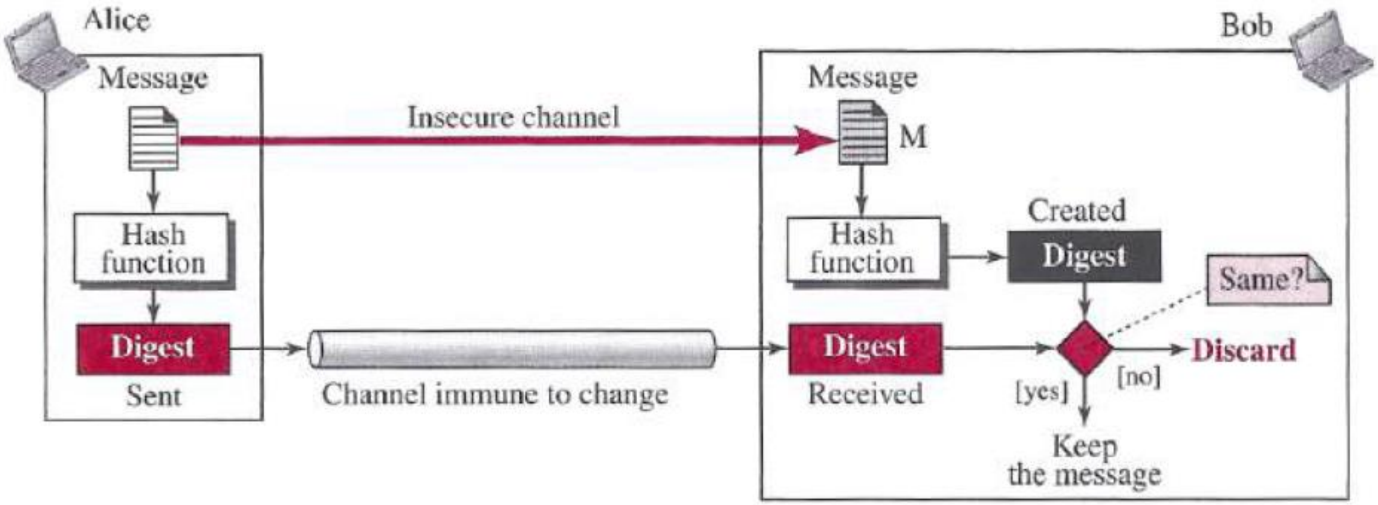
# CS321: Computer Networks

# Message Digest, Digital Signature, End-point Authentication


Dr. Manas Khatua
Assistant Professor
Dept. of CSE
IIT Jodhpur
E-mail: manaskhatua@iitj.ac.in

# Message Integrity

- In many instances, we must have integrity: the message should remain unchanged.
- One way to preserve the integrity of a document is through the use of a *fingerprint.*
- The electronic equivalent of the document and fingerprint pair is the *message* and *digest* pair.
- Message Digest is generated by a hash function



# Hash Function

- Cryptographic Hash Function is required to have the following properties:
  - takes an input, *m*, and computes a fixed size string *H(m)* known as a hash
  - It is computationally infeasible to find any two different messages *x* and *y* such that *H(x) = H(y)*

- Popularly used Hash Functions:
  - MD5
  - SHA-1

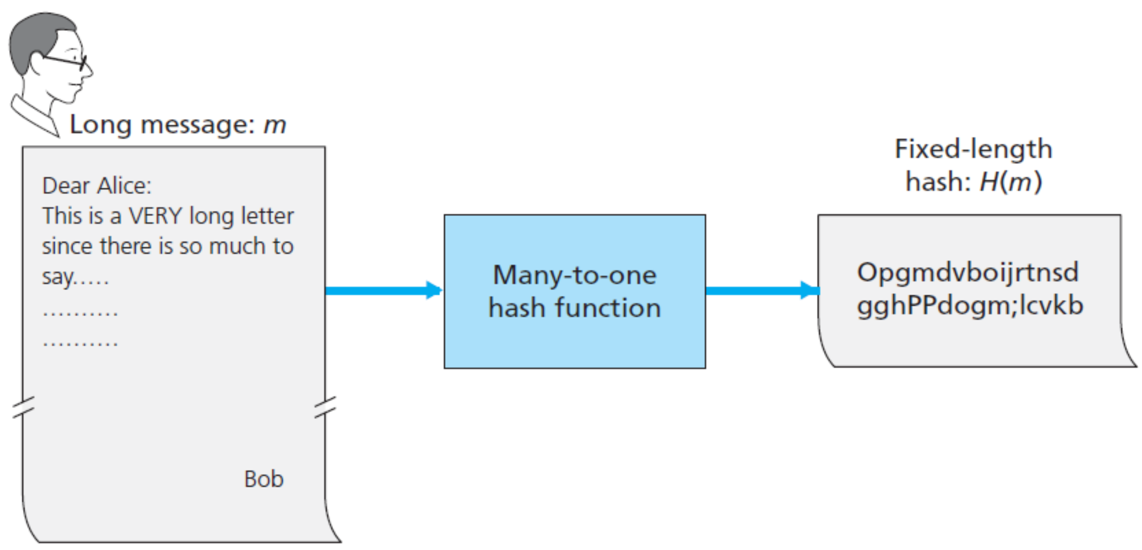
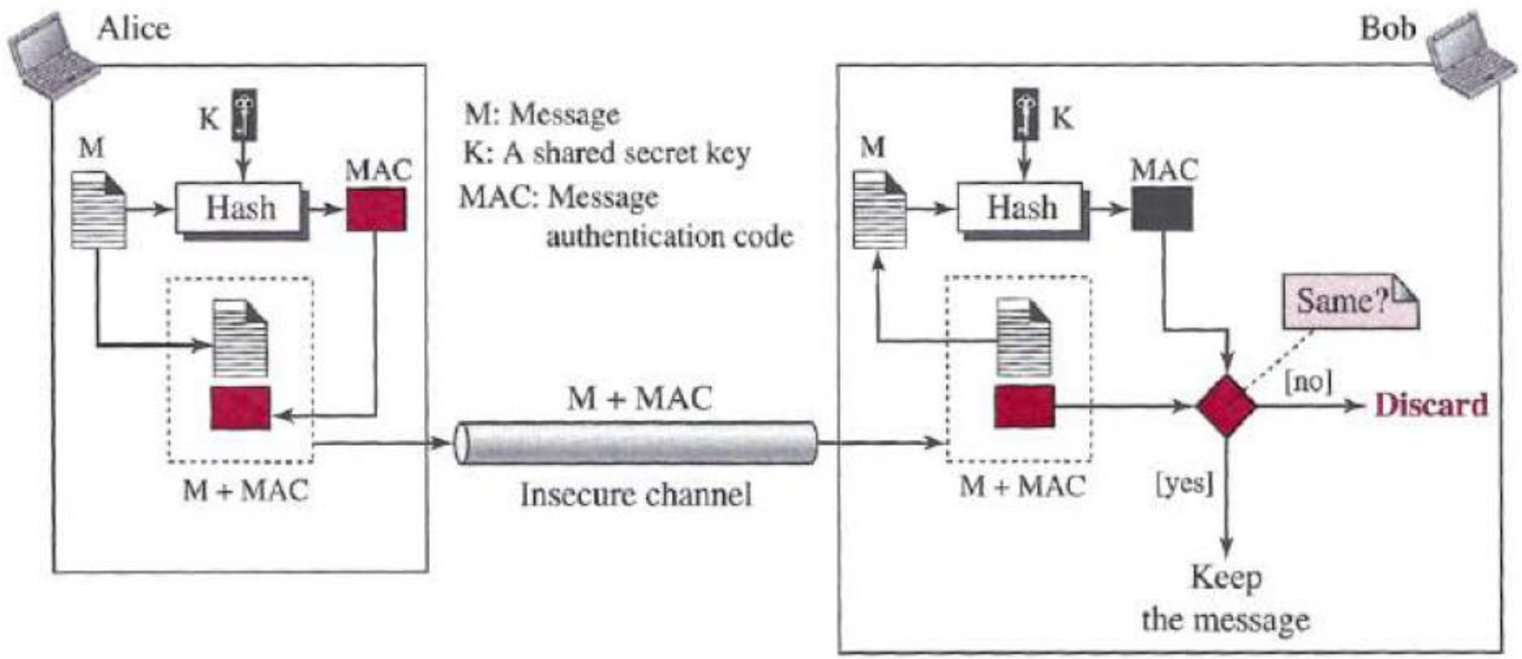



**Figure 8.7** ♦ Hash functions

# Message Authentication Code

- To authenticate a message, Bob needs to verify:
  - The message indeed originated from Alice
  - The message was not tampered with on its way to Bob
- A digest can be used to check the integrity of a message.
- 
- But, to ensure the integrity and authentication of the message, we need to include a secret shared between Alice and Bob in the process.
- This shared secret, which is nothing more than a string of bits, is called the **authentication key**.
- Message digest of message and authentication key is called Message authentication code (**MAC**).

# Cont…




- One nice feature of a MAC is that it does not require an encryption algorithm
- Application:
  - In the link-state routing algorithm, communicating entities are only concerned with message integrity and are not concerned with message confidentiality

# Digital Signature

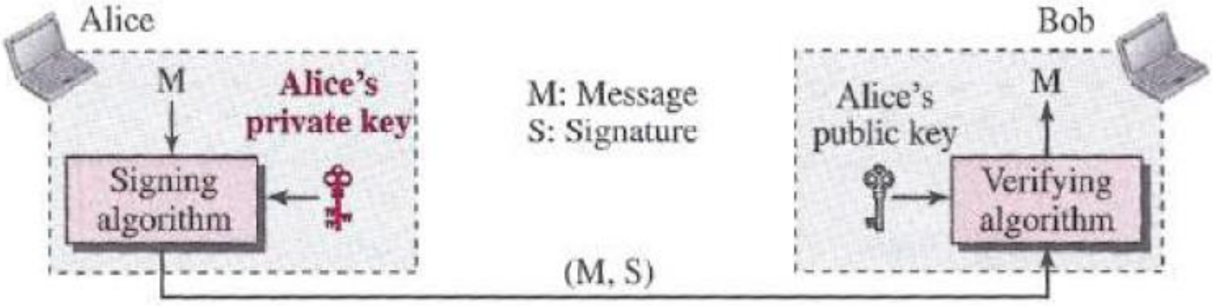
- Your signature attests to the fact that you (as opposed to someone else) have acknowledged and/or agreed with the document's contents.

- A digital signature is a cryptographic technique for achieving the same goals in a digital world.

- Digital signature must be verifiable and nonforgeable.
  - possible to prove that a document signed by an individual was indeed signed by that individual
  - *only* that individual could have signed the document

- MAC is not sufficient to certify sender authentication! Why?
  - MAC is created by appending his key (unique to Bob) to the message, and then taking the hash.
  - But for Alice to verify the signature, she must also have a copy of the key of Bob!
  - So, key would not be unique to Bob.
  - Hence, it is not nonforgeable

# Cont…




- A cryptosystem uses the public and private keys of the receiver;
- a digital signature uses the private and public keys of the sender.

- Does the digital signature $K_B^-(m)$ meet our requirements of being verifiable and nonforgeable?
  - Suppose Alice has $m$ and $K_B^-(m)$. She wants to prove.
  - Alice takes Bob's public key, $K_B^+$, and computes $K_B^+(K_B^-(m))$.
  - she produces $m$, which exactly matches the original document!

  - Alice claimed that whoever signed the message must have used the private key, $K_B^-$.
  - The only person who could have known the private key, $K_B^-$, is Bob.

- Thus the digital signatures provide message integrity, allowing the receiver to verify that the message was unaltered as well as the source of the message.

# Overhead in Digital Signature

- Signing data by encryption is that encryption and decryption are computationally expensive.
- A more efficient approach is to introduce hash functions into the digital signature
- Using a hash function, Bob signs the hash of a message rather than the message itself, i.e., $K_B^-(H(m))$.
- So, the computational effort required to create the digital signature is substantially reduced.
- Digital signature (DS) v/s Message authentication code (MAC)
  - To create a MAC of the message, we append an authentication key to the message, and then take the hash of the result. No encryption is involved.
  - To create a DS, we first take the hash of the message and then encrypt the message with our private key (using public key cryptography).

# Sending Digitally Signed Message

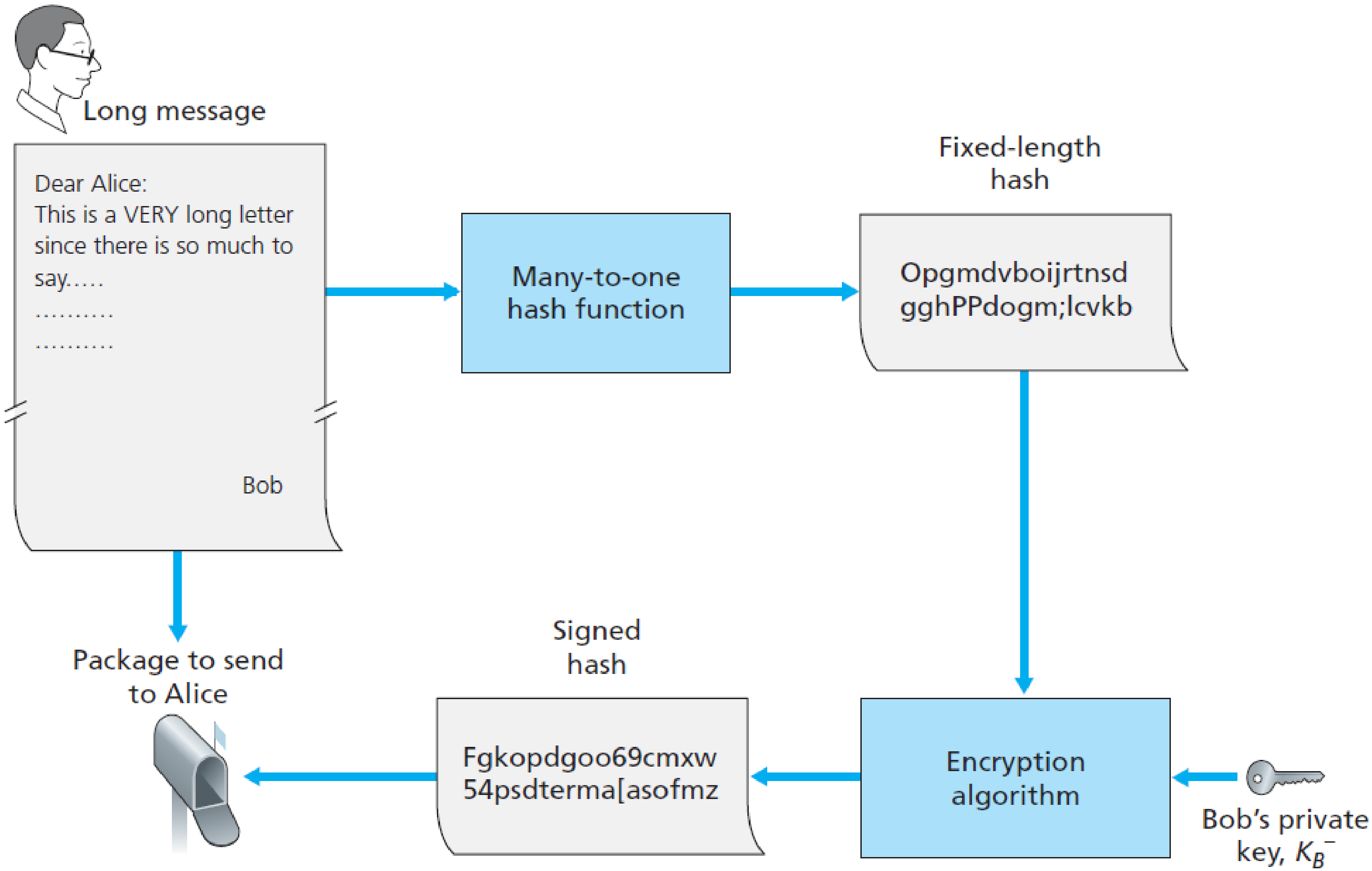



**Figure 8.11** ♦ Sending a digitally signed message

# Verifying a Signed Message

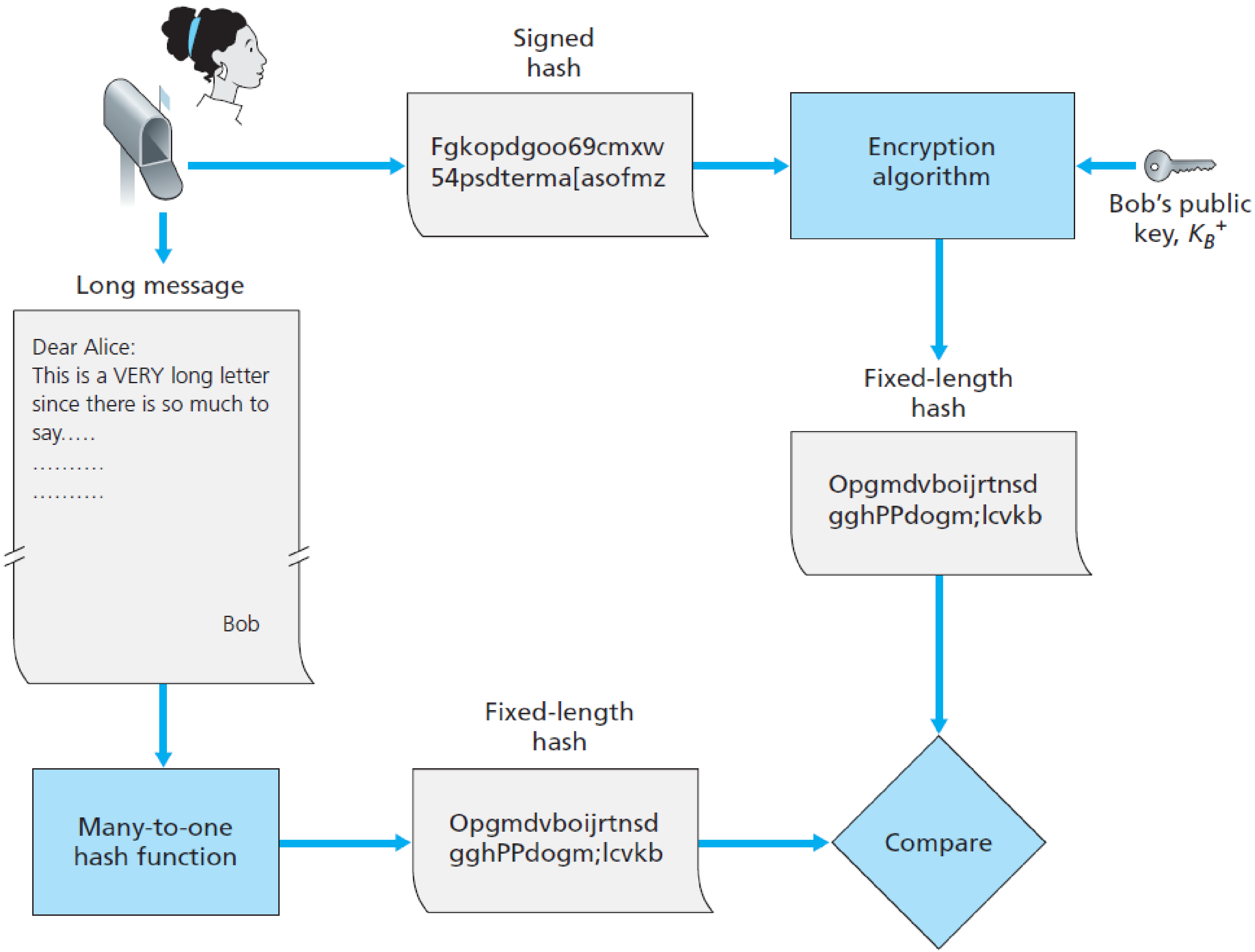


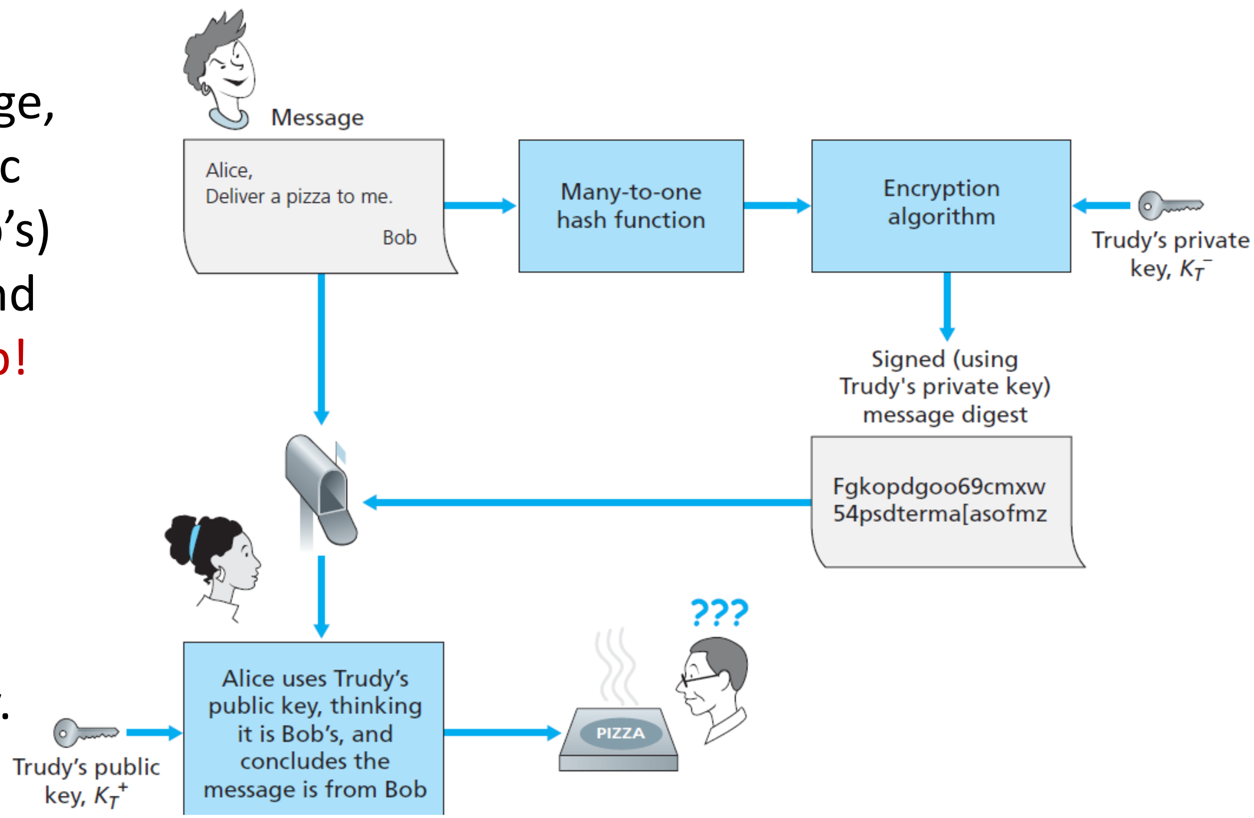

**Figure 8.12** ♦ Verifying a signed message

# Public Key Infrastructure

- Problem:
  - Let, Trudy sends a message to Alice in which she says she is Bob!
  - In this message she also includes her (Trudy's) public key, although Alice naturally assumes it is Bob's public key.
  - Trudy also attaches a digital signature, which was created with her own (Trudy's) private key.

After receiving the message, Alice applies Trudy's public key (thinking that it is Bob's) to the digital signature, and concludes it is sent by Bob!

Solution: We need public key certification, that is, certifying that a public key belongs to a specific entity.



# Cont…

- Binding a public key to a particular entity is done by a Certification Authority (CA)

- When Bob places his order he also sends his CA-signed certificate.

- Alice uses the CA's public key to check the validity of Bob's certificate and extract Bob's public key.

- [RFC 1422] describes CA-based key management for use with secure Internet e-mail.

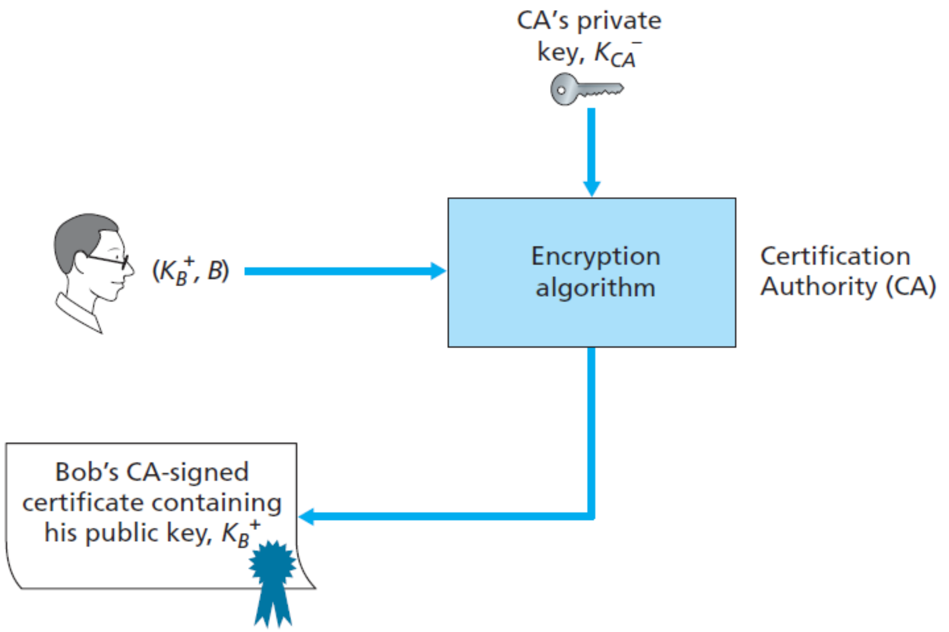
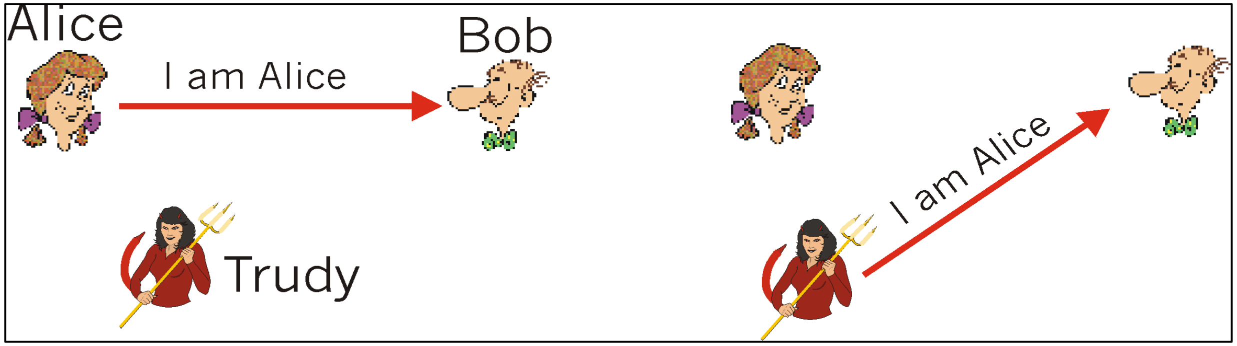



**Figure 8.14** ♦ Bob has his public key certified by the CA

# End-point Authentication

- **End-point authentication** is the process of one entity proving its identity to another entity over a computer network
  - for example, a user proving its identity to an email server

- We focus here on authenticating a "live" party, at the point in time when communication is actually occurring.




- As humans, we authenticate each other in many ways
  - We recognize each other's faces when we meet,
  - we recognize each other's voices on the telephone,
  - the customs official checks us against the picture on our passport

# Simple scenario

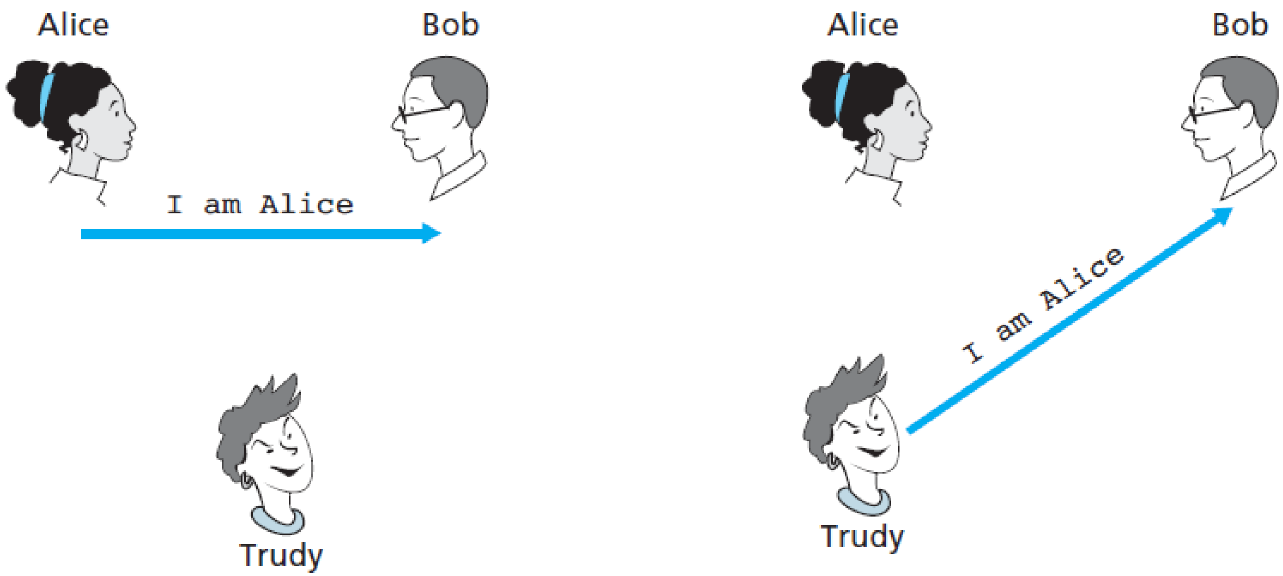



**Figure 8.15** ♦ Protocol *ap1.0* and a failure scenario

- Let’s assume that Alice needs to authenticate herself to Bob.
- Problem: there is no way for Bob actually to know that the person sending the message “I am Alice” is indeed Alice.

# Use of well-known address

- Assume that Alice has a well-known network address (e.g., an IP address) from which she always communicates.
- Bob could attempt to authenticate Alice by verifying the source address on the IP datagram.
- Problem: it can not avoid IP spoofing attack which violates the authentication of Alice.

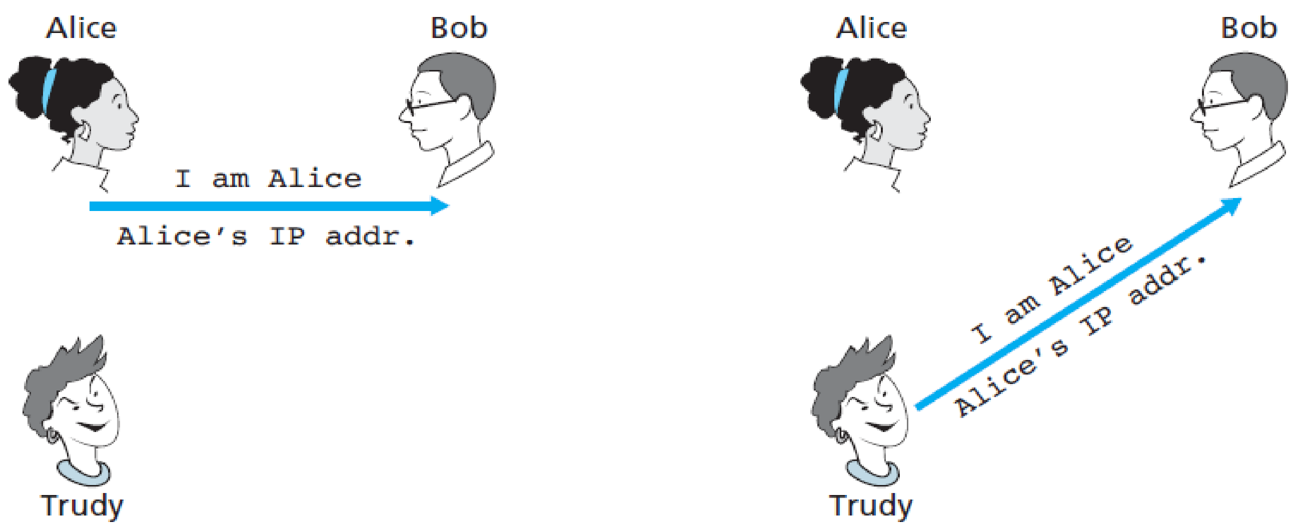
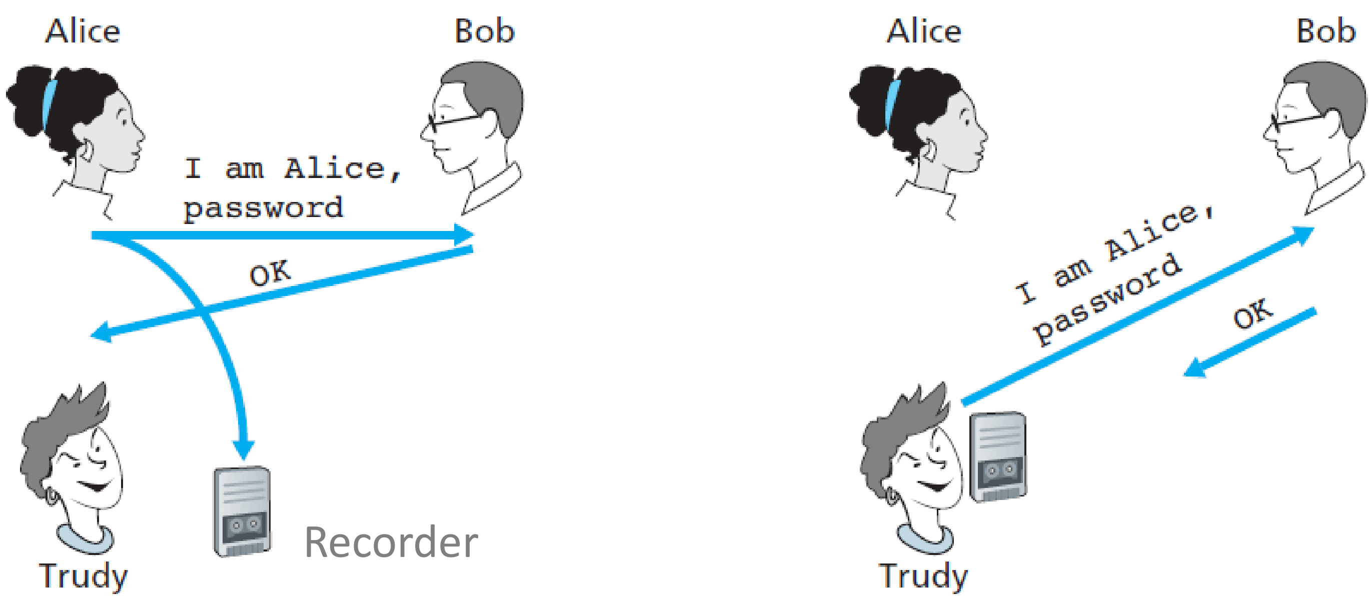



**Figure 8.16** ♦ Protocol *ap2.0* and a failure scenario

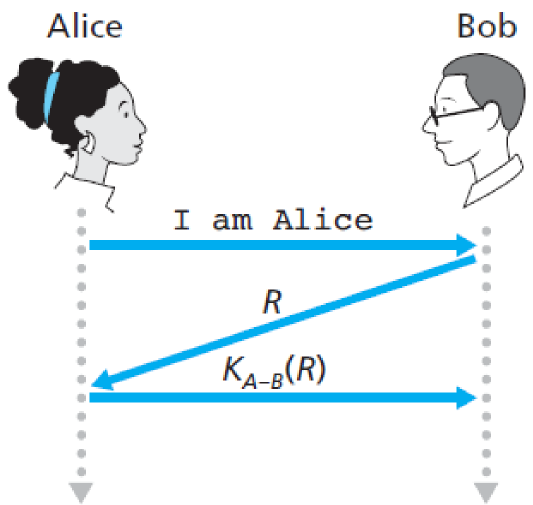
# Password based authentication

- classic approach to authentication is to use a secret password
- The password is a shared secret between the authenticator and the person being authenticated
- Gmail, Facebook, telnet, FTP, and many other services use password authentication.
- Password based authentications are widely used.
- It has serious security flaw indeed !
- **playback attack:** record the encrypted version of the password, and play back



# Nonce

- The previous failure scenario resulted from the fact that Bob could not distinguish between the original authentication of Alice and the later playback of Alice's original authentication
- That is, Bob could not tell if Alice was live
- Solution: use of *nonce* with cryptography (symmetric key)
- A **nonce** is a number that a protocol will use only once in a lifetime. That is, once a protocol uses a nonce, it will never use that number again.




- $R$ is the nonce, and $K_{A\text{-}B}$ is the private key
- Bob can be sure that Alice is both
  - who she says she is (since she knows the secret key value needed to encrypt $R$), and
  - live (since she has encrypted the nonce, $R$, that Bob just created).

# Multi-factor authentication (MFA)

- It is a method of confirming a user's claimed identity in which a user is granted access only after successfully presenting 2 or more pieces of evidence (or factors) to an authentication mechanism
  - knowledge (something they and only they know),
  - possession (something they and only they have),
  - inherence (something they and only they are)

- Example, withdrawing of money from a ATM
  - 2F authentication is used
  - Bank ATM card (something that the user possesses)
  - PIN (something that the user knows)

- Two-step authentication
  - utilizing something they know (password) and
  - a second factor **other** than possession or inherence

- Example, one time password (OTP)
  - second step is the user repeating back something that was sent to them through an out-of-band mechanism.

# Thanks!