Module 03

Network Security

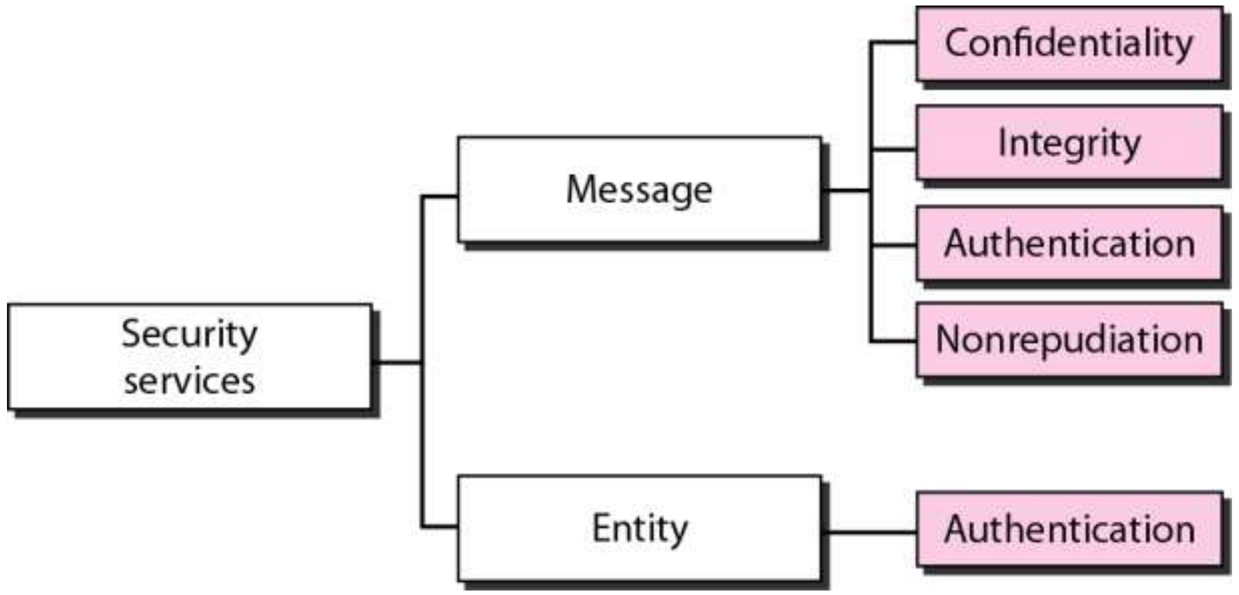
31-1 SECURITY SERVICES

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*



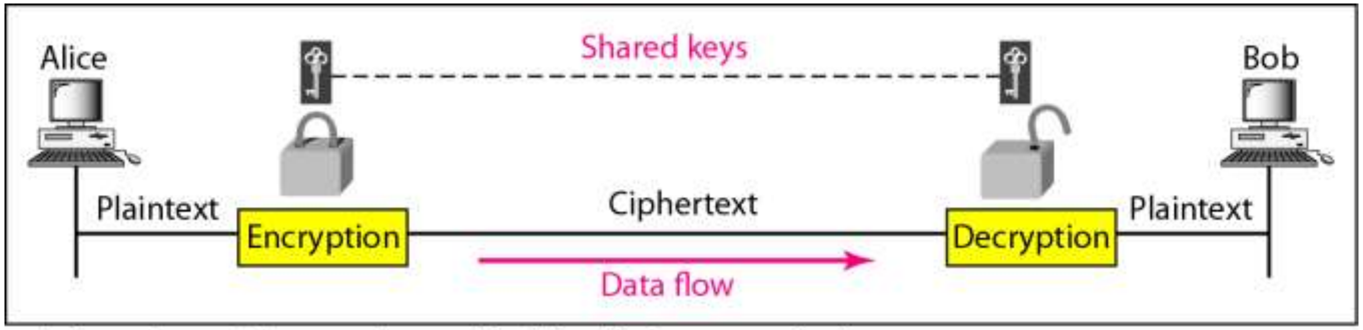
31-2 MESSAGE CONFIDENTIALITY

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.

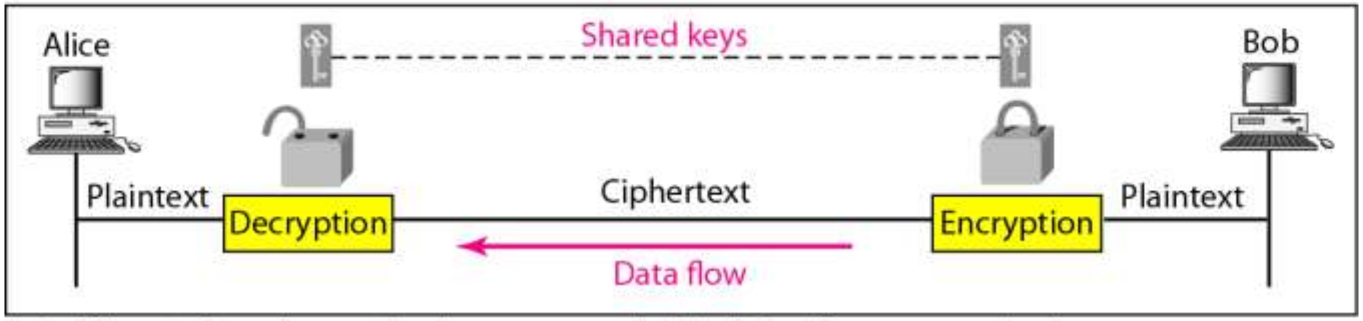
Topics discussed in this section:

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

Figure 31.2 *Message confidentiality using symmetric keys in two directions*

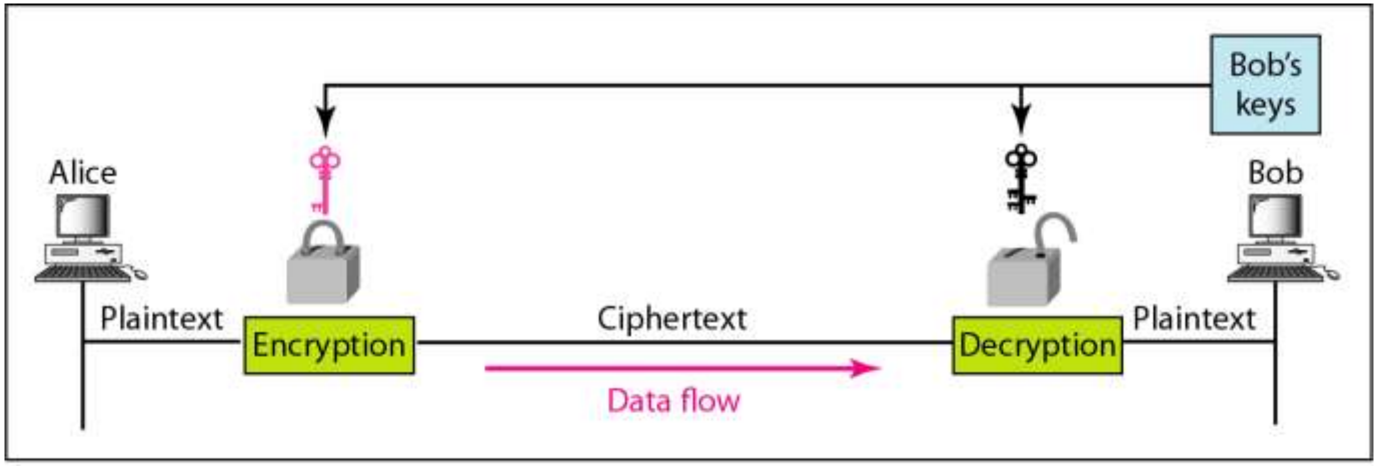


a. A shared secret key can be used in Alice-Bob communication

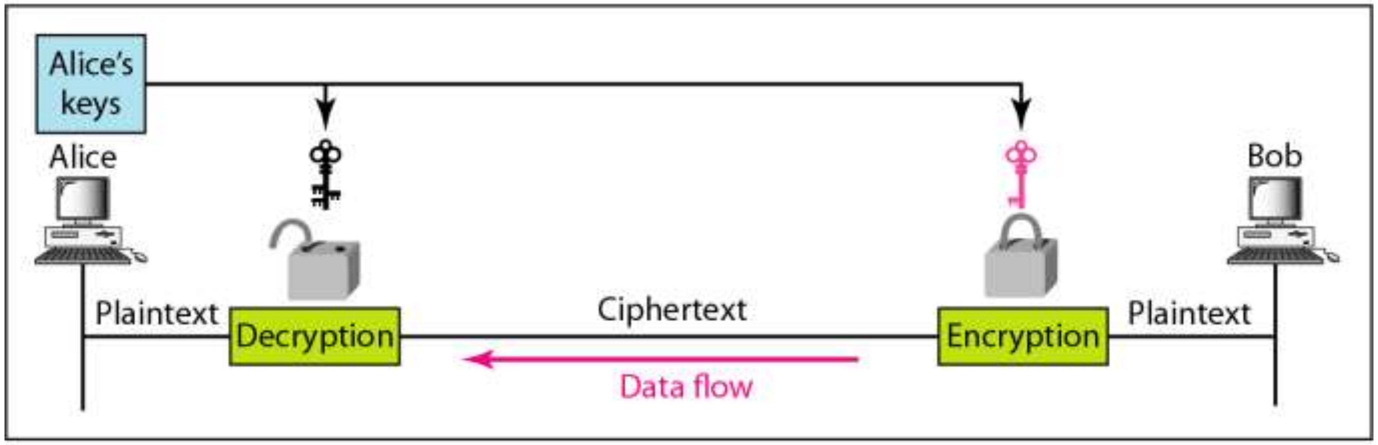


b. A different shared secret key is recommended in Bob-Alice communication

Figure 31.3 *Message confidentiality using asymmetric keys*



a. Bob's keys are used in Alice-Bob communication



b. Alice's keys are used in Bob-Alice communication

31-3 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.

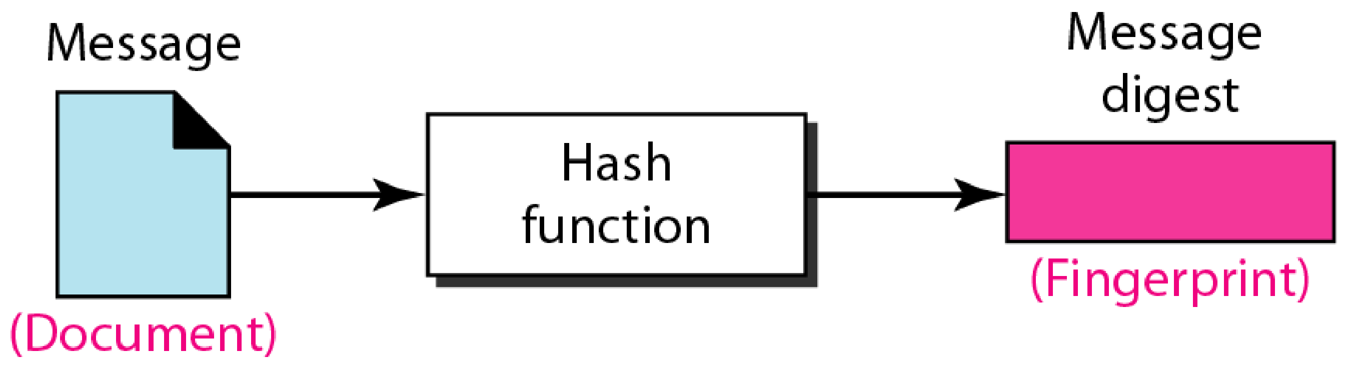
Topics discussed in this section:

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

Note

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

Figure 31.4 *Message and message digest*



Note

The message digest needs to be kept secret.

Figure 31.5 *Checking integrity*

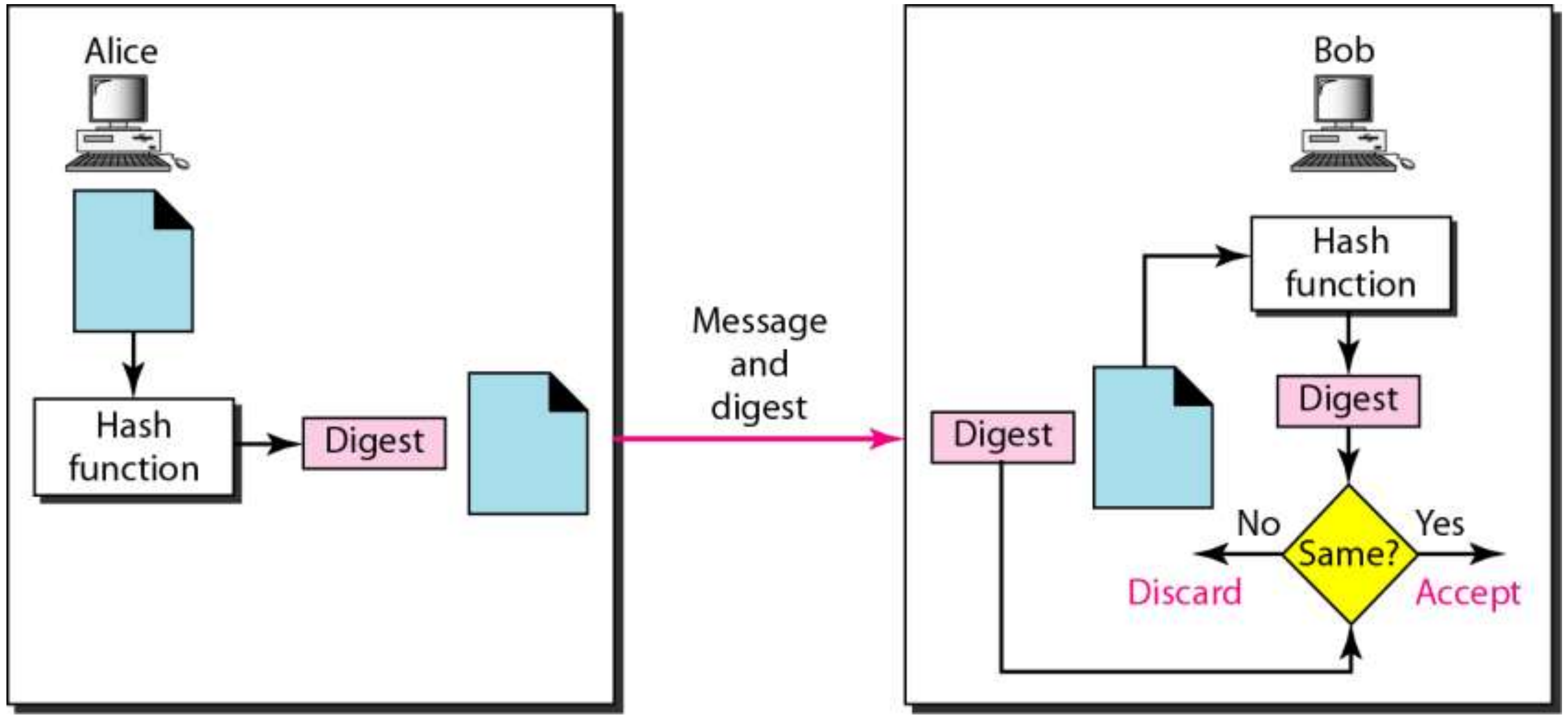
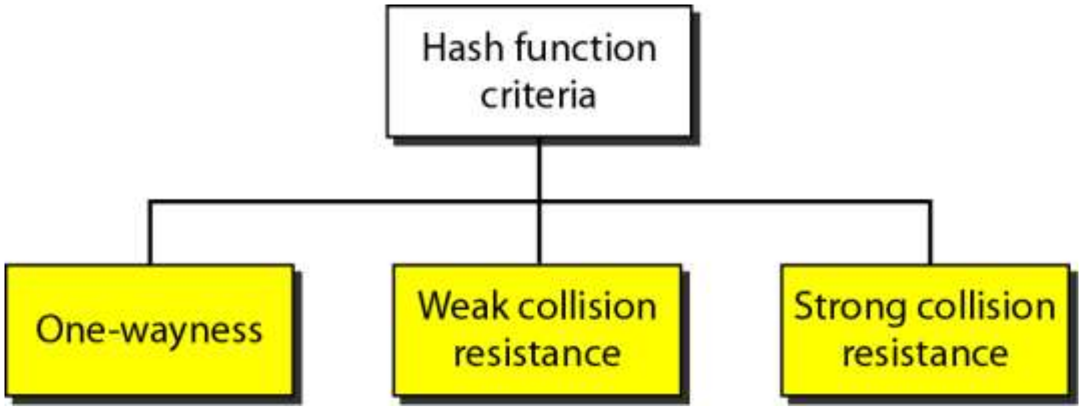


Figure 31.6 *Criteria of a hash function*



Example 31.1

Can we use a conventional lossless compression method as a hashing function?

Solution

We cannot. A lossless compression method creates a compressed message that is reversible. You can uncompress the compressed message to get the original one.

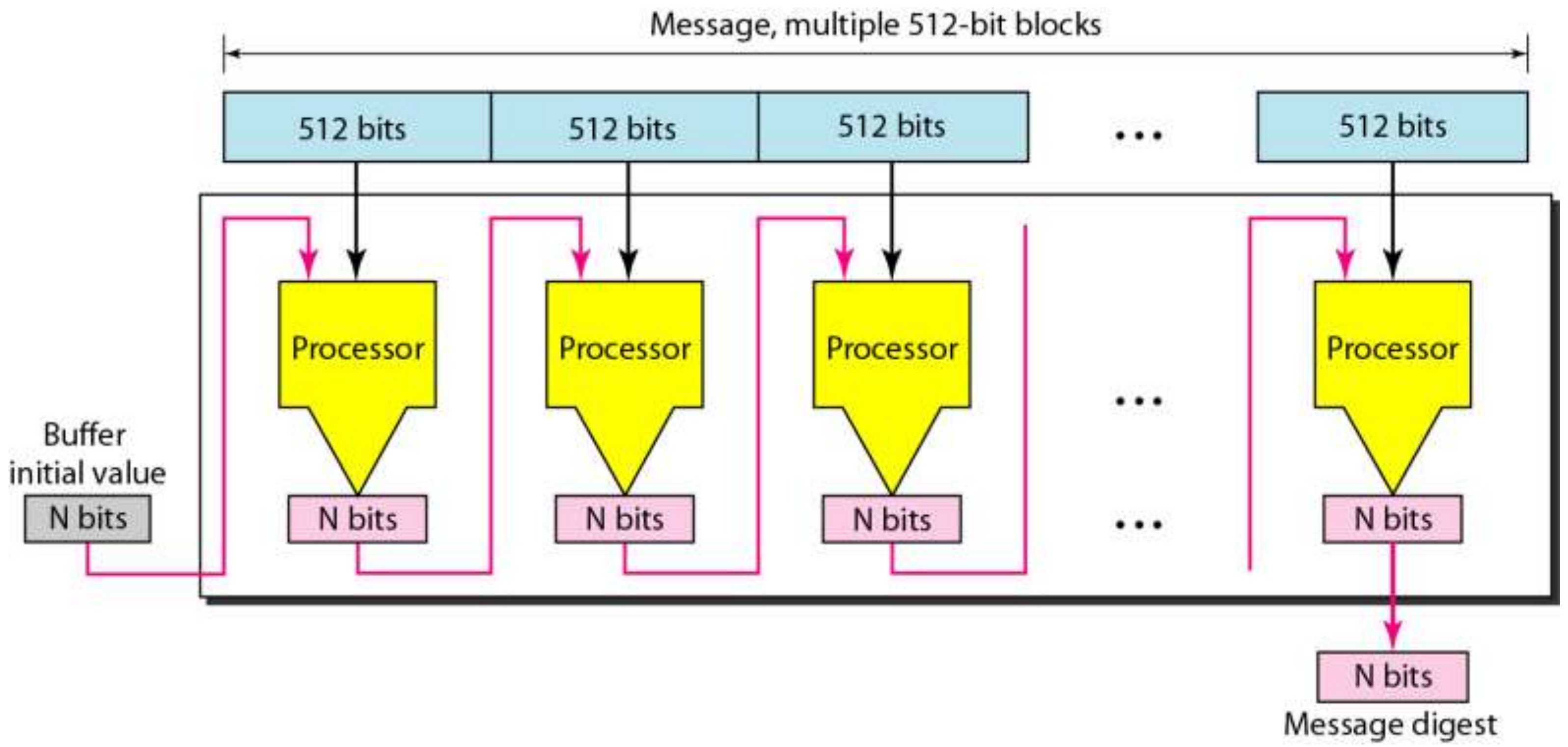
Example 31.2

Can we use a checksum method as a hashing function?

Solution

We can. A checksum function is not reversible; it meets the first criterion. However, it does not meet the other criteria.

Figure 31.7 *Message digest creation*

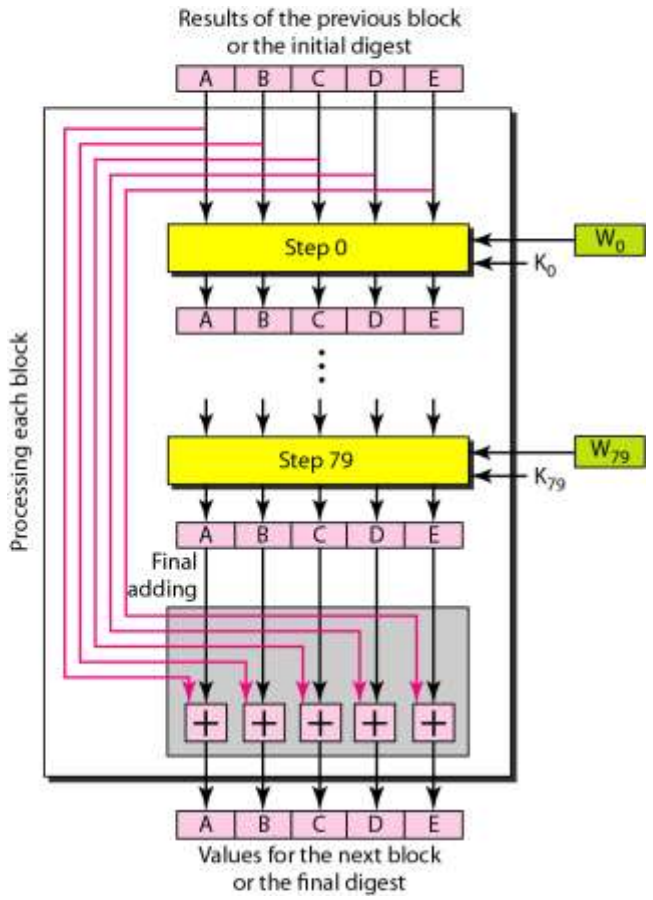


Note

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).

Figure 31.8 ***Processing of one block in SHA-1***



31-4 MESSAGE AUTHENTICATION

A hash function per se 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

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

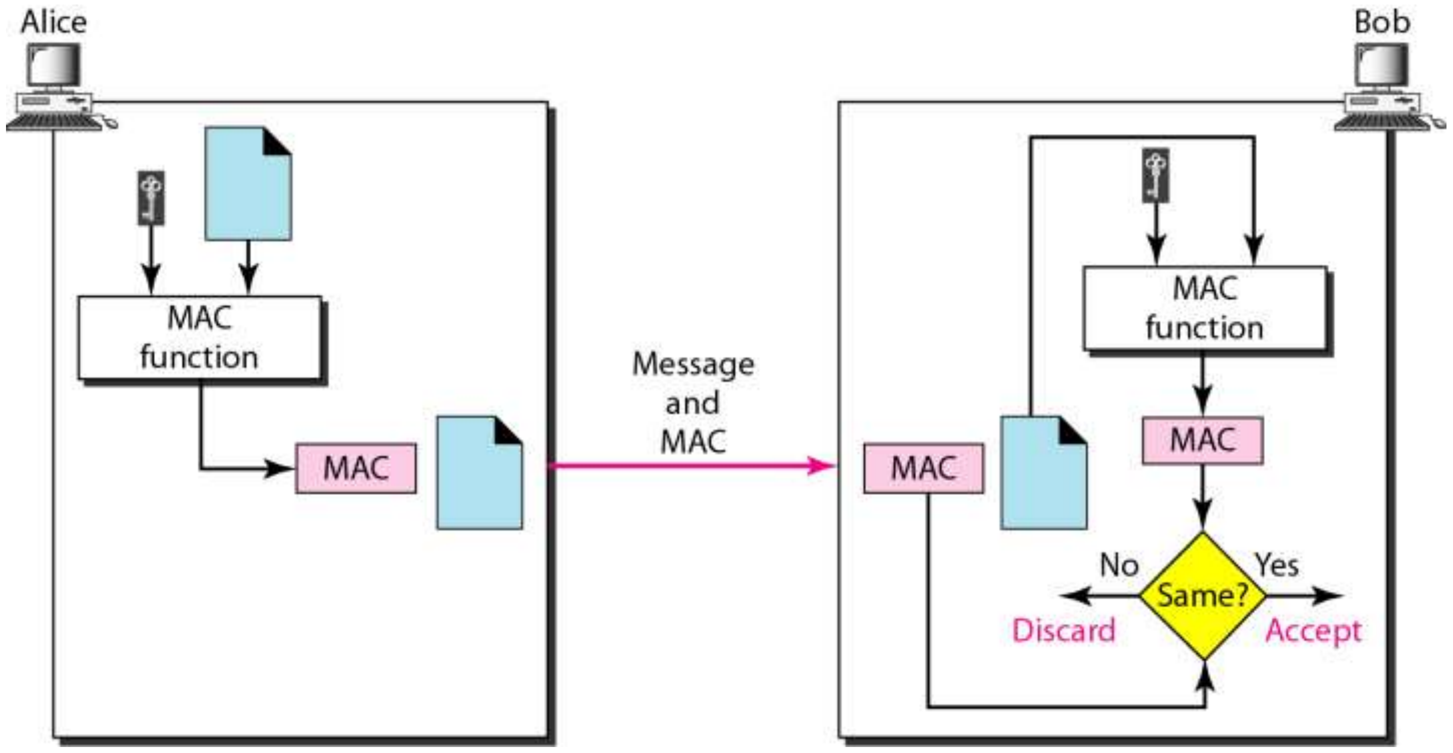
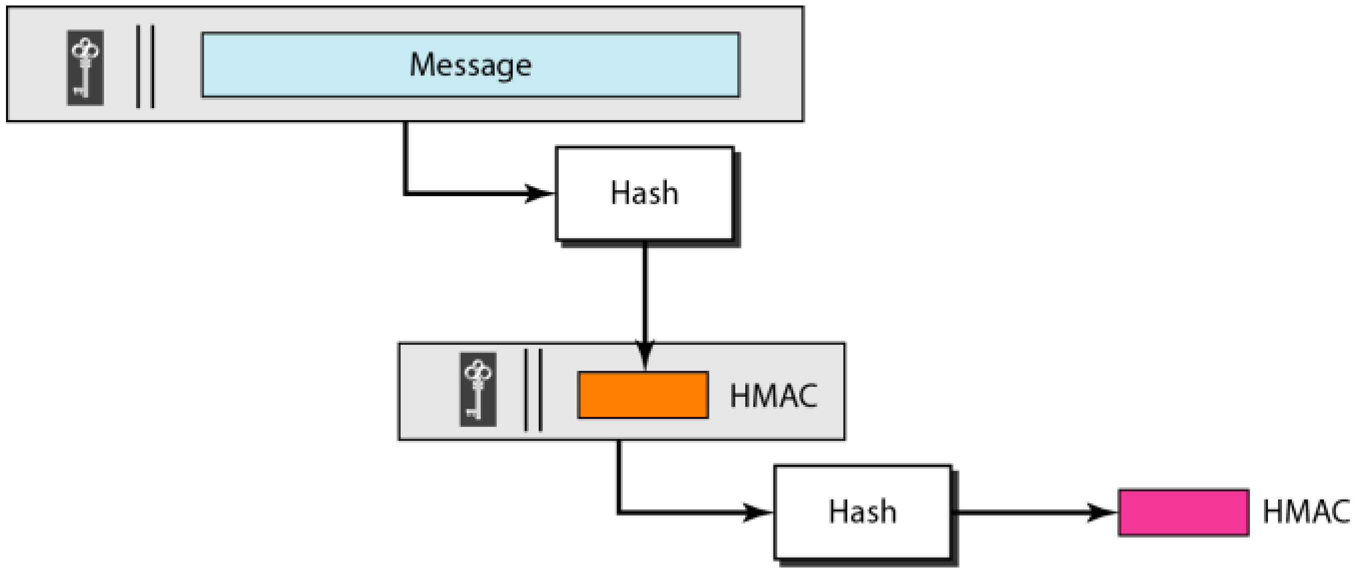


Figure 31.10 *HMAC*



31-5 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.

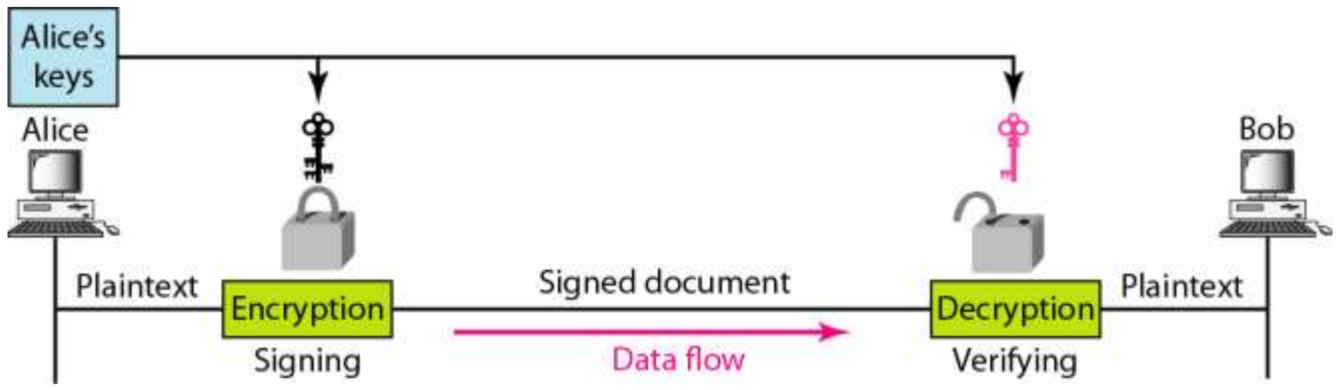
Topics discussed in this section:

Comparison
Need for Keys
Process

Note

A digital signature needs a public-key system.

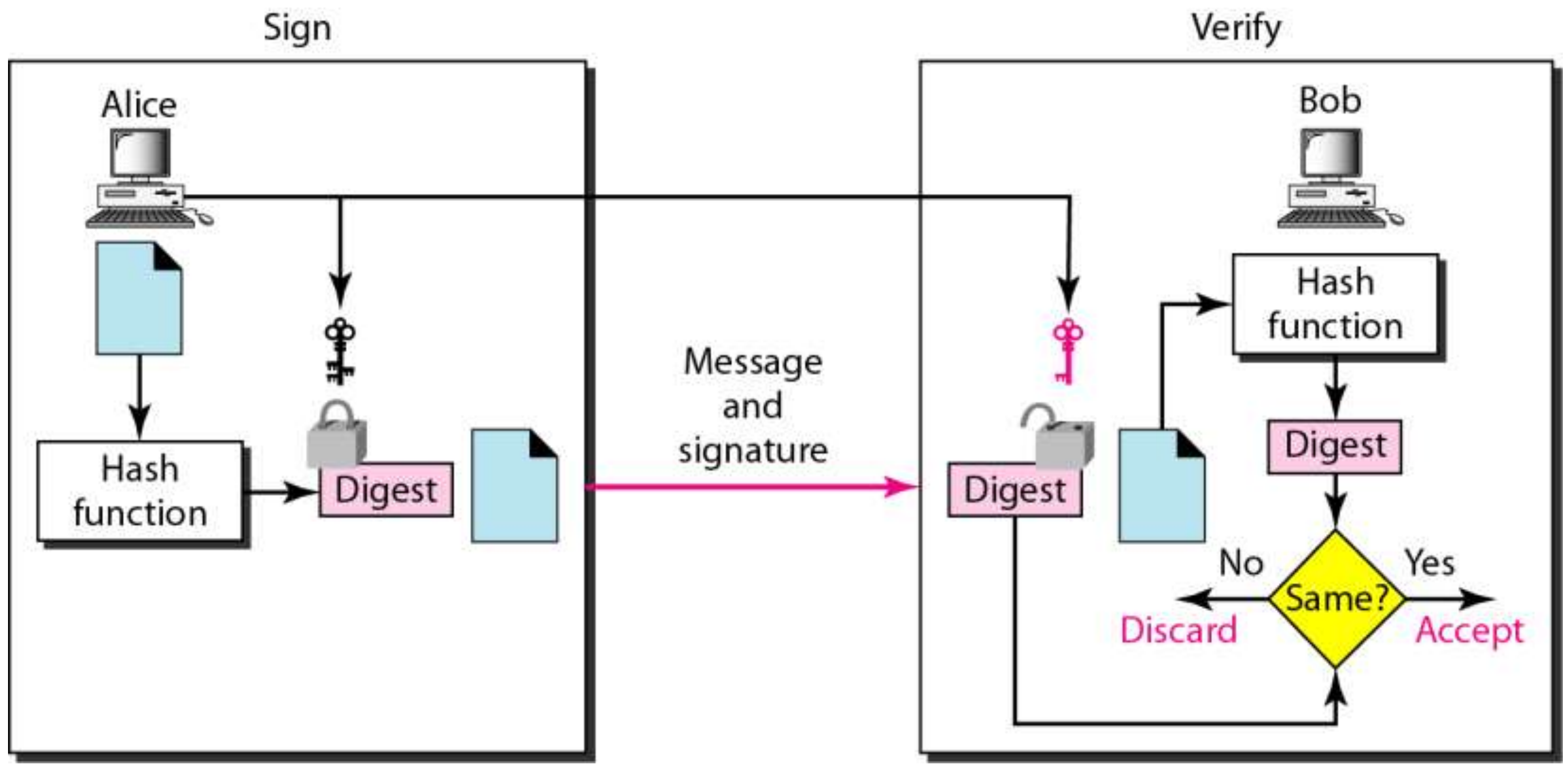
Figure 31.11 *Signing the message itself in digital signature*



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*



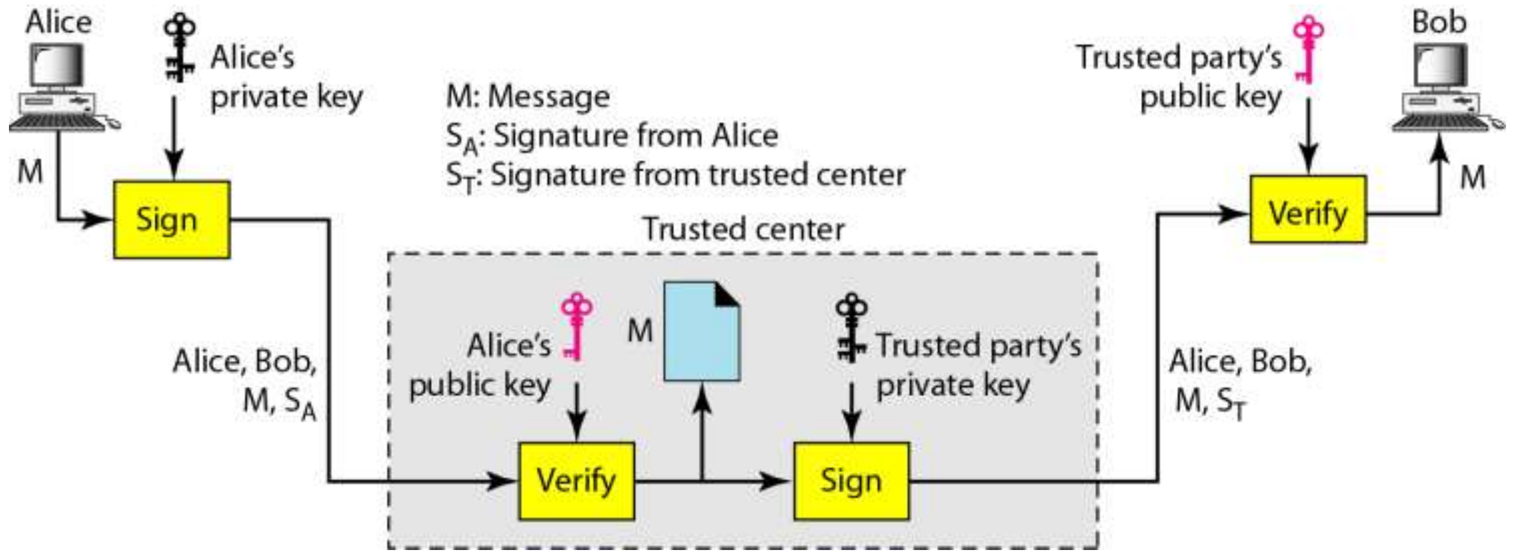
Note

A digital signature today provides message integrity.

Note

Digital signature provides message authentication.

Figure 31.13 *Using a trusted center for nonrepudiation*



Note

Nonrepudiation can be provided using a trusted party.

31-6 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

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

Note

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

Figure 31.14 *Challenge/response authentication using a nonce*

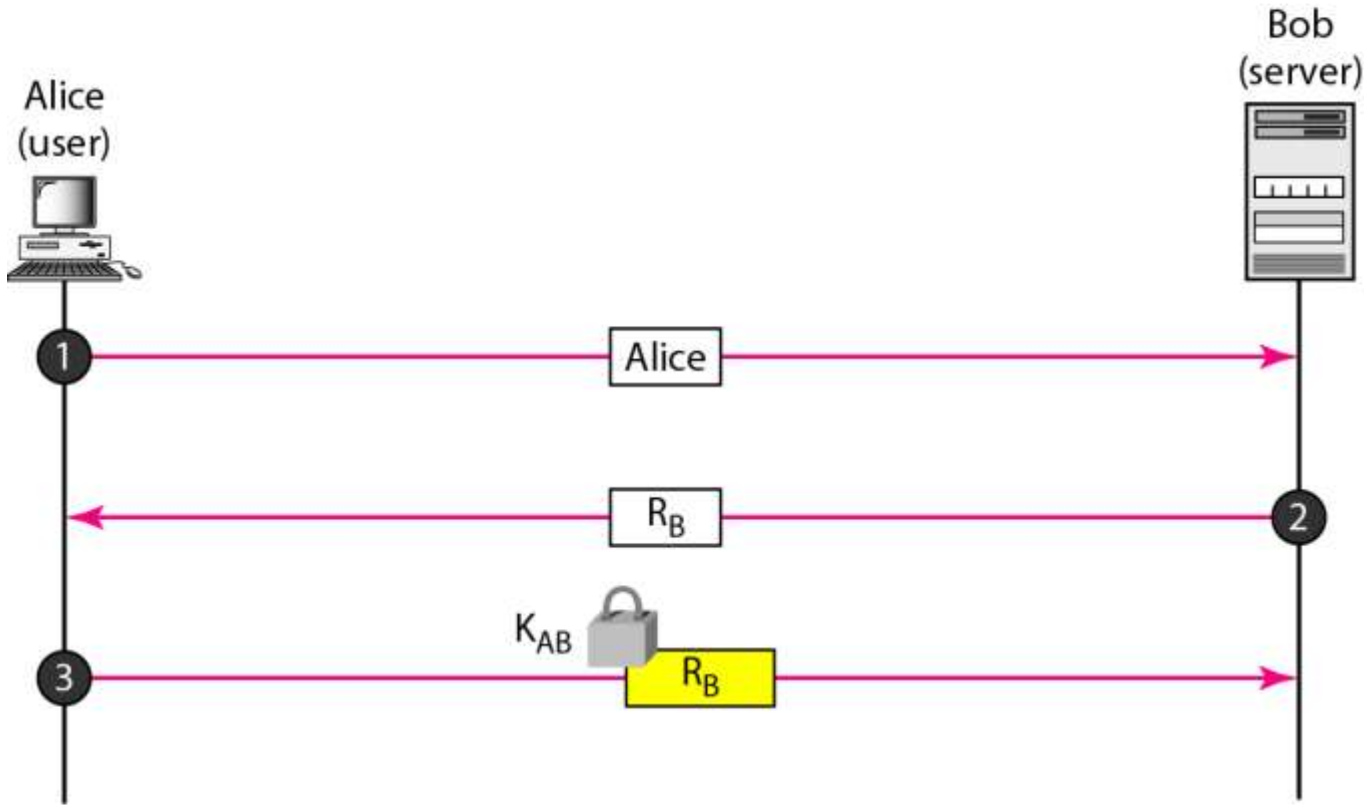


Figure 31.15 *Challenge-response authentication using a timestamp*

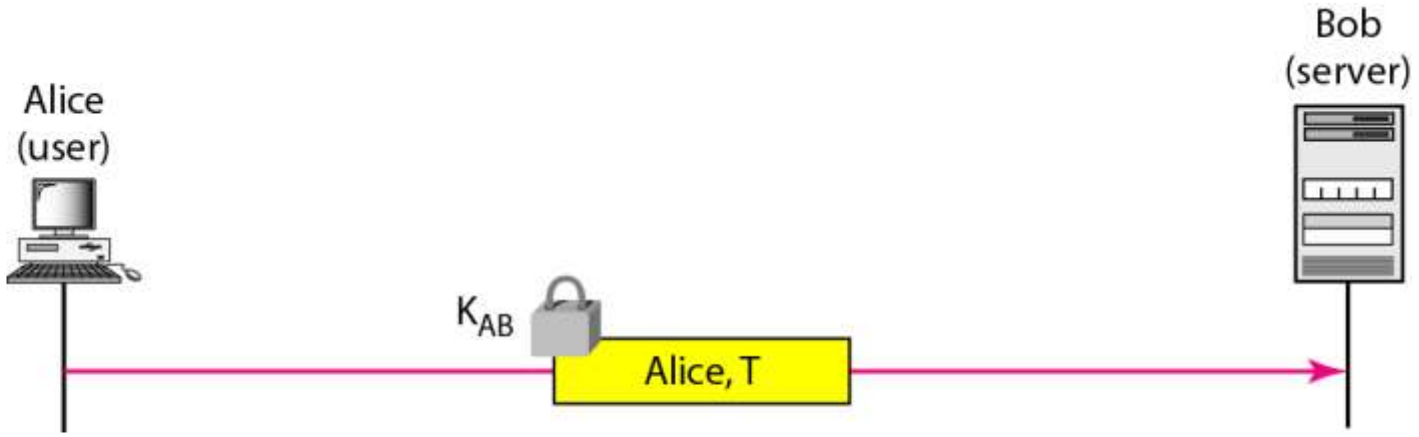


Figure 31.16 *Challenge-response authentication using a keyed-hash function*

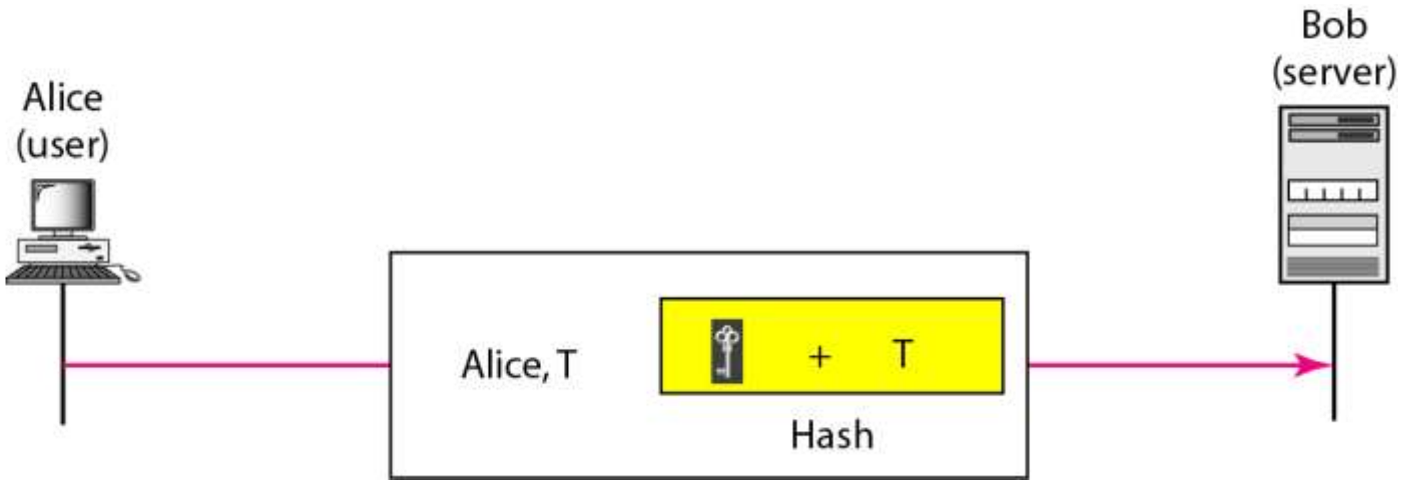


Figure 31.17 *Authentication, asymmetric-key*

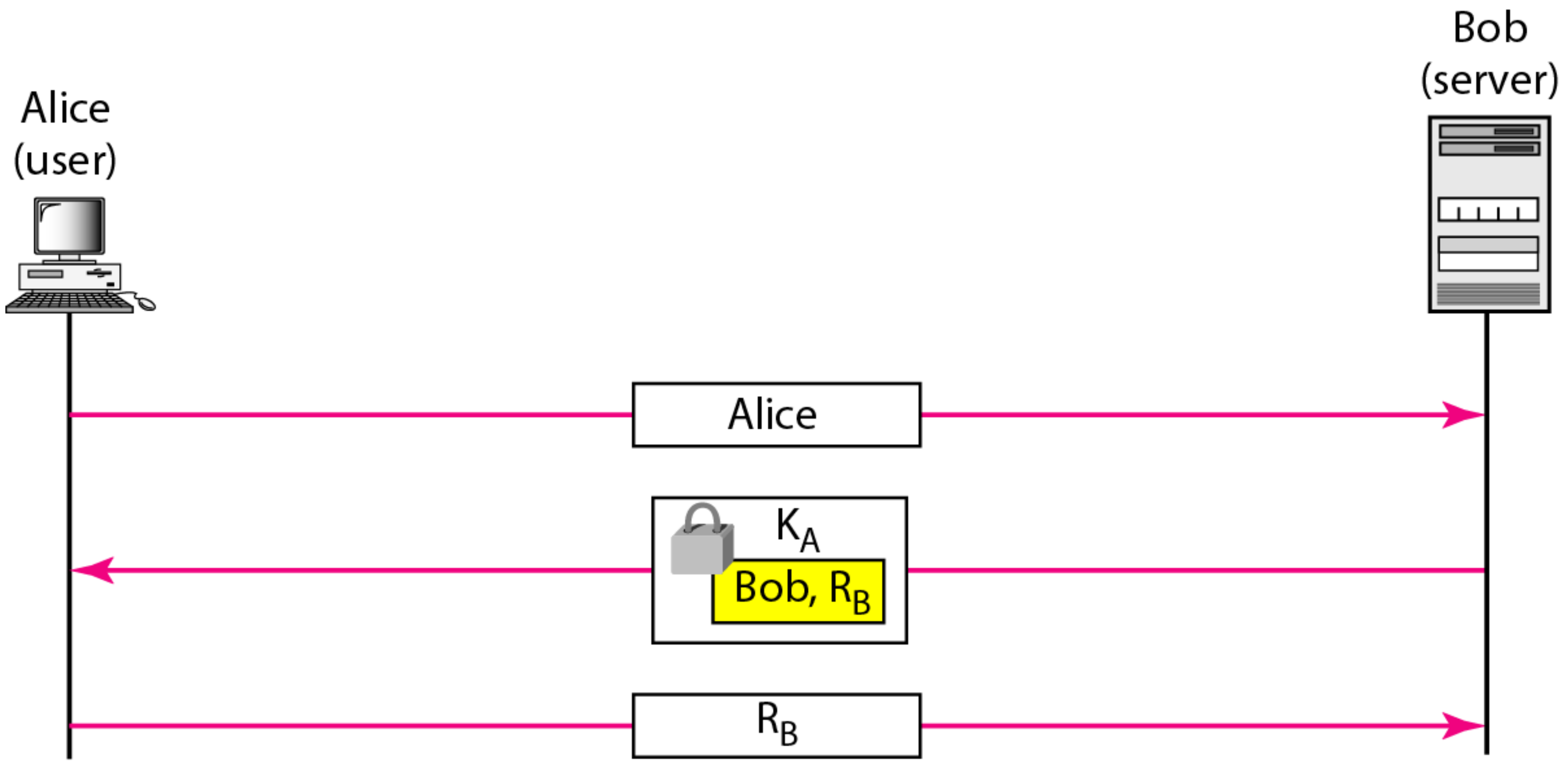
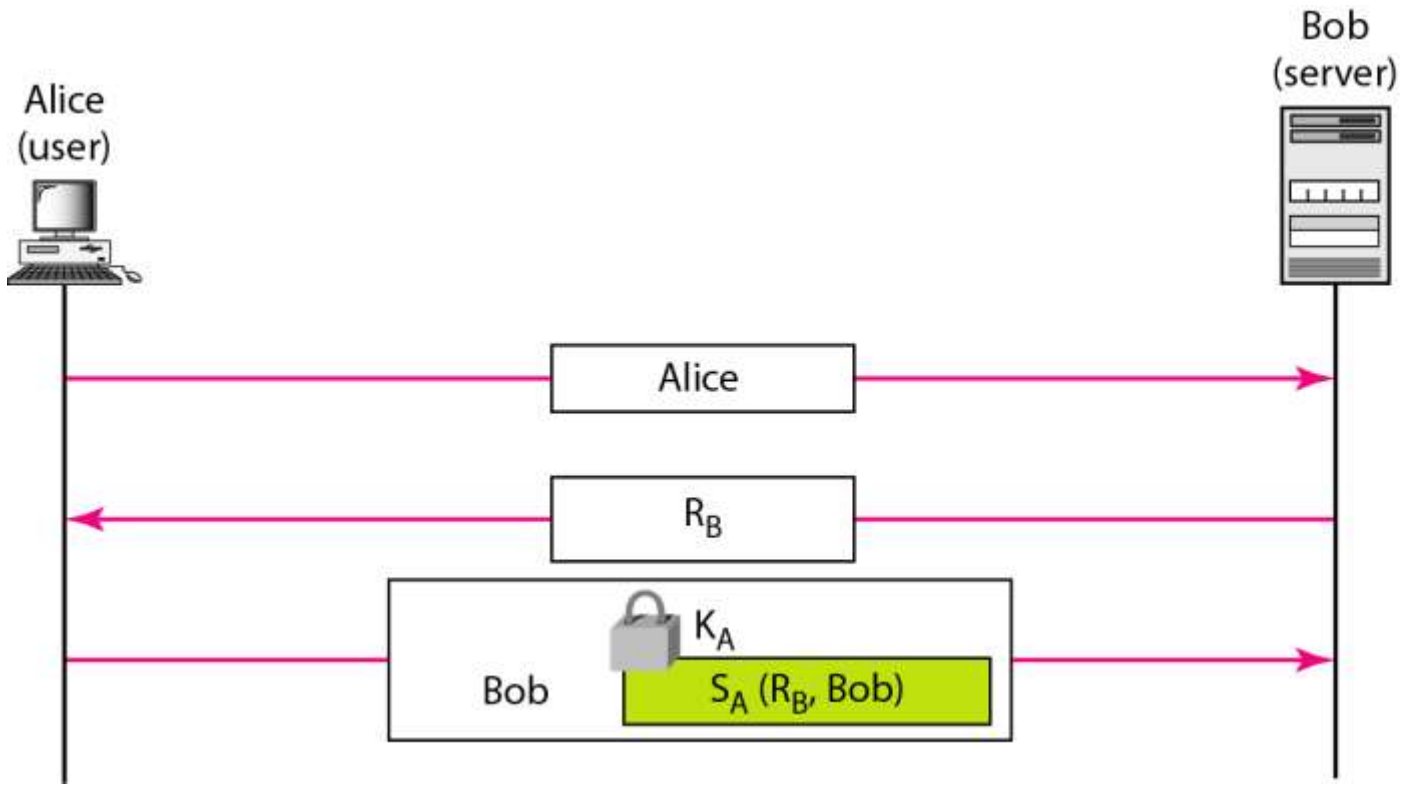


Figure 31.18 *Authentication, using digital signature*



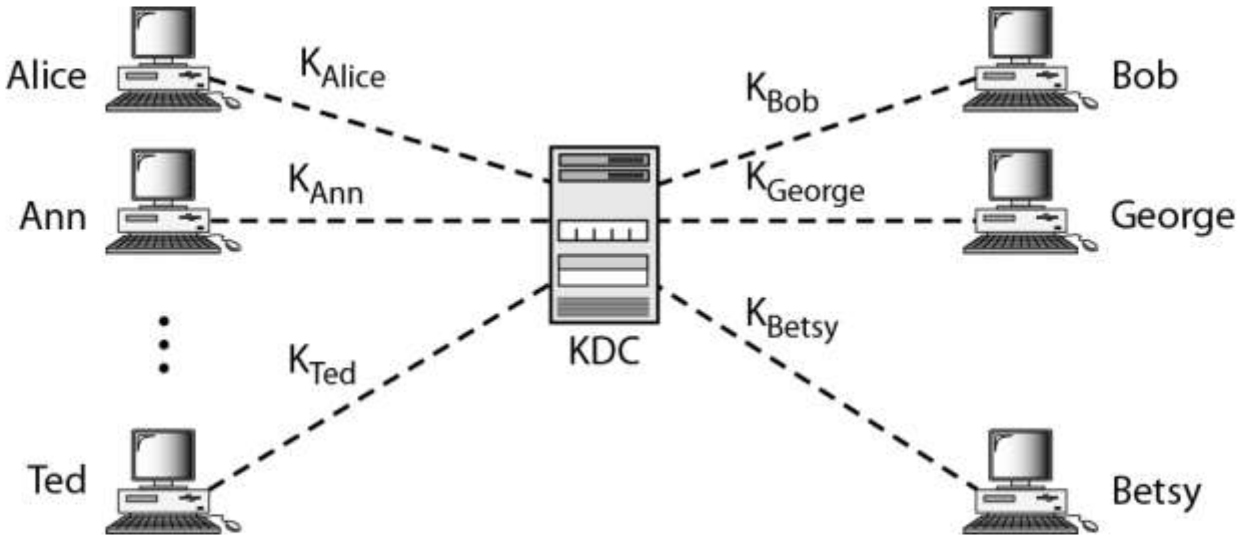
31-7 KEY MANAGEMENT

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 31.19 *KDC*



Note

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

Figure 31.30 *Creating a session key between Alice and Bob using KDC*

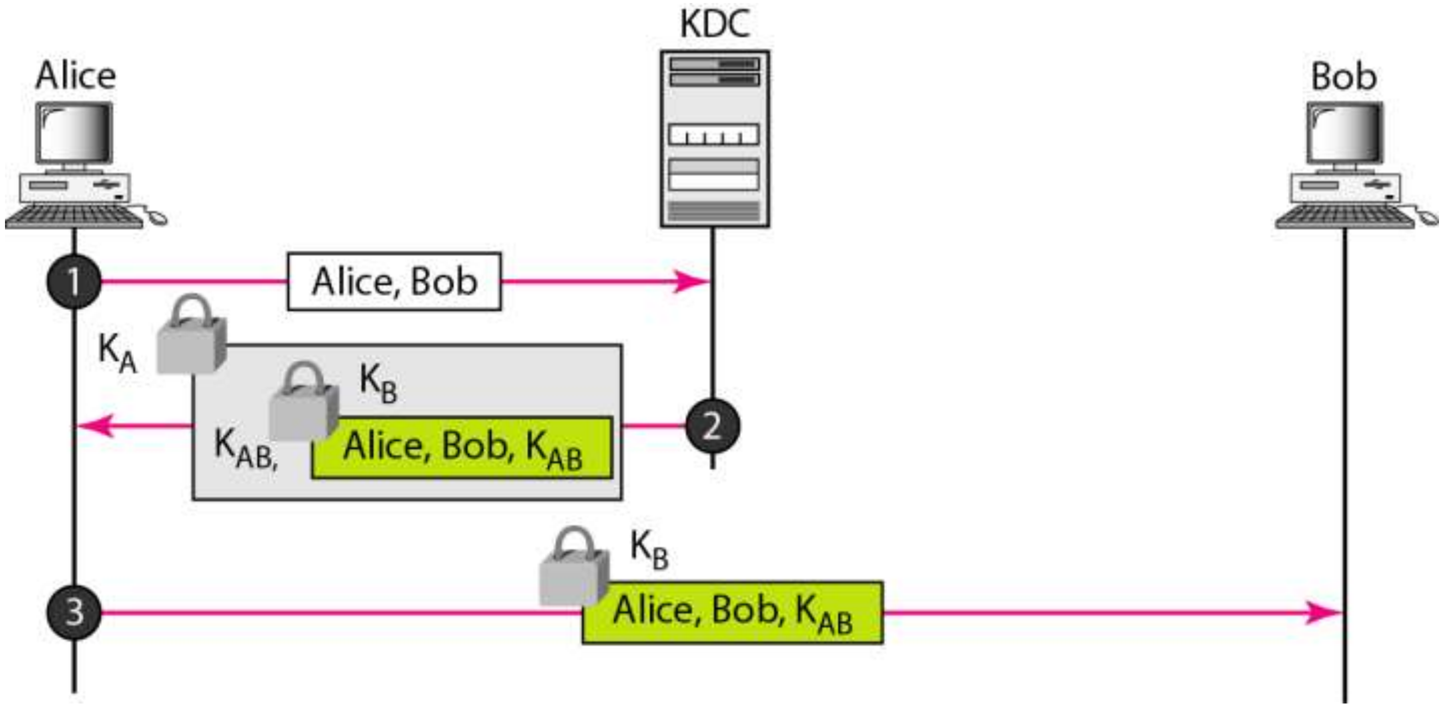


Figure 31.21 *Kerberos servers*

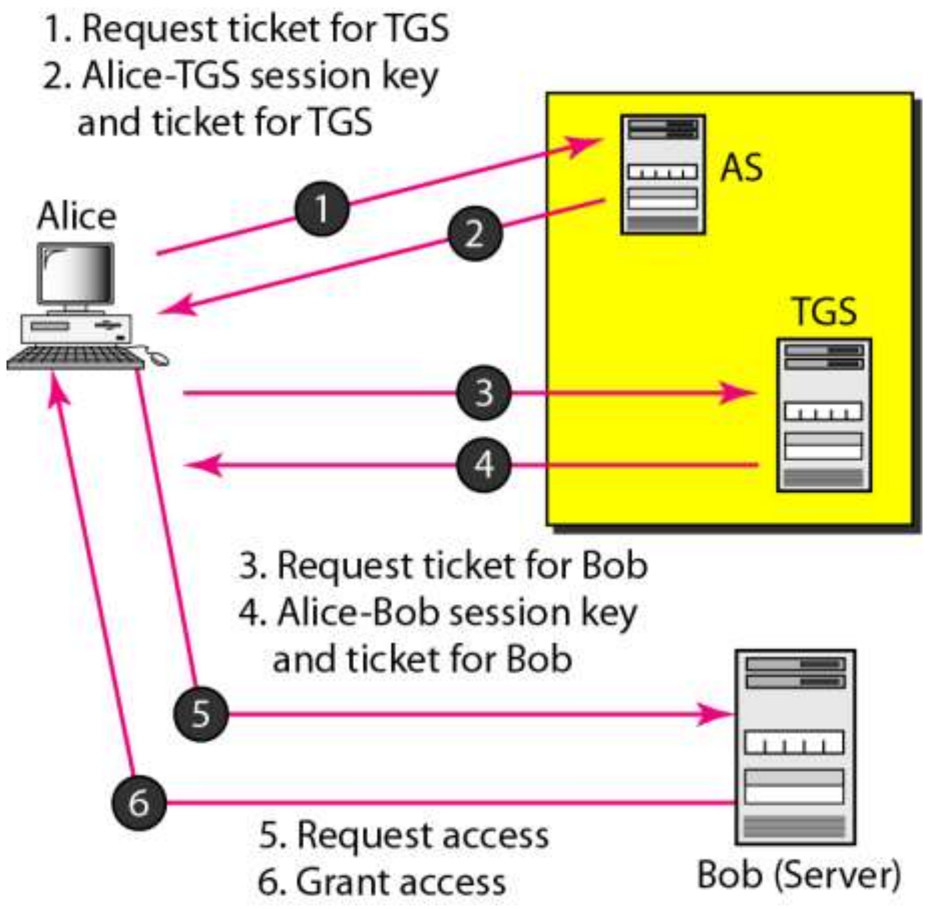
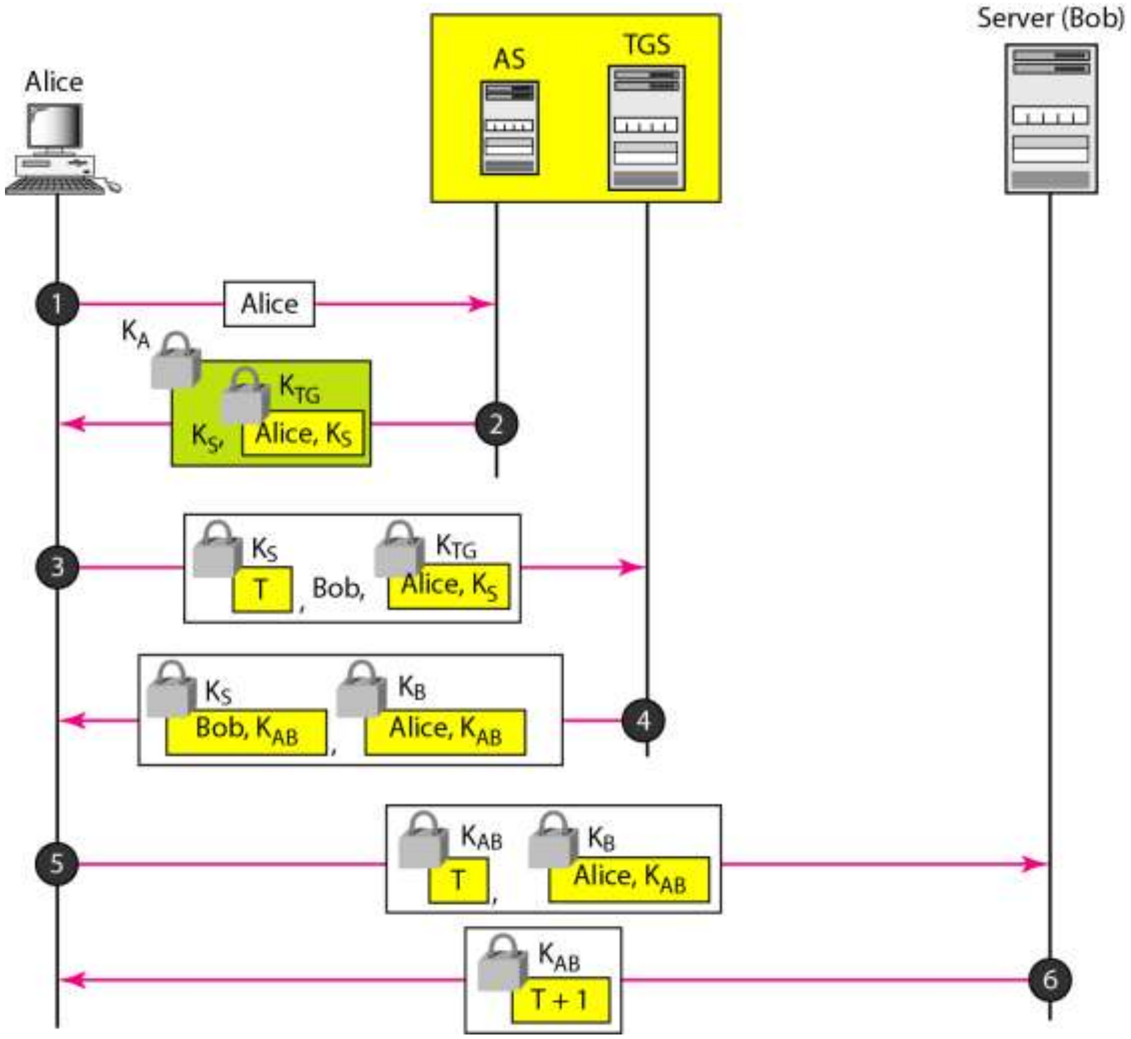

1. Request ticket for TGS
2. Alice-TGS session key and ticket for TGS
Alice
AS
TGS
1
2
3
4
5
6
3. Request ticket for Bob
4. Alice-Bob session key and ticket for Bob
5. Request access
6. Grant access
Bob (Server)

Figure 31.22 *Kerberos example*



Note

In public-key cryptography, everyone has access to everyone's public key; public keys are available to the public.

Figure 31.23 *Announcing a public key*

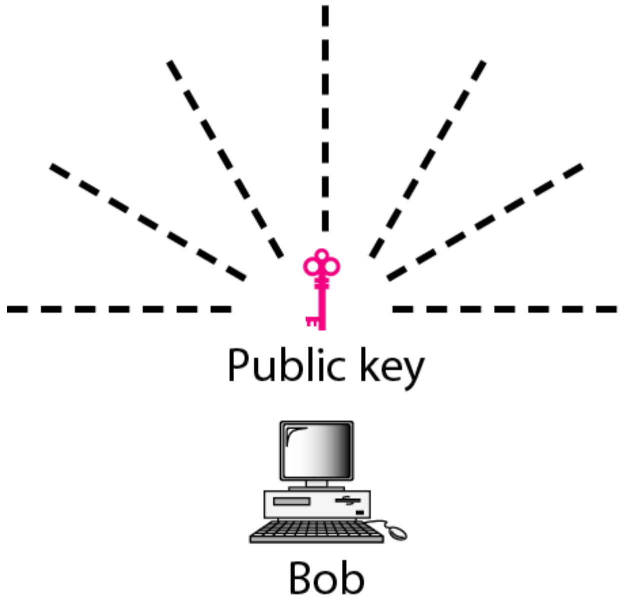


Figure 31.24 *Trusted center*

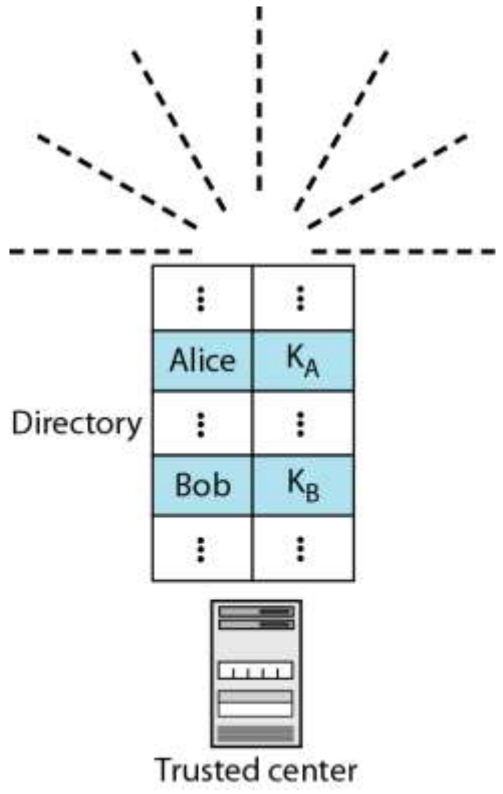


Figure 31.25 *Controlled trusted center*

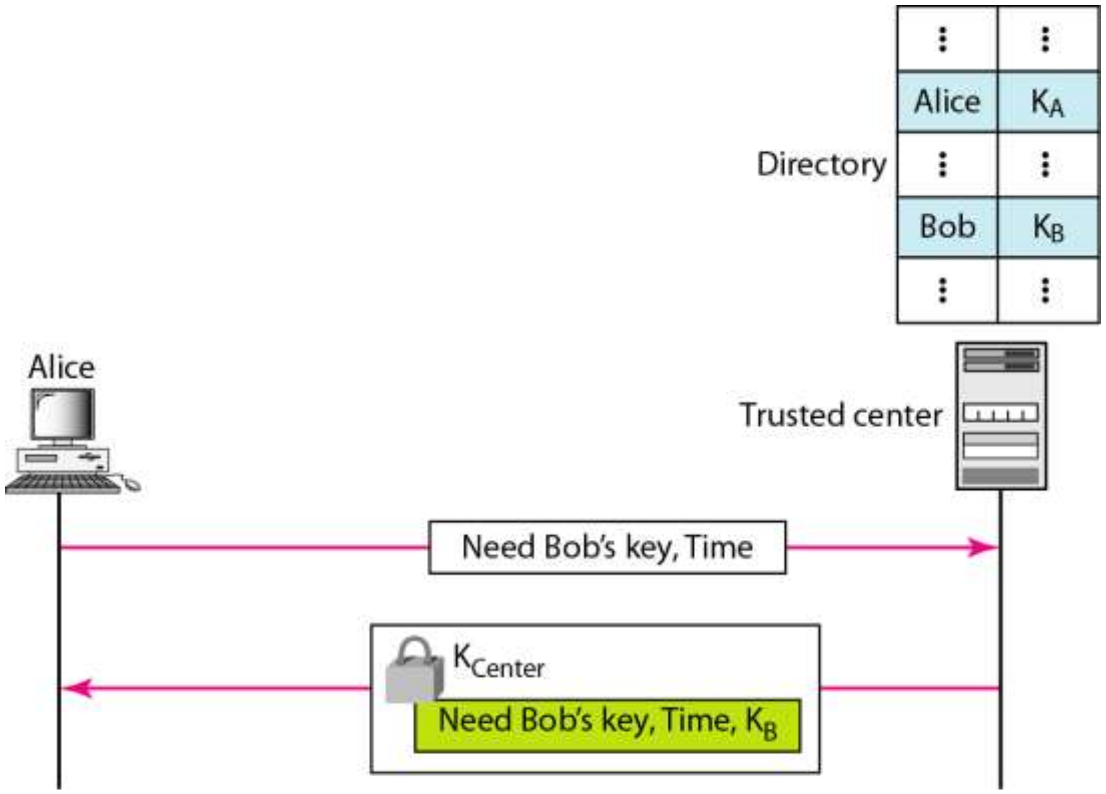


Figure 31.26 *Certification authority*

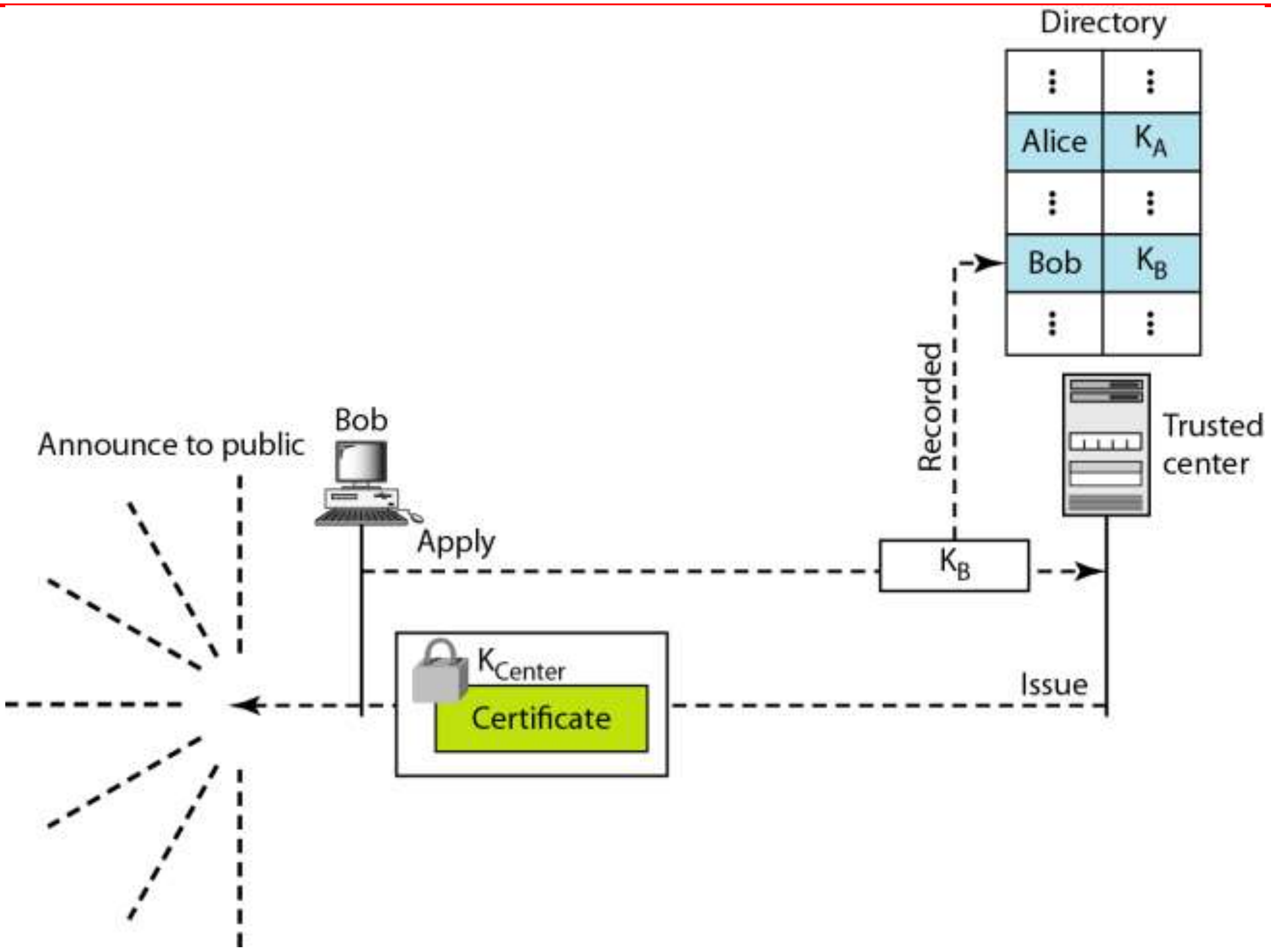
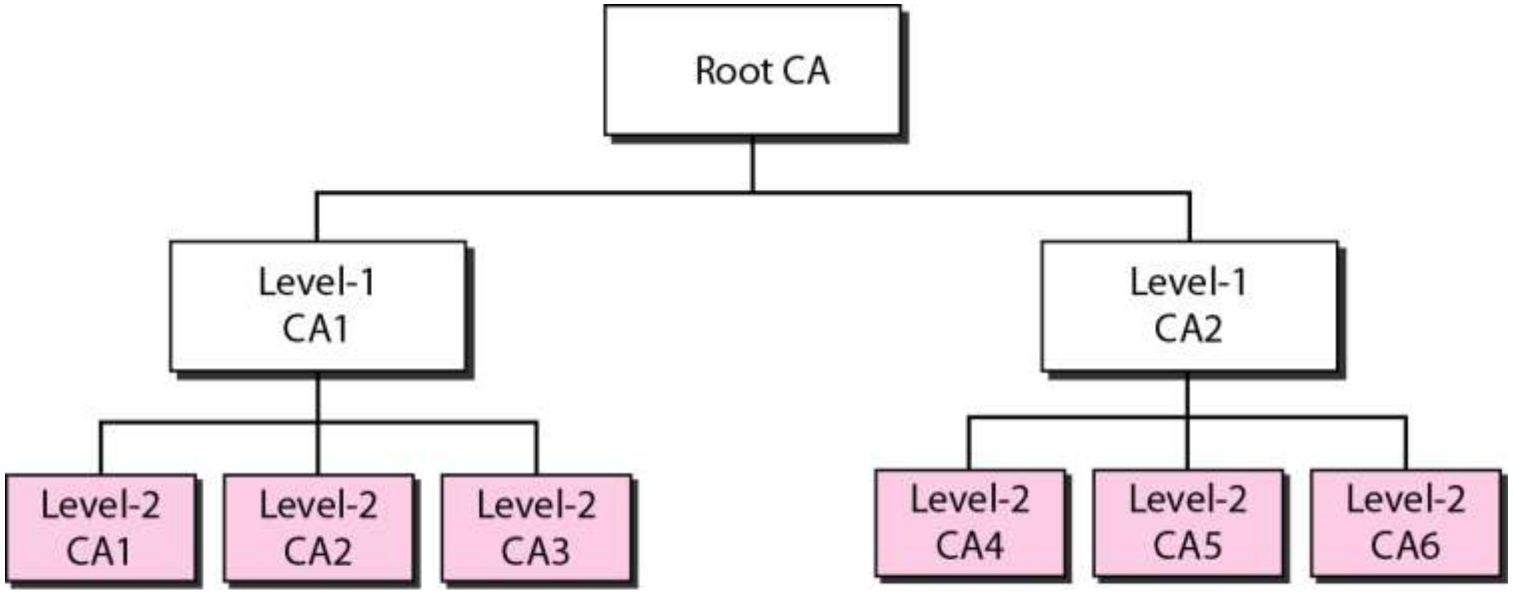


Figure 31.27 *PKI hierarchy*


Root CA
Level-1 CA1
Level-1 CA2
Level-2 CA1
Level-2 CA2
Level-2 CA3
Level-2 CA4
Level-2 CA5
Level-2 CA6