UNIT - IV

Cryptographic Hash Function

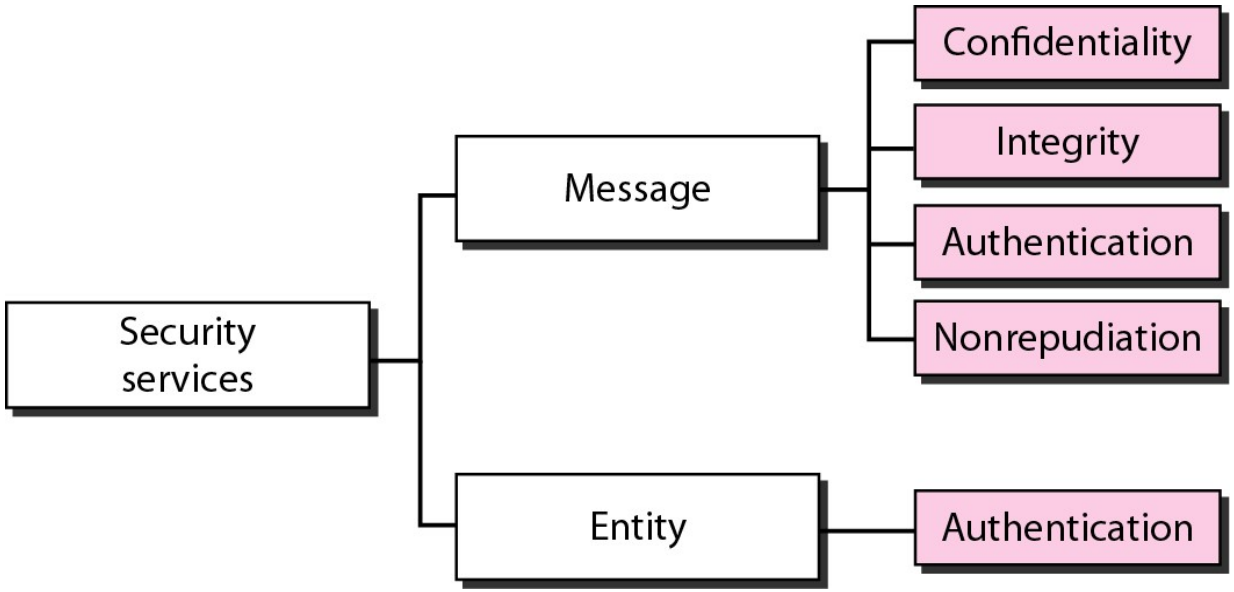
31-1 SECURITY SERVICES

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*



31-3 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.

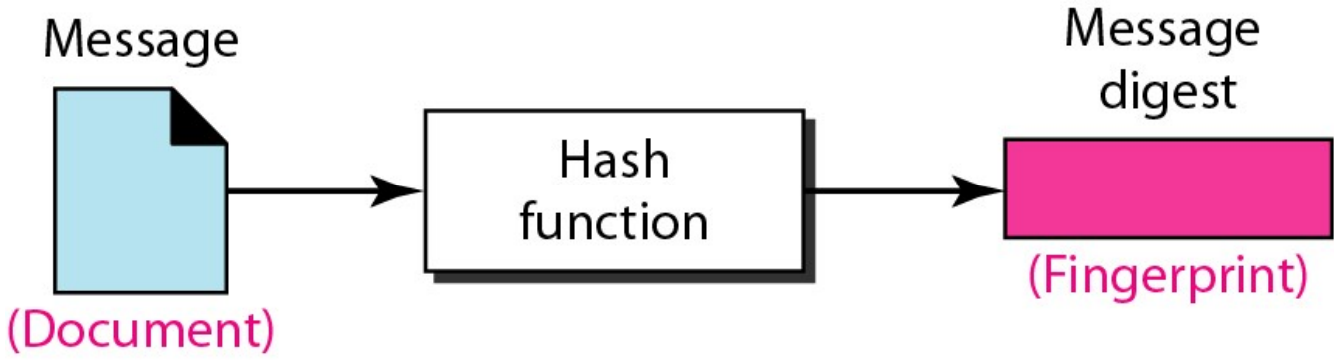
Topics discussed in this section:

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*



Notations:

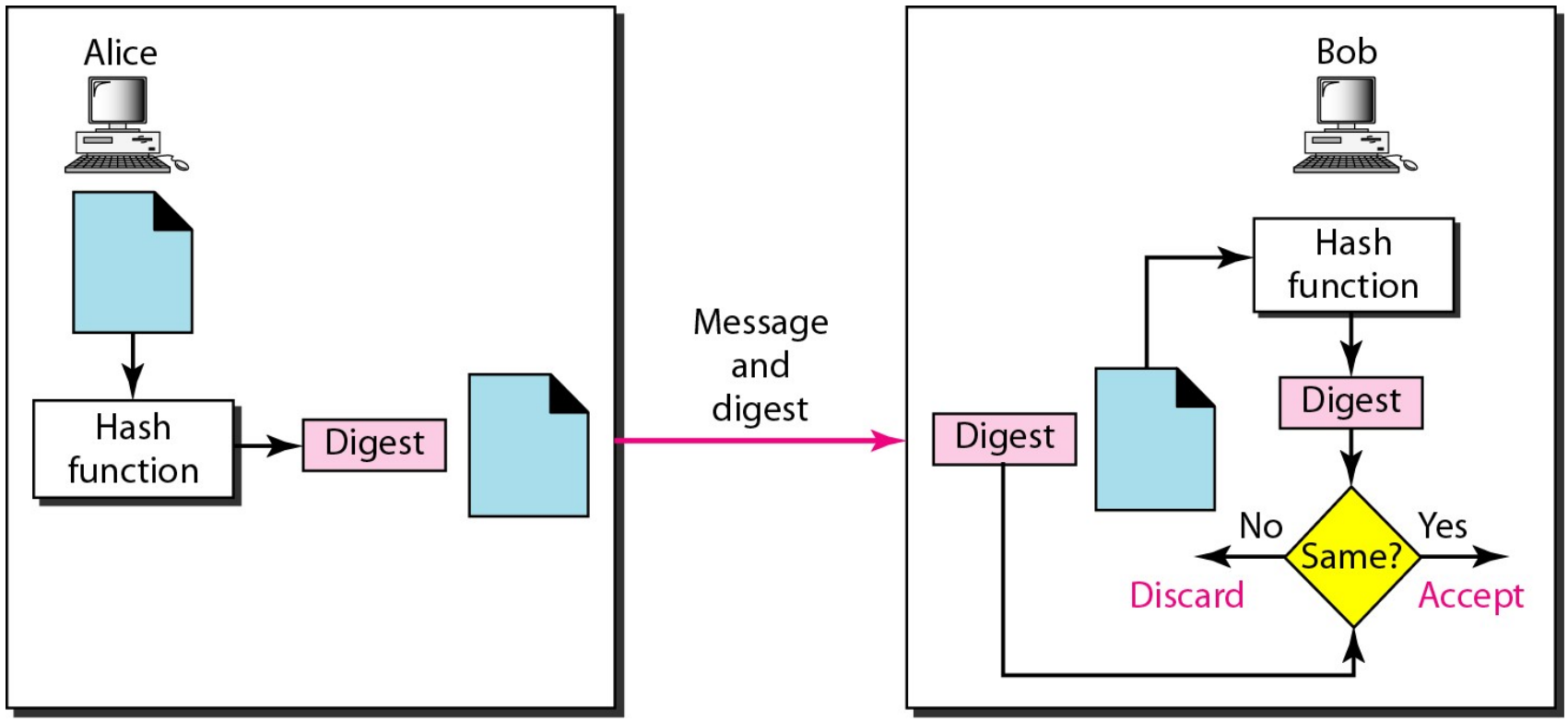
m: message

H(m): message digest of m by using hash function H()

Note

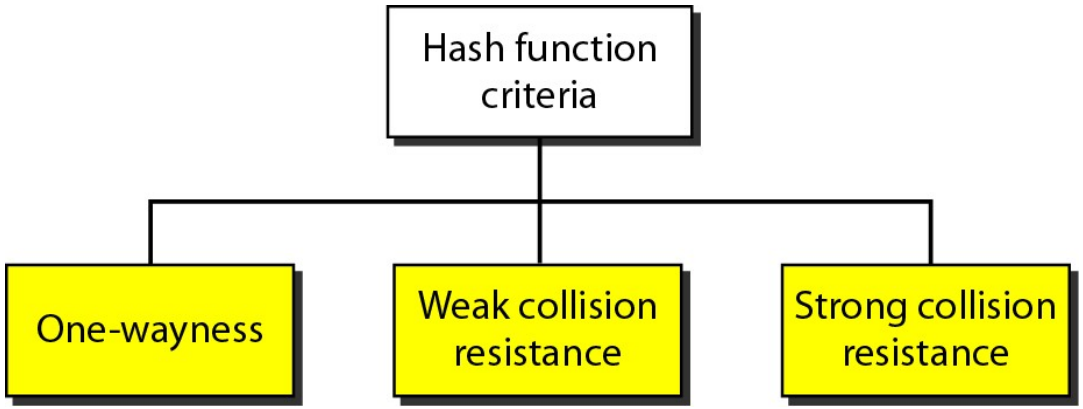
The message digest needs to be kept secret, or unalterable by others.

Figure 31.5 *Checking integrity*



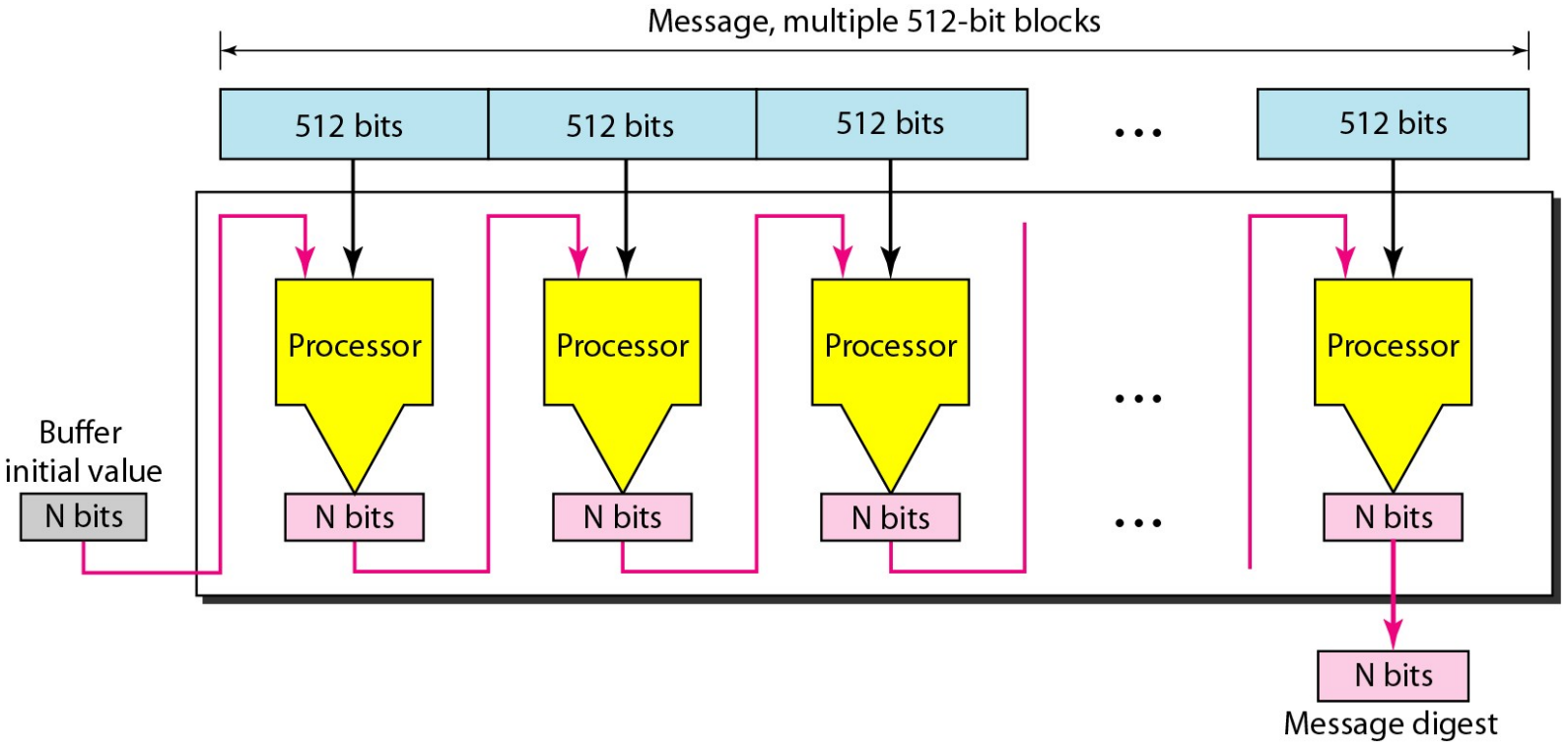
Notes: We need to make sure the digest cannot be altered by attacker

Figure 31.6 *Criteria of a hash function*



- One-wayness:
  - *Cannot recover message m given its digest H(m)*
- Weak collision resistance:
  - *Given message m, cannot generate another message m' such that H(m')=H(m)* → ensure integrity
- Strong collision resistance: *(a stronger requirement than above one)*
  - *Sender cannot generate two messages m and m' such that H(m)=H(m')* → ensure nonrepudation

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.

Another popular hash algorithm is MD5 (message digest algorithm 5). It is an older generation than SHA-1.

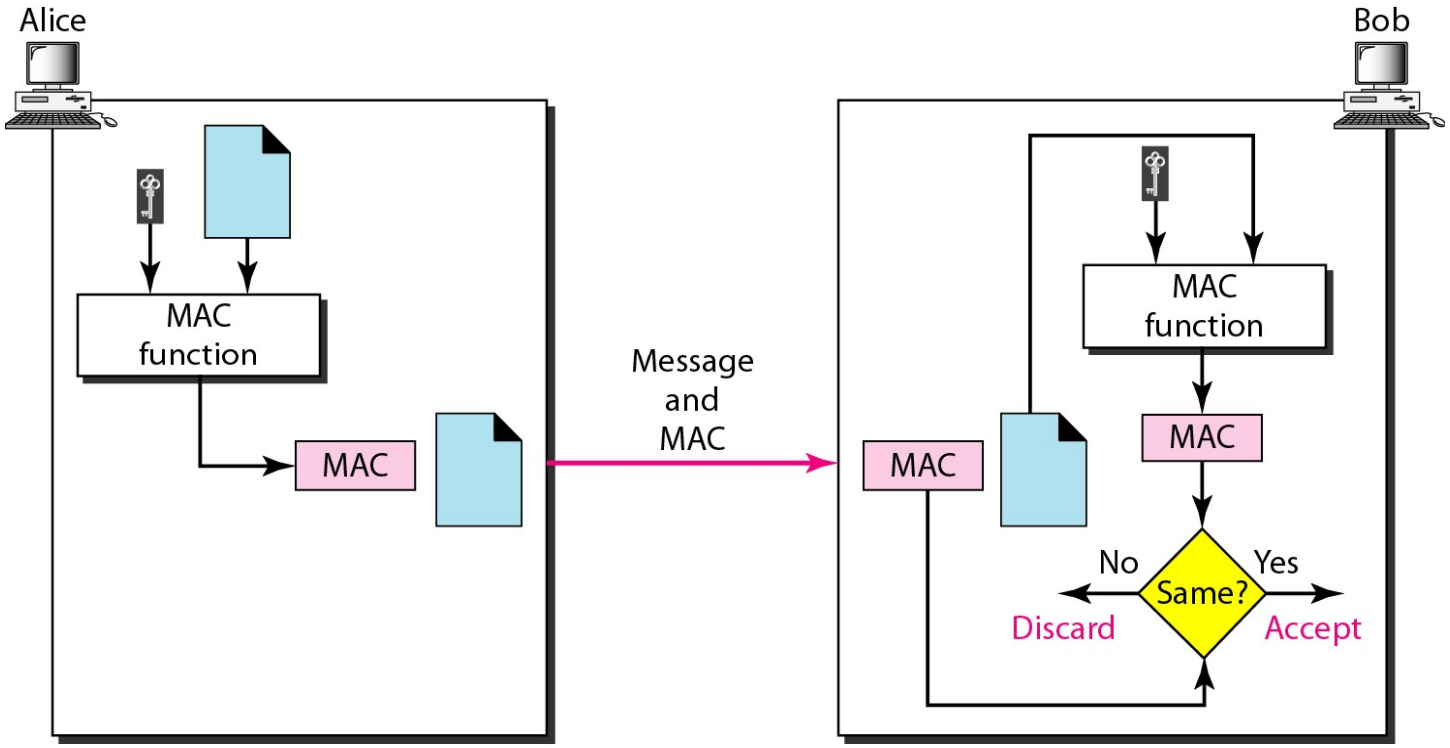
31-4 MESSAGE AUTHENTICATION

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:

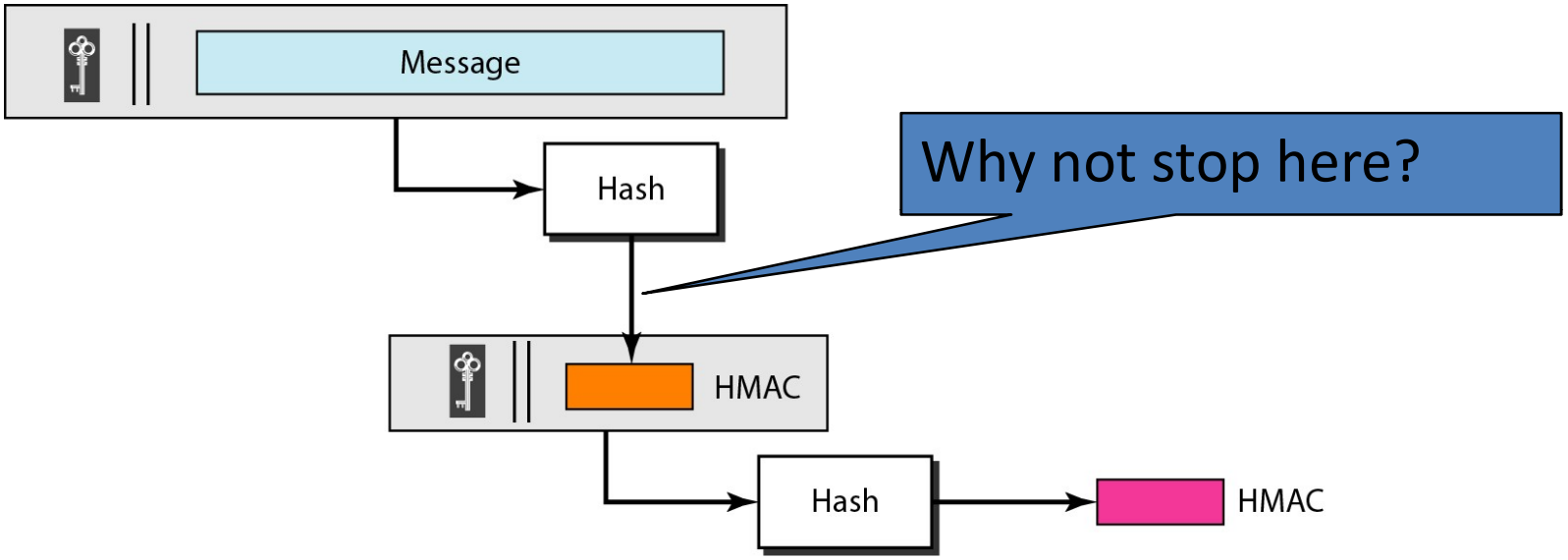
MAC (message authentication code): can be used to ensure both integrity and authentication

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



Message itself is unencrypted

Figure 31.10 *HMAC (Hashed MAC): uses keyless hash function*



Reason: we can directly use mature keyless hash function such as SHA-1 or MD5

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.

Notations:

m: message

H(m): message digest of m by using hash function H()

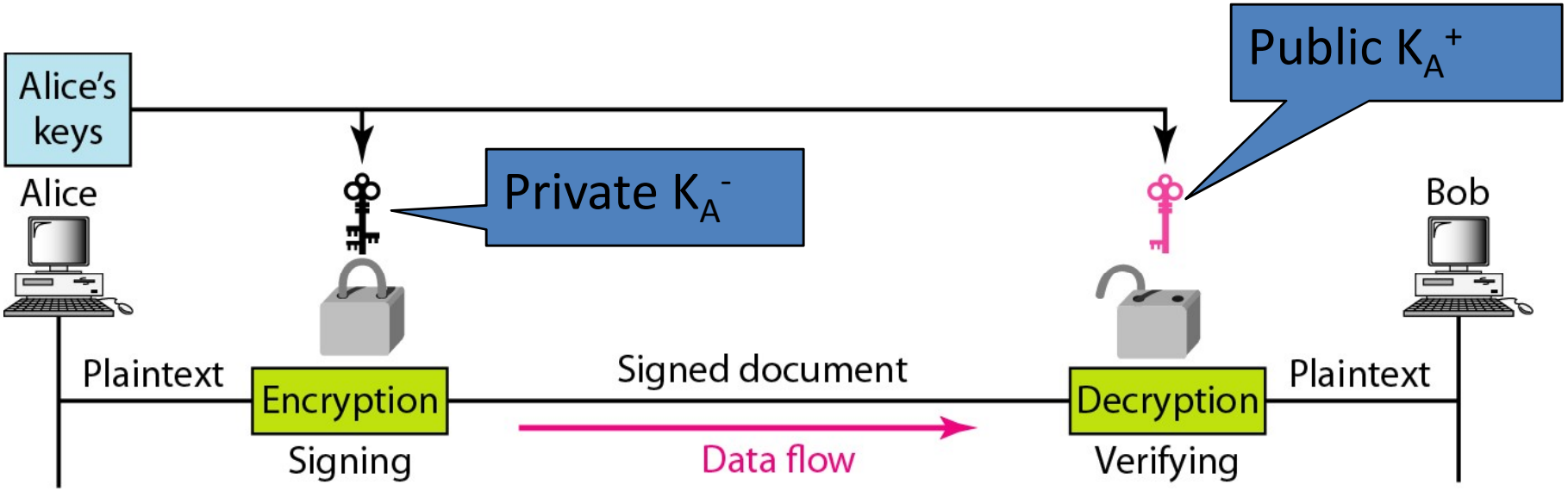
K_A^- *:Private key of user A*

K_A^+ *:Public key of user A*

K_{AB}*: Symmetric key between A and B*

K(m): ciphertext of message m by using encryption key K

Figure 31.11 *Signing the message itself in digital signature*



Provide no confidentiality (message is not secret)

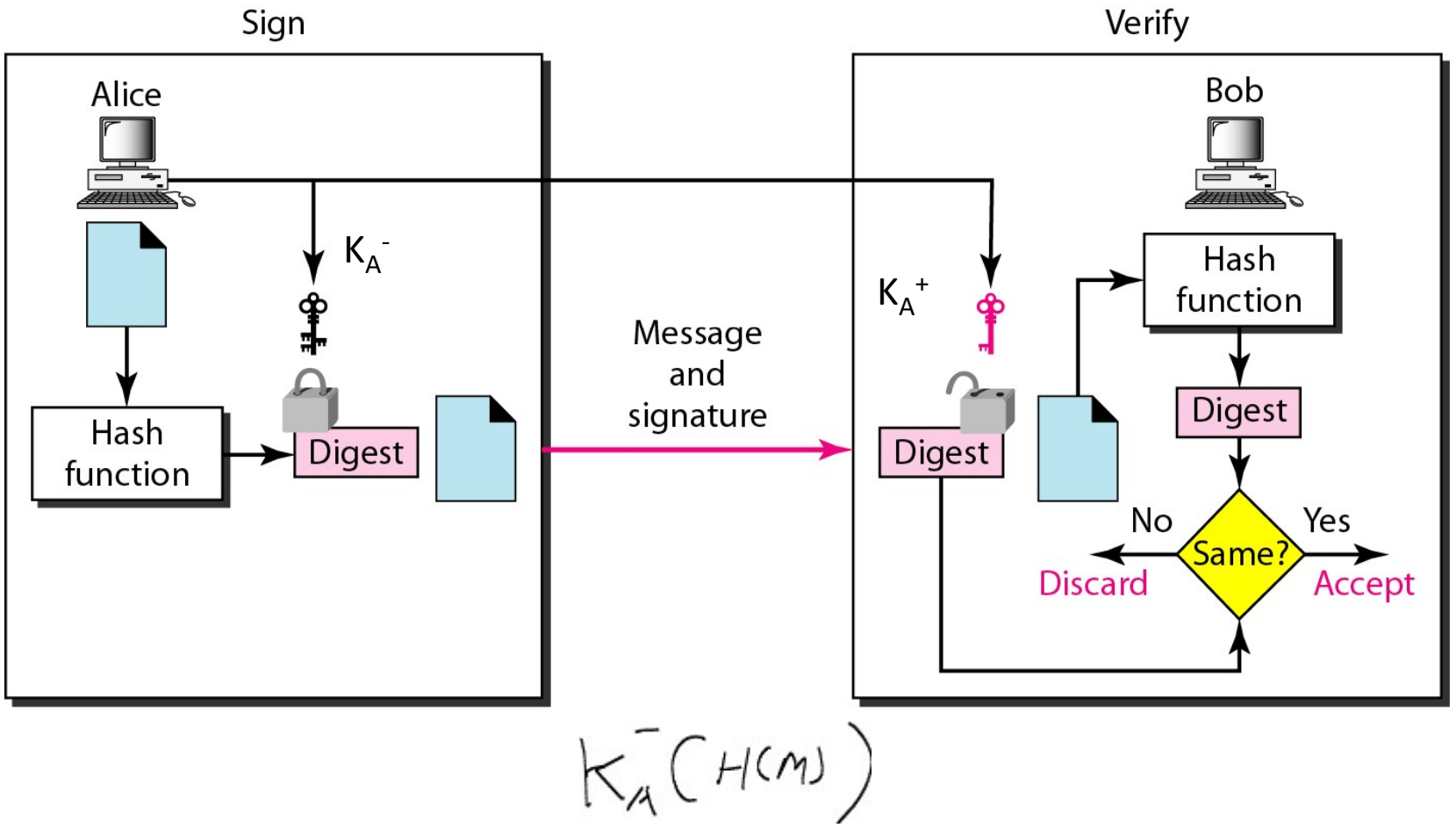
Problem: Too expensive to sign message itself using public key system

Note

For message confidentiality, we use the private and public keys of the receiver;

In digital signature (integrity, authentication, nonrepudation), we use the private and public keys of the sender.

Figure 31.12 *Signing the digest in a digital signature*



Digital signature provides three out of the five services we mentioned for security systems

- ✓Integrity
- ✓Authentication
- ✓Nonrepudiation

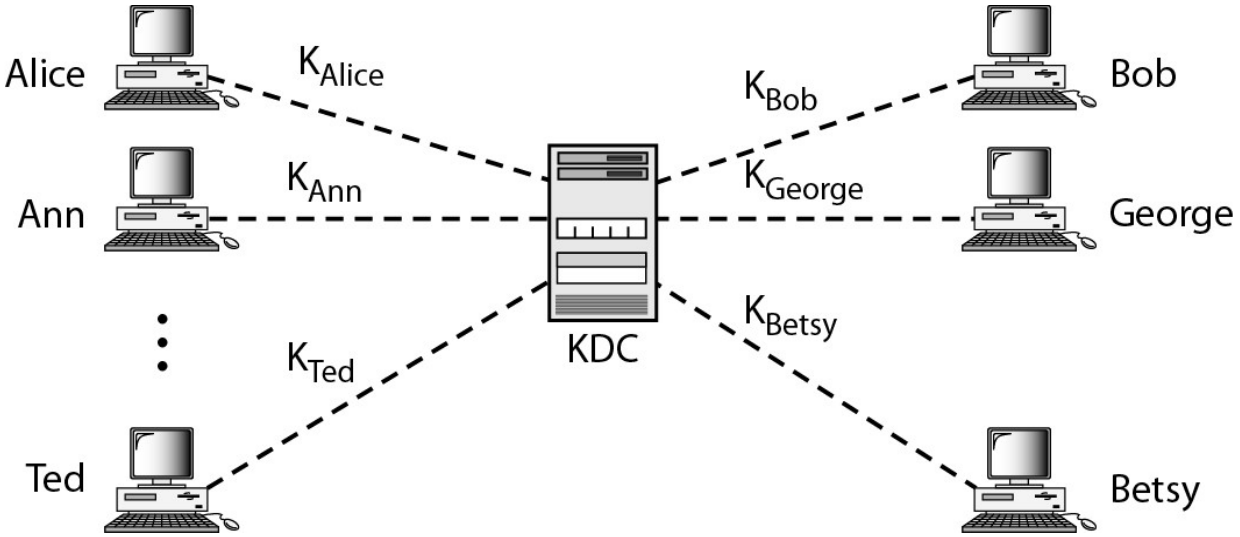
31-7 KEY MANAGEMENT

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

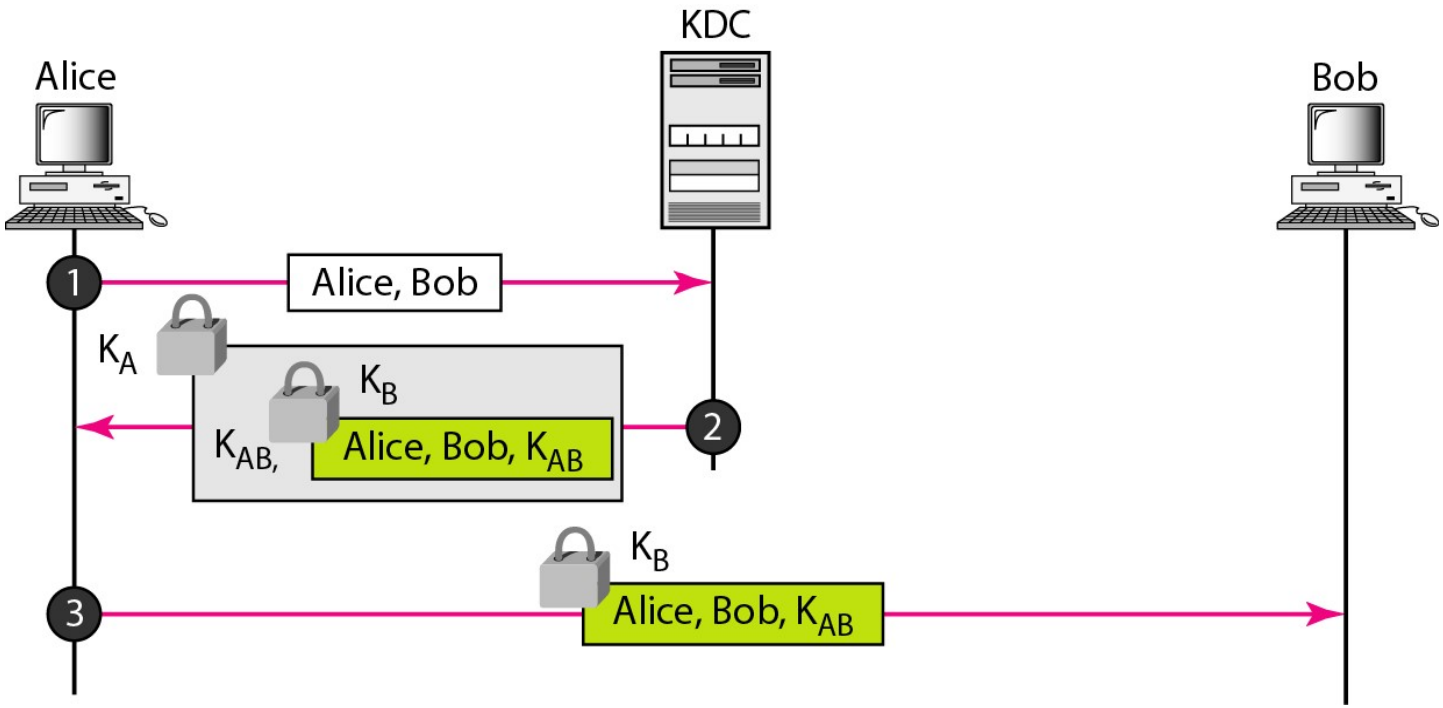
Figure 31.19 *KDC (key distribution center)*



Note

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

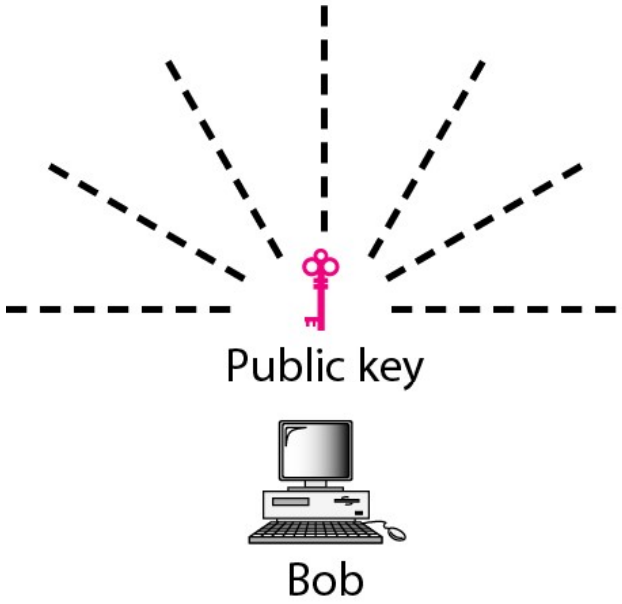
Figure 31.30 *Creating a session key between Alice and Bob using KDC*



Note

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

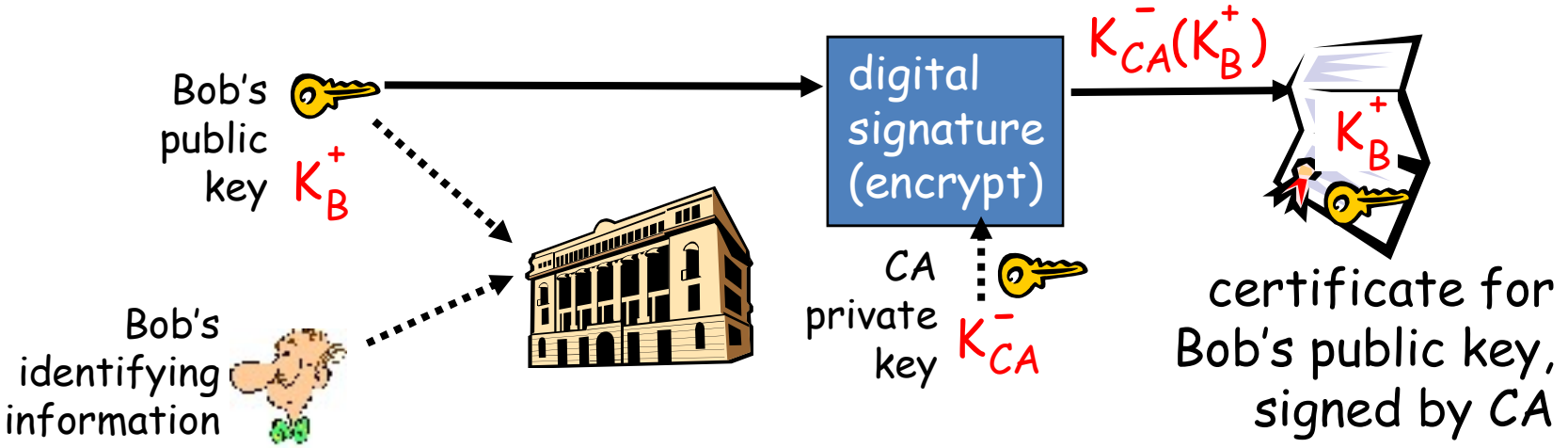
Figure 31.23 *Announcing a public key*



Problem: How can you know what you get is really Bob's public key?

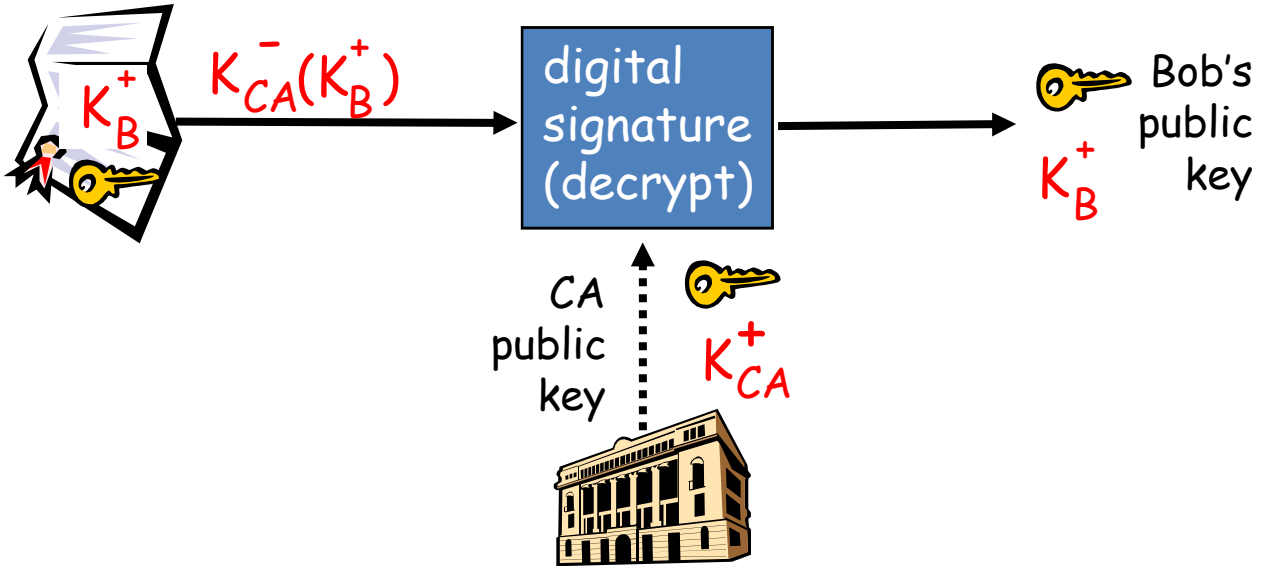
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



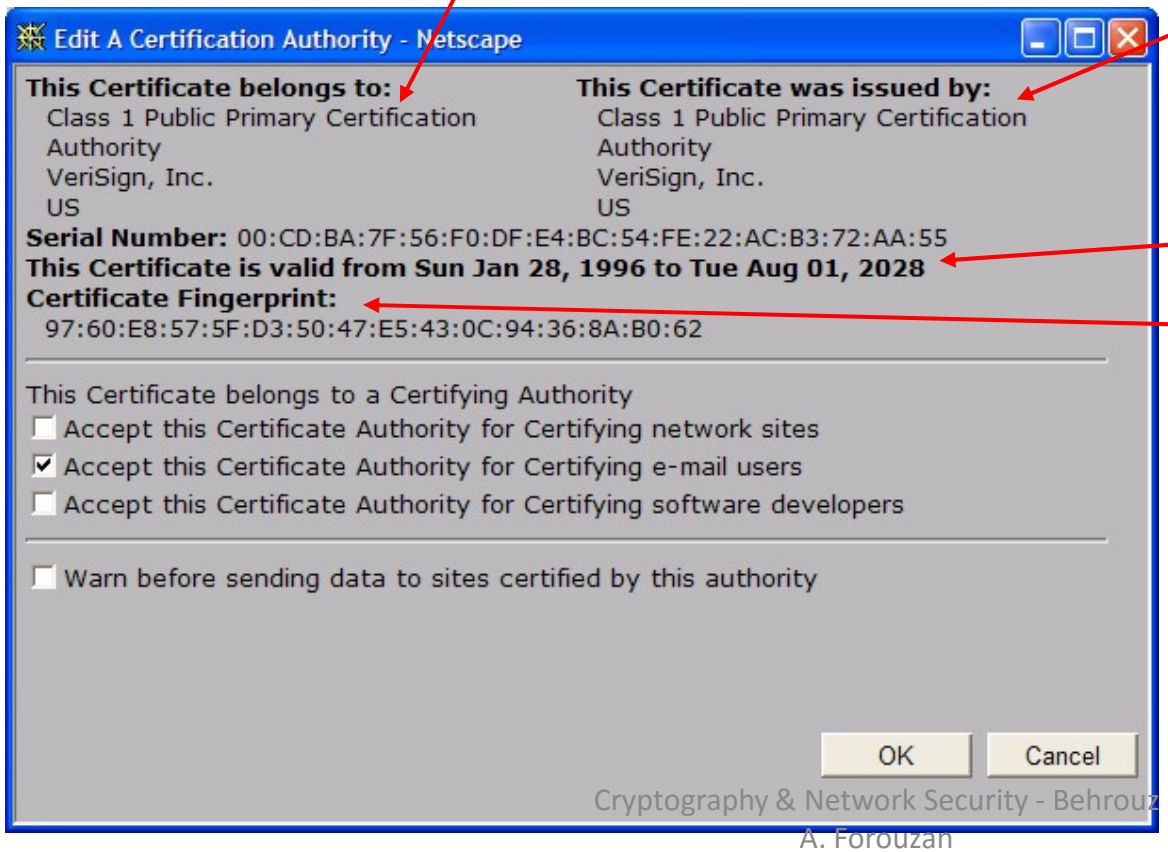
Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



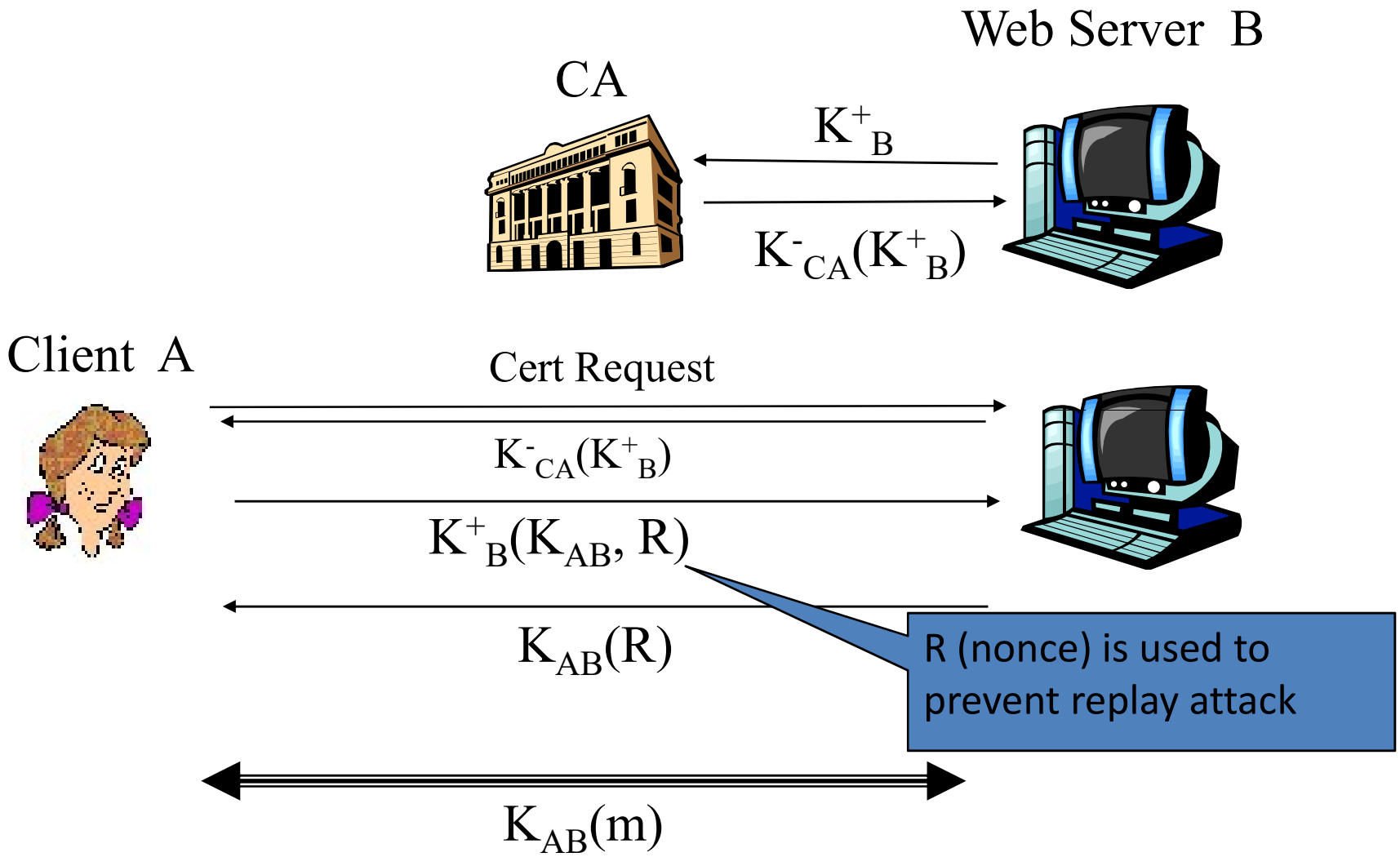
A certificate contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

Internet Web Security Architecture



Internet Web Security Conditions

- Clients' web browsers have built-in CAs.
- CAs are trustable
- Web servers have certificates in CAs.

- Q: What if a server has no certificate?
  - Example: SSH servers