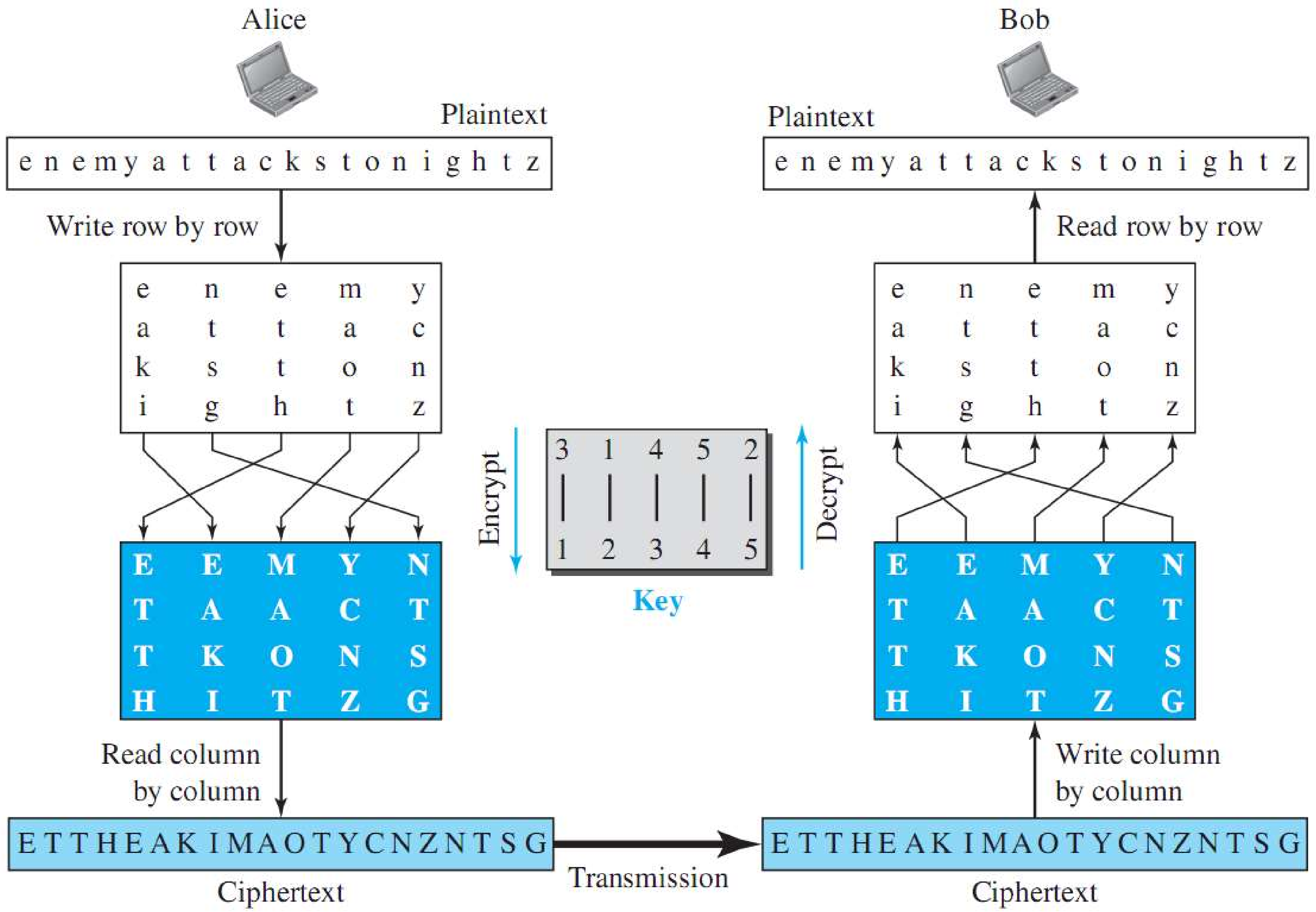


Figure 31.8 *Components of a modern block cipher*

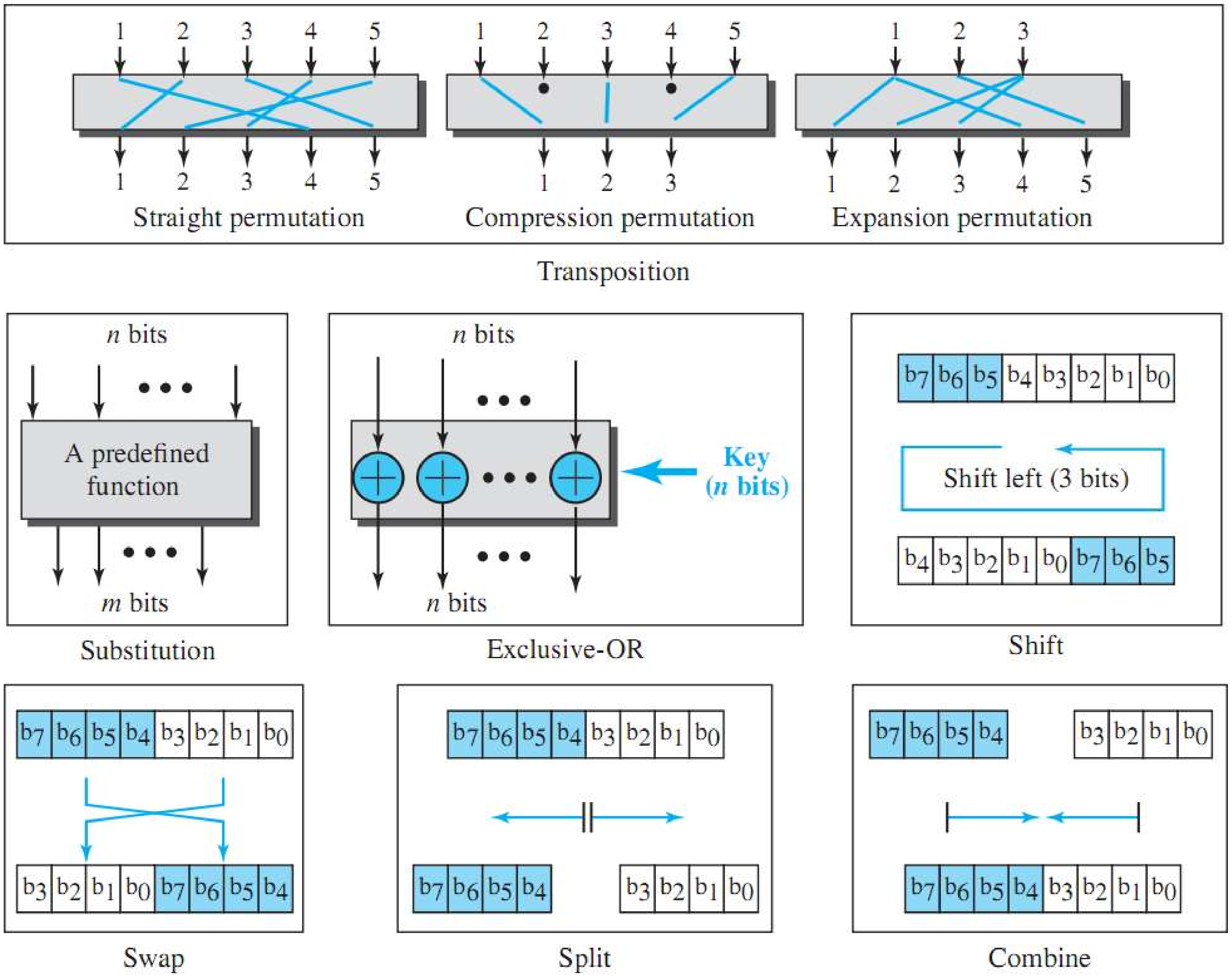
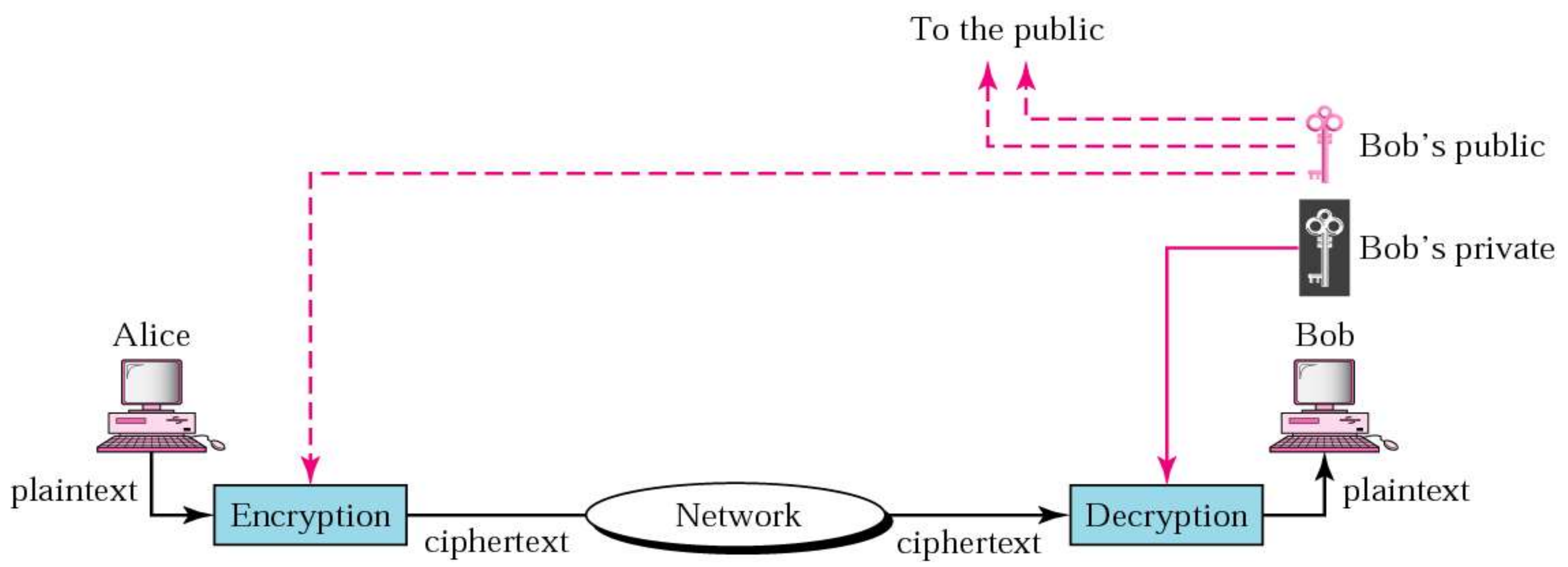


Figure 31.3 *Message confidentiality using asymmetric keys*



31-3-1 MESSAGE INTEGRITY

Encryption and decryption provide secrecy, or confidentiality, but not integrity. However, on occasion we may not even need secrecy, but instead must have integrity.

Topics discussed in this section:

Document and Fingerprint
Message and Message Digest
Creating and Checking the Digest
Hash Function Criteria
Hash Algorithms: SHA-1

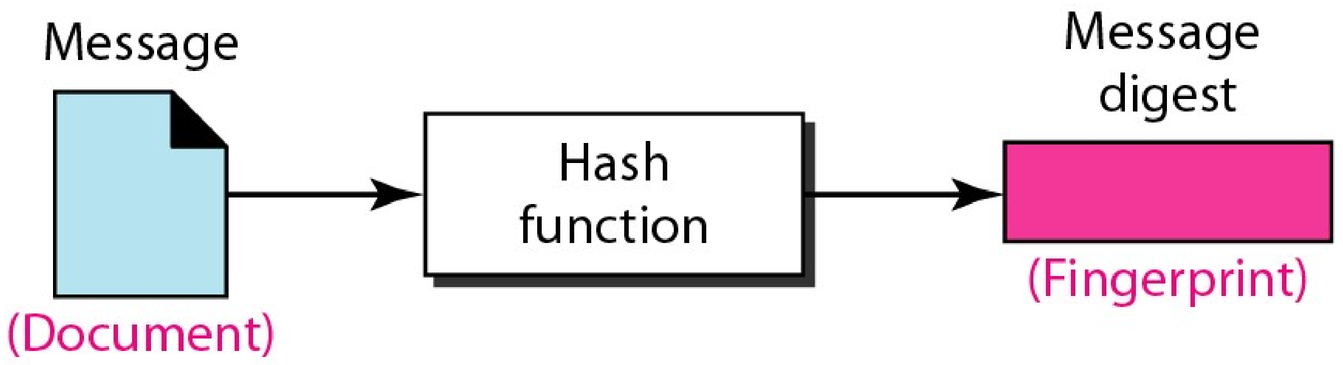
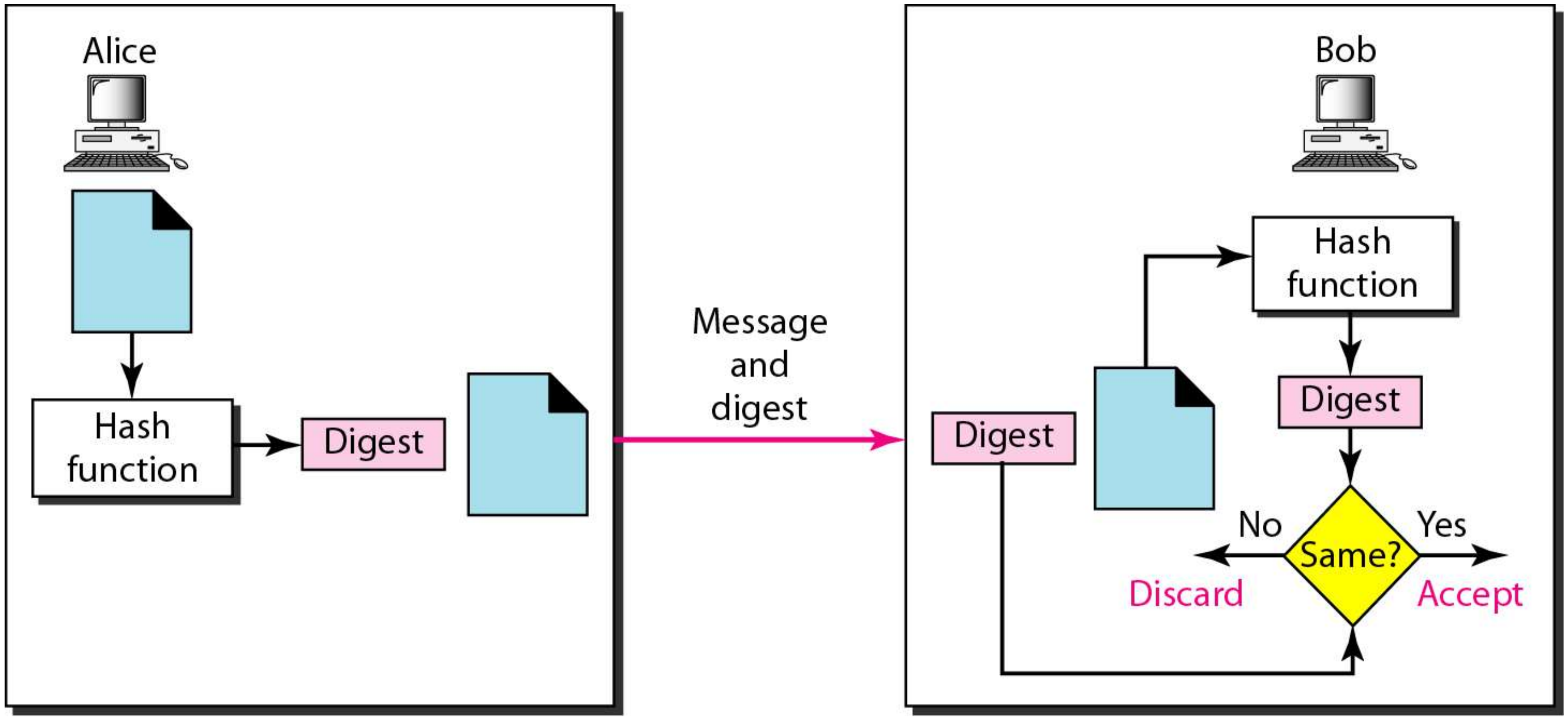


Figure 31.4 *Message and message digest*

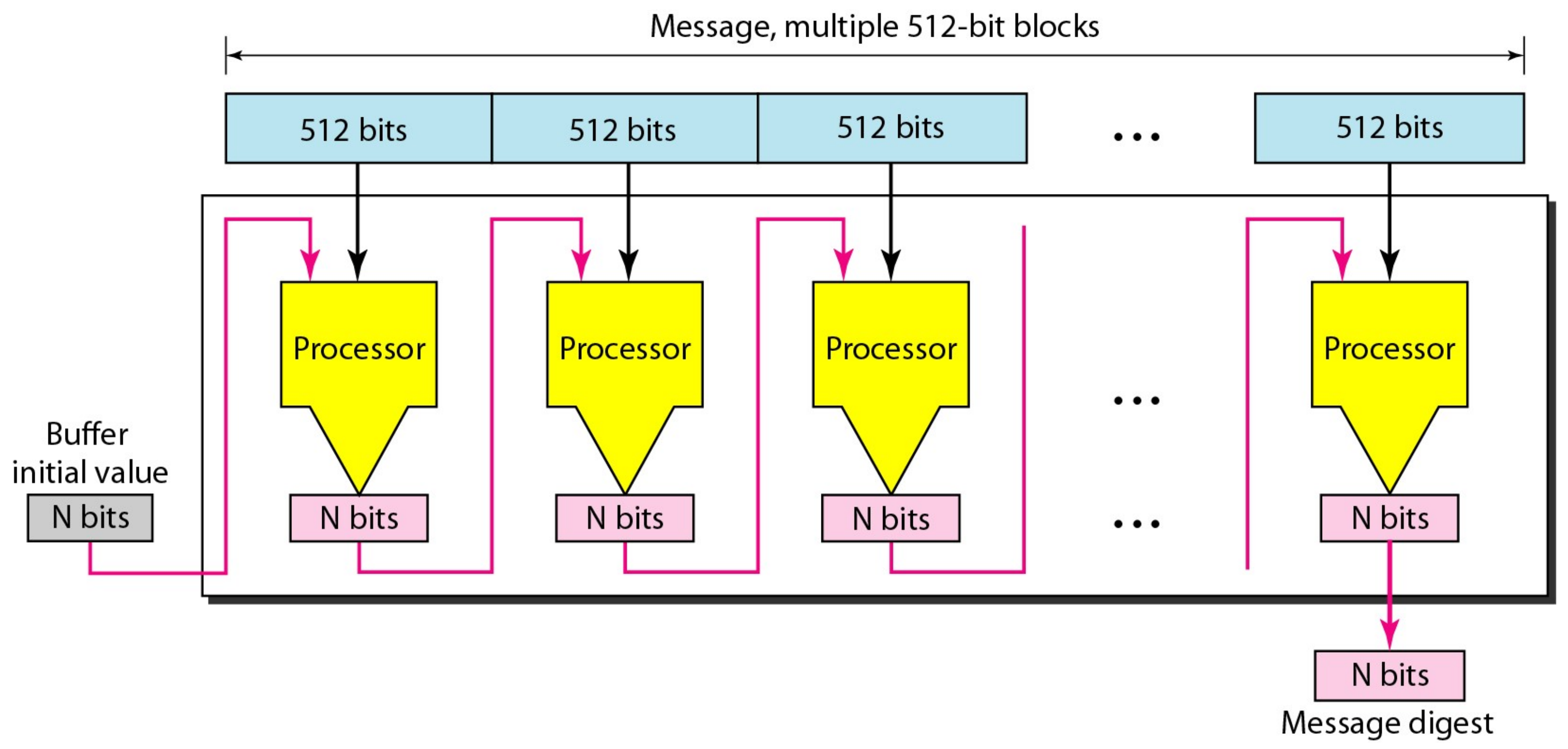
To preserve the integrity of a document, both the document and the fingerprint are needed.

Figure 31.5 *Checking integrity*



The message digest needs to be kept secret.

Figure 31.7 *Message digest creation*



Note

SHA-1 hash algorithms create an N-bit message digest out of a message of 512-bit blocks.

SHA-1 has a message digest of 160 bits (5 words of 32 bits).

SHA : Secure Hash Algorithm

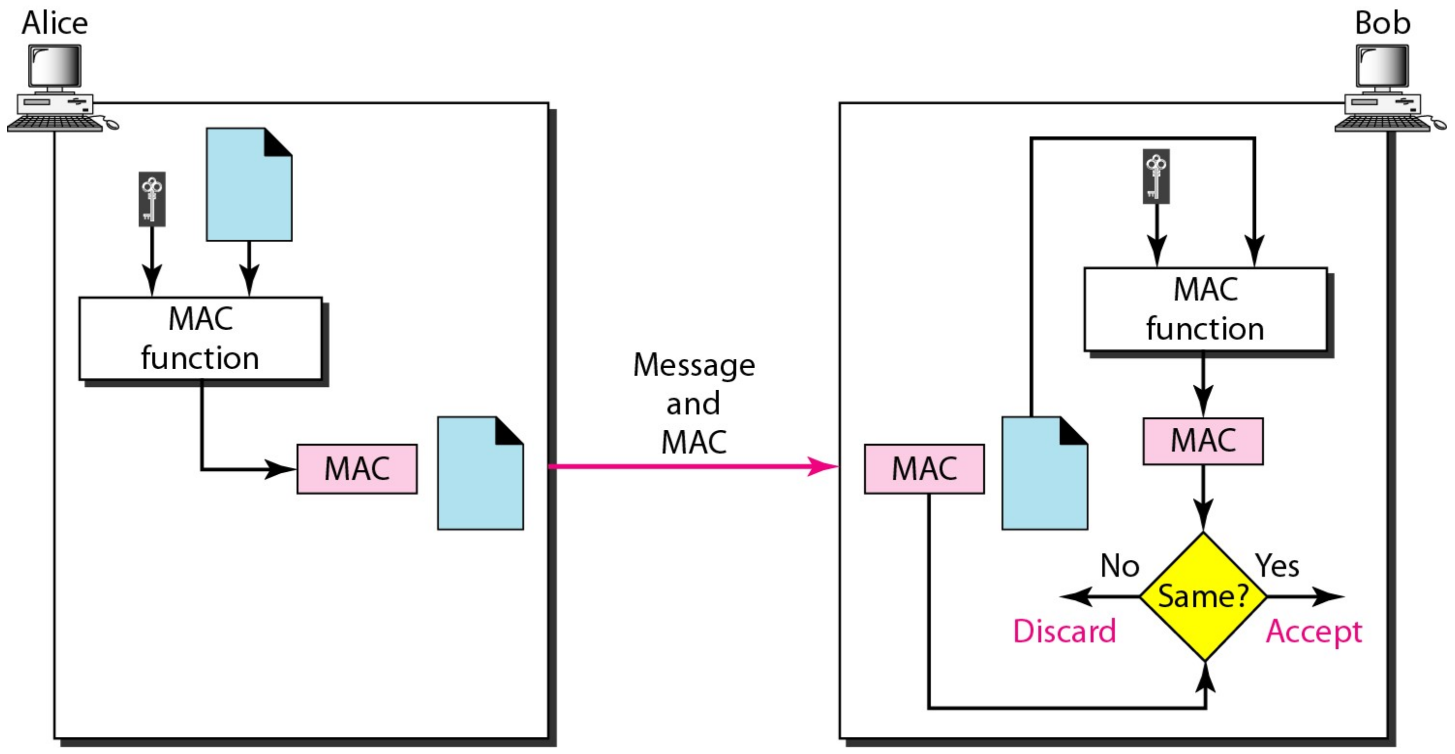
31-3-2 MESSAGE AUTHENTICATION

A hash function cannot provide authentication. The digest created by a hash function can detect any modification in the message, but not authentication.

Topics discussed in this section:

MAC: Message Authentication Code

Figure 31.9 *MAC, created by Alice and checked by Bob*



MAC provides message integrity and message authentication using a combination of a hash function and a secret.

31-3-3 DIGITAL SIGNATURE

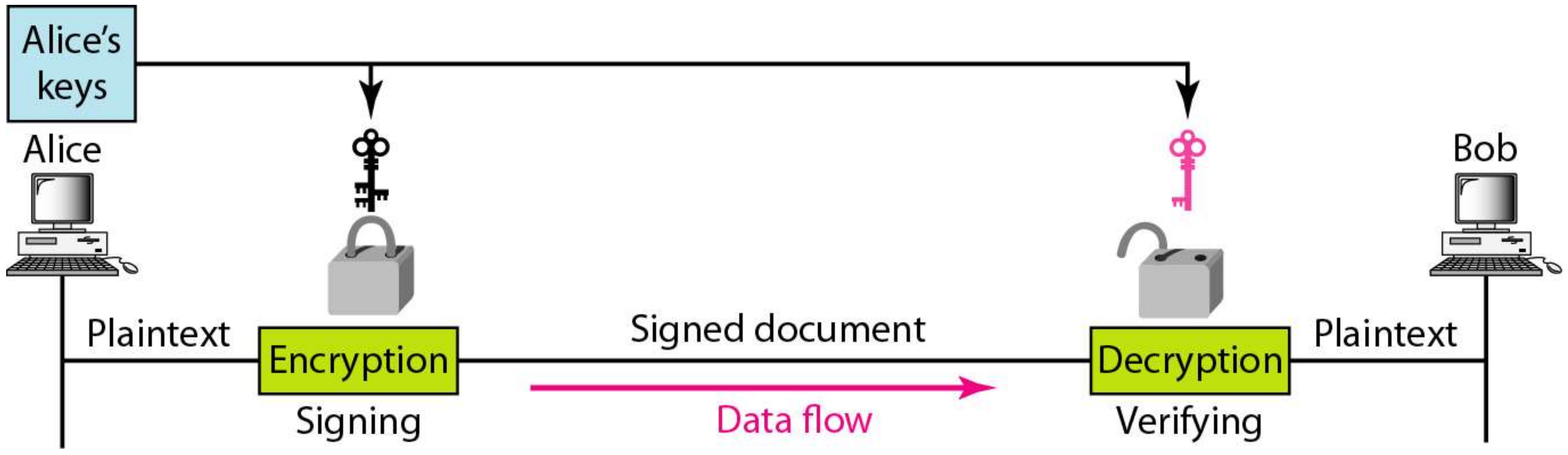
When Alice sends a message to Bob, Bob needs to check the authenticity of the sender; he needs to be sure that the message comes from Alice and not Eve. Bob can ask Alice to sign the message electronically. In other words, an electronic signature can prove the authenticity of Alice as the sender of the message. We refer to this type of signature as a digital signature.

Topics discussed in this section:

Comparison
Need for Keys
Process

Figure 31.11 *Signing the message itself in digital signature*

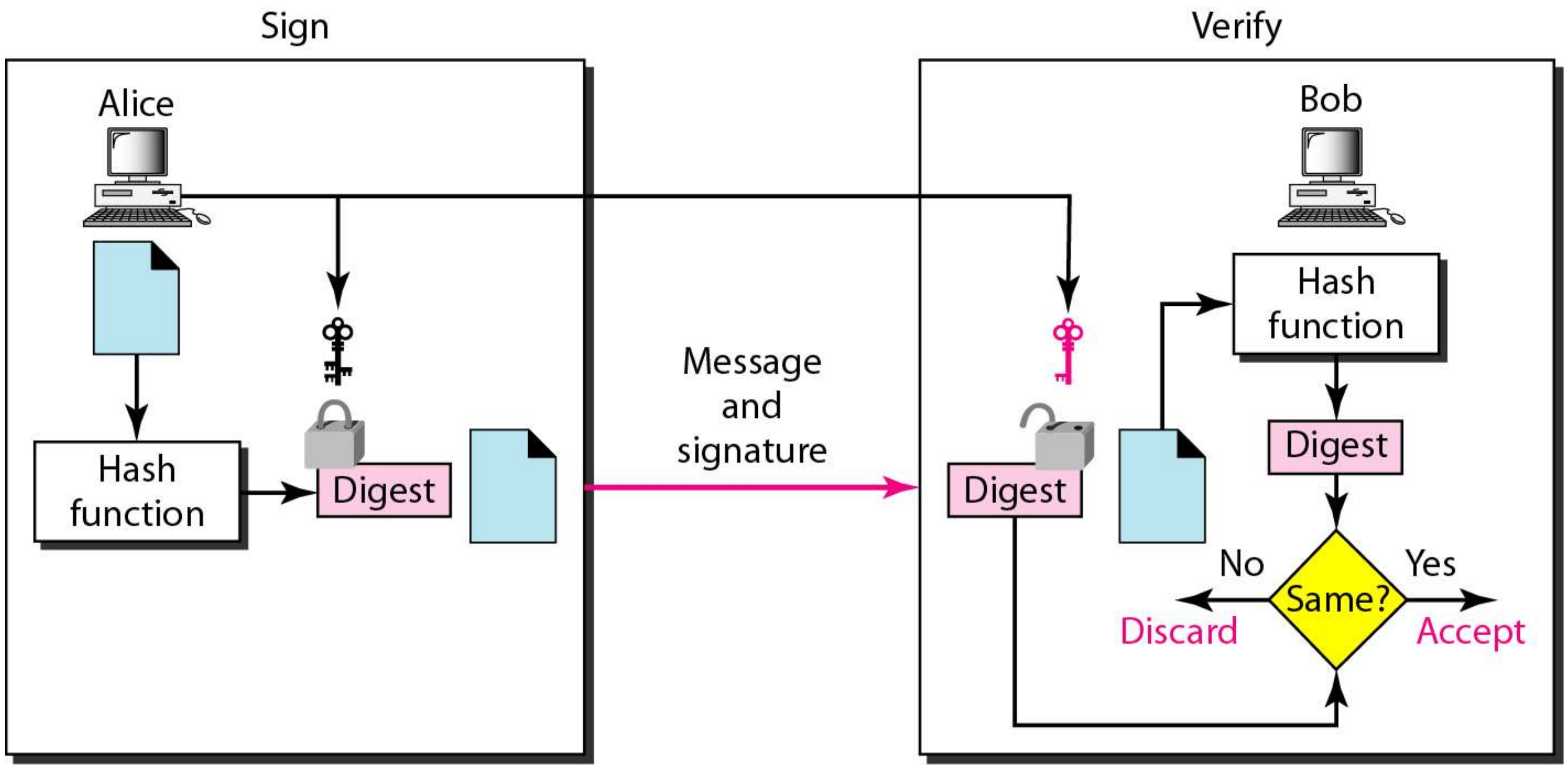
A digital signature uses a pair of private-public keys.



Note

**In a cryptosystem, we use the private and public keys of the receiver;
in digital signature, we use the private and public keys of the sender.**

Figure 31.12 *Signing the digest in a digital signature*



Note

A digital signature today provides message integrity.

Digital signature provides message authentication.

Nonrepudiation

If Alice signs a message and then denies it, can Bob later prove that Alice actually signed it? For example, if Alice sends a message to a bank (Bob) and asks to transfer $10,000 from her account to Ted's account, can Alice later deny that she sent this message?

With the scheme we have presented so far, Bob might have a problem. Bob must keep the signature on file and later use Alice's public key to create the original message to prove the message in the file and the newly created message are the same. This is not feasible because Alice may have changed her private or public key during this time; she may also claim that the file containing the signature is not authentic.

Nonrepudiation can be provided using a trusted party.

31-3-4 ENTITY AUTHENTICATION

Entity authentication is a technique designed to let one party prove the identity of another party. An entity can be a person, a process, a client, or a server. The entity whose identity needs to be proved is called the claimant; the party that tries to prove the identity of the claimant is called the verifier.

Topics discussed in this section:

Passwords
Challenge-Response

Note

The simplest and oldest method of entity authentication is the use of a *password*.

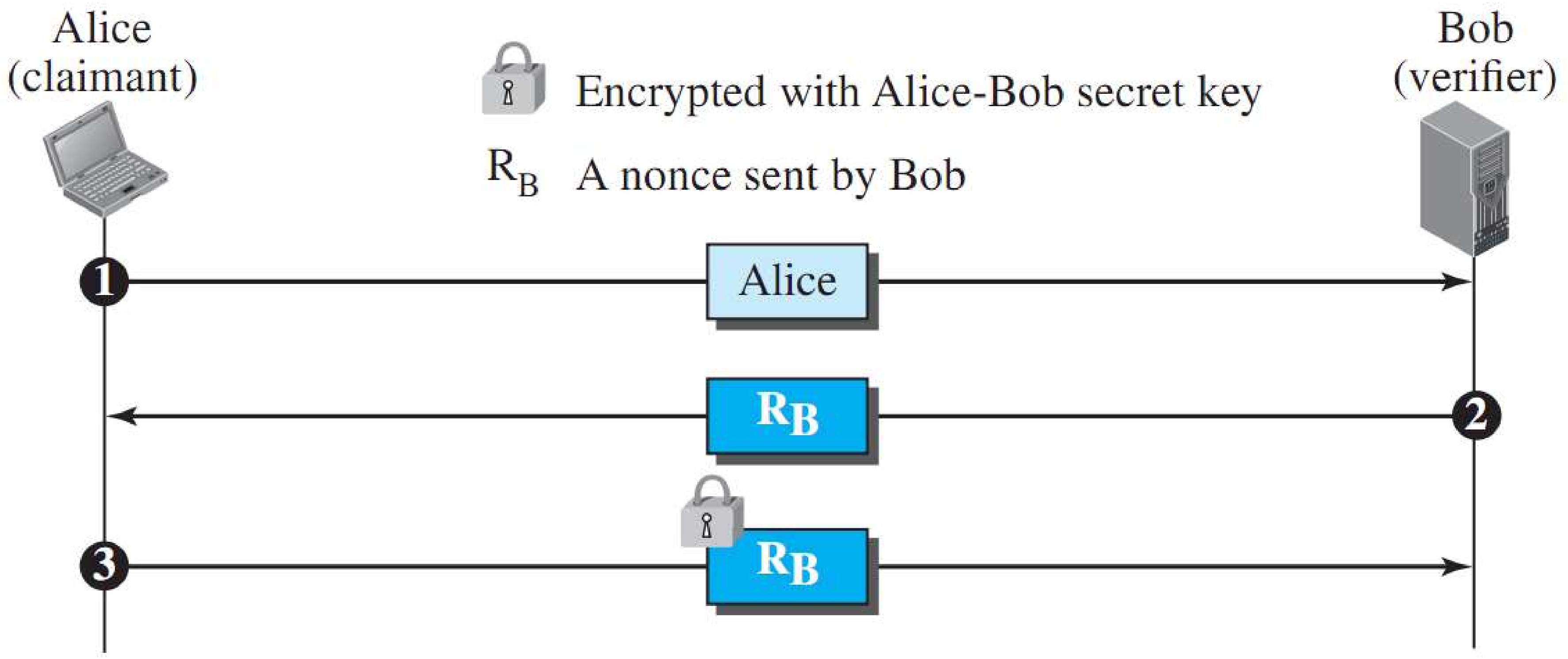
Passwords are very prone to attacks. A password can be stolen, intercepted, guessed, and so on.

Note

In challenge-response authentication, the claimant proves that she knows a secret without revealing it.

The challenge is a time-varying value sent by the verifier; the response is the result of a function applied to the challenge.

Figure 31.22 *Unidirectional, symmetric-key authentication*



Alice
(claimant)
Encrypted with Alice-Bob secret key
R_B A nonce sent by Bob
Bob
(verifier)
1
Alice
2
R_B
3
R_B

Figure 31.23 *Unidirectional, asymmetric-key authentication*

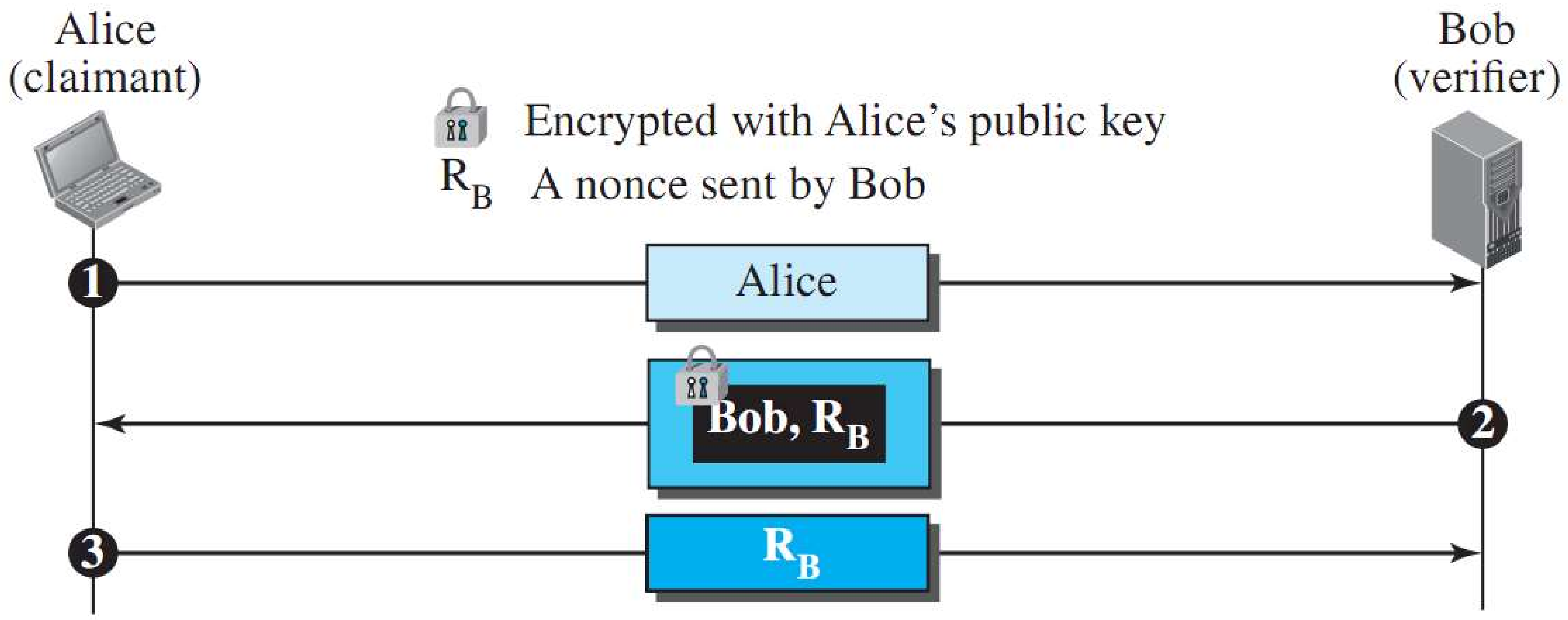
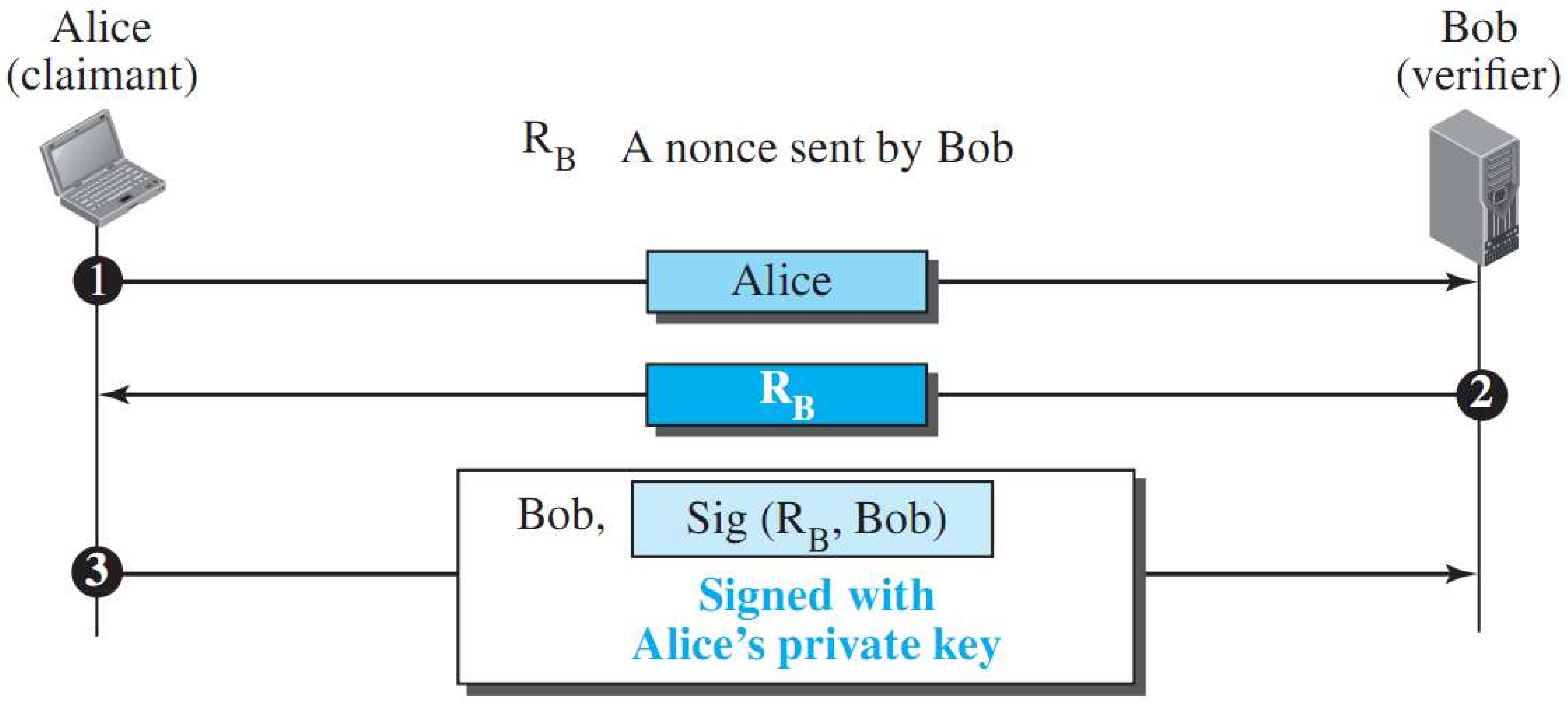


Figure 31.24 *Authentication, using digital signature*



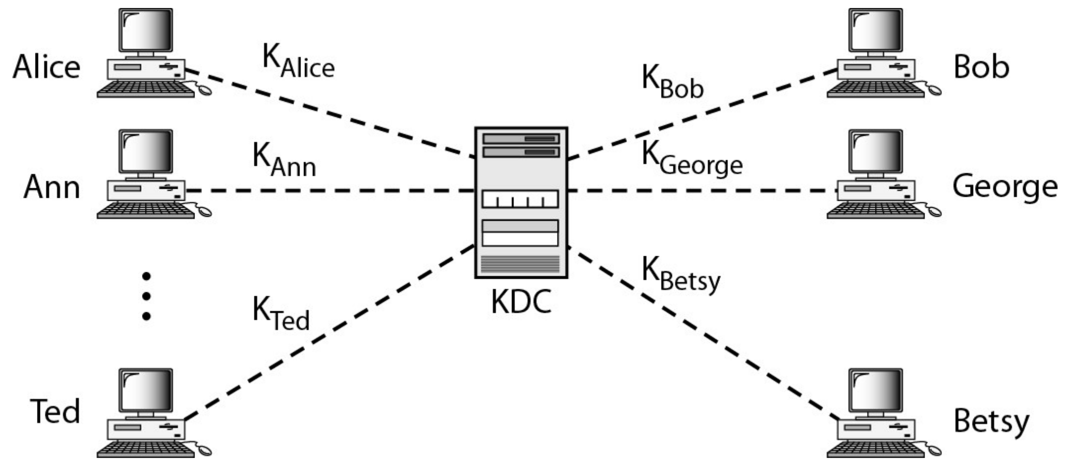
31-7 KEY MANAGEMENT

We never discussed how secret keys in symmetric-key cryptography and how public keys in asymmetric-key cryptography are distributed and maintained. In this section, we touch on these two issues. We first discuss the distribution of symmetric keys; we then discuss the distribution of asymmetric keys.

Topics discussed in this section:

Symmetric-Key Distribution
Public-Key Distribution

Figure 31.25 *Key Distribution Center (KDC)*



Hierarchical multiple KDC

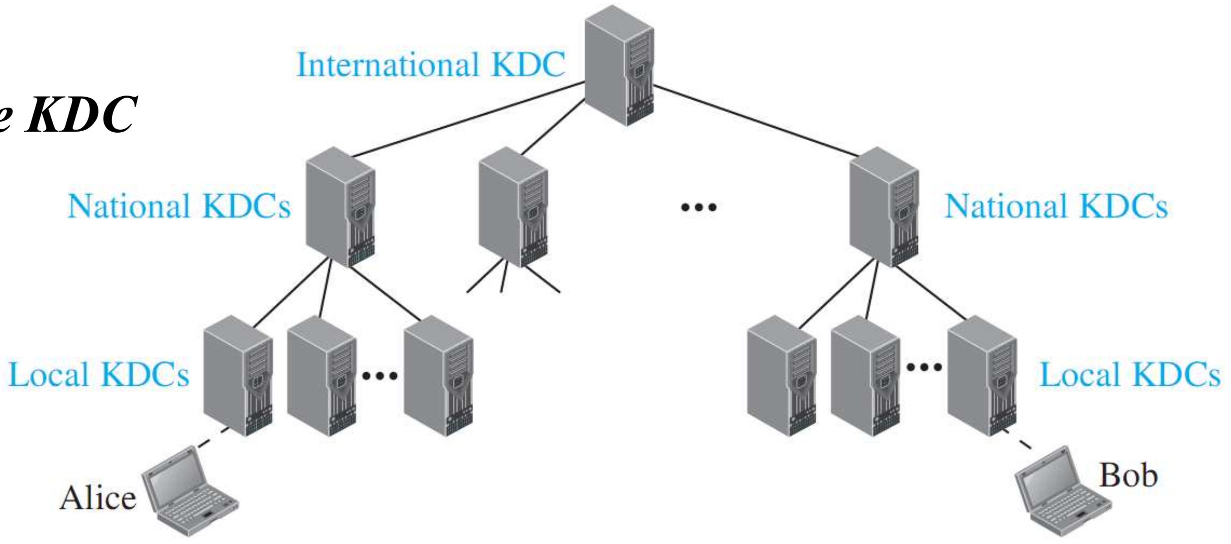
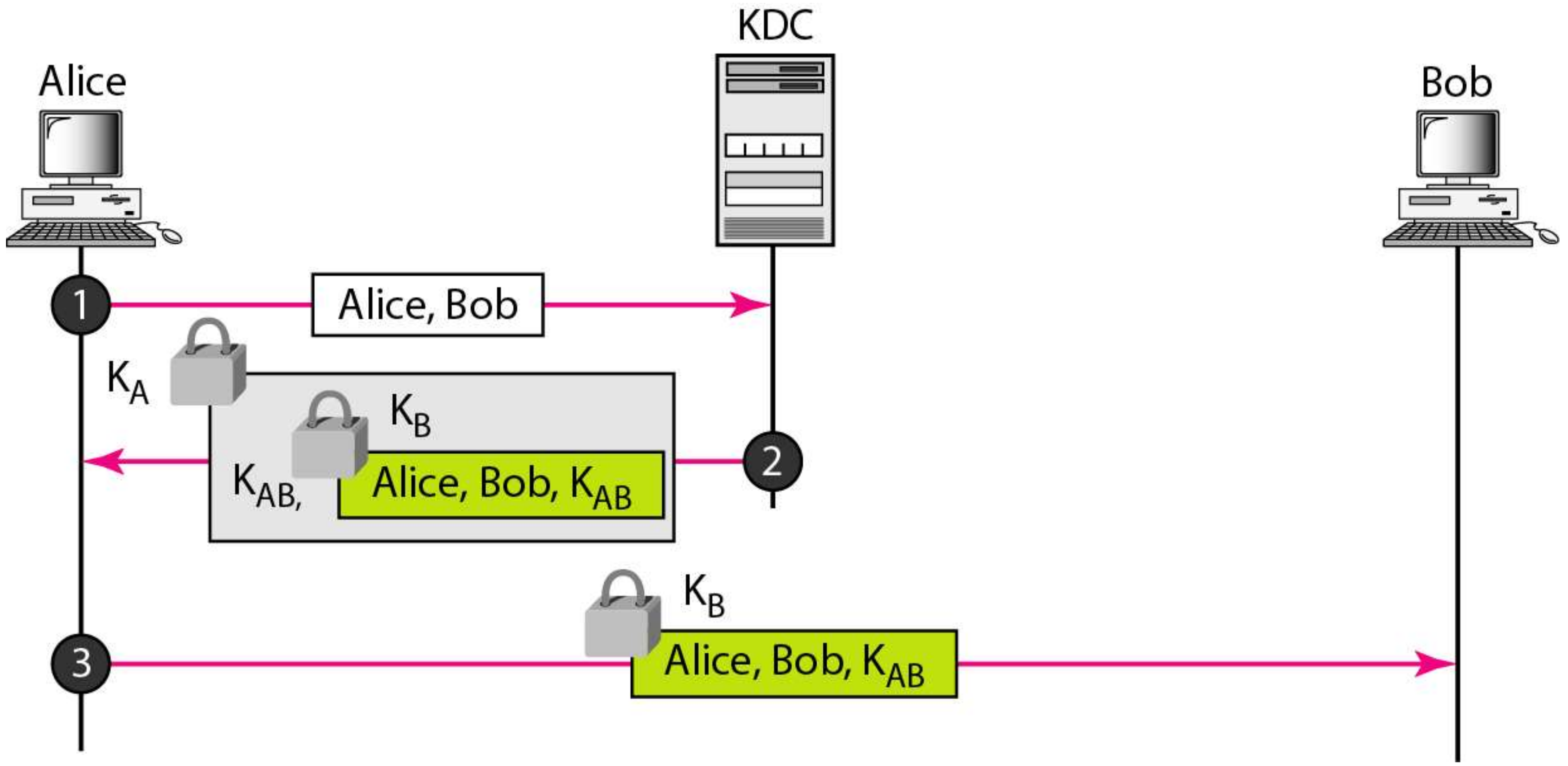


Figure 31.26 ***Creating a session key between Alice and Bob using KDC***

A session symmetric key between two parties is used only once.


KDC
Alice
Bob
1
Alice, Bob
K_A
K_B
2
K_{AB},
Alice, Bob, K_{AB}
K_B
3
Alice, Bob, K_{AB}