University of Southern California

Viterbi School of Engineering

EE450
Computer Networks

Network Security

Shahin Nazarian

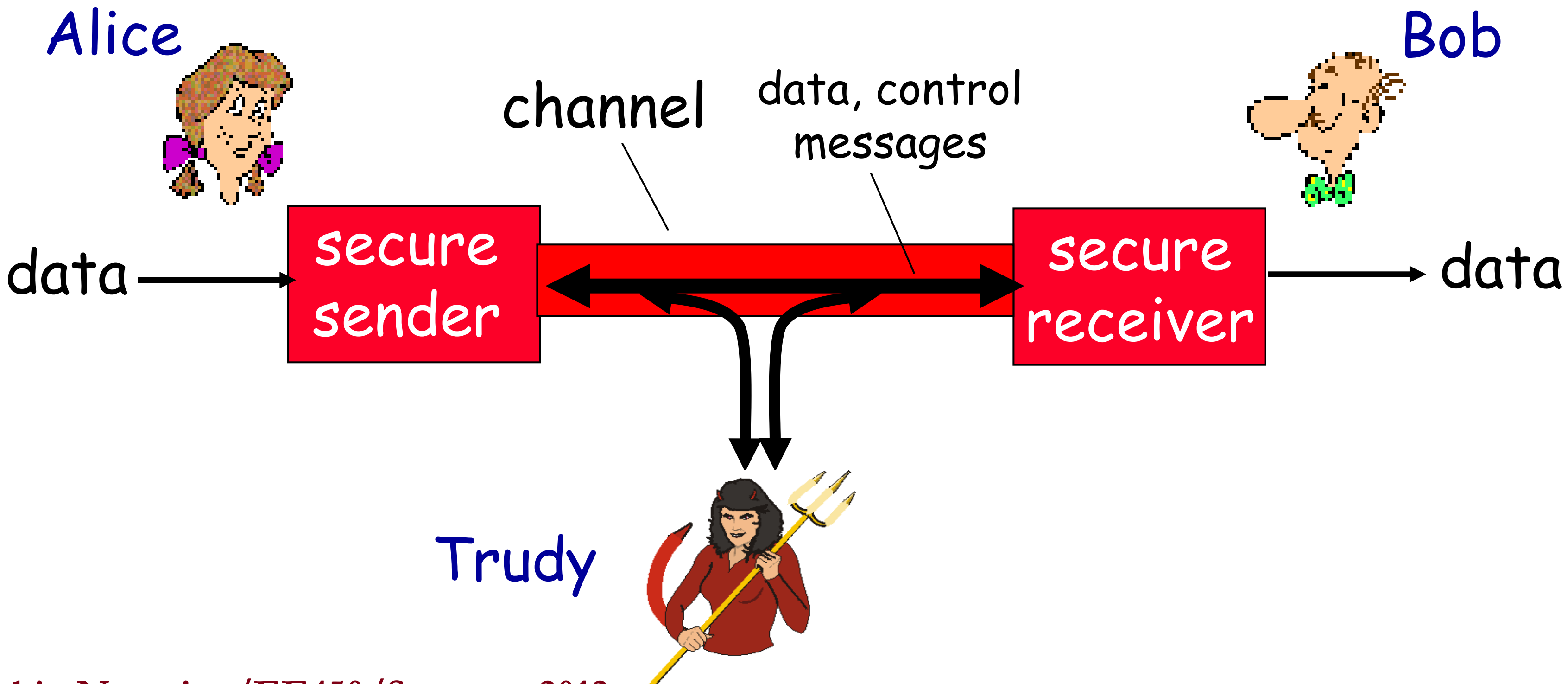
Summer 2012

What is Network Security?

- Only sender, intended receiver should “understand” message contents
  - Sender encrypts message
  - Receiver decrypts message
- Sender, and receiver want to confirm identity of each other
- Sender, and receiver want to ensure message not altered (in transit, or afterwards) without detection
- Services must be accessible and available to users

Friends & Enemies: Alice, Bob, & Trudy

- Well-known in network security world!
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy or Eve (intruder) may intercept, delete, add messages



Who Might Bob and Alice Be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- On-line banking client/server
- DNS servers
- Routers exchanging routing table updates
- Two email applications that want to exchange secure email
- …

There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: a lot!

- *Eavesdrop:* intercept messages
- Actively *insert* messages into connection
- *Impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *Hijacking:* “take over” ongoing connection by removing sender or receiver, inserting himself in place
- *Denial of service*: prevent service from being used by others (e.g., by overloading resources)

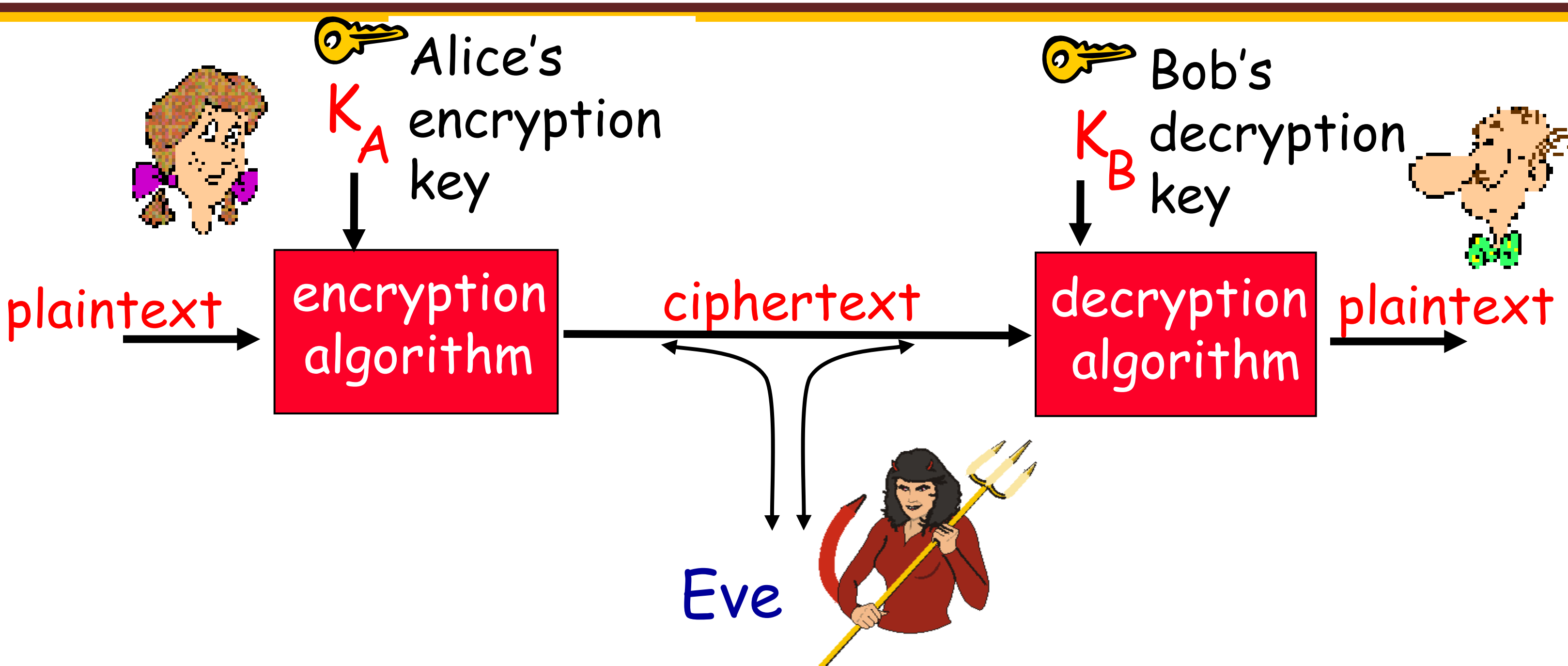
Message Confidentiality

- The concept of how to achieve message confidentiality or privacy has not changed for thousands of years

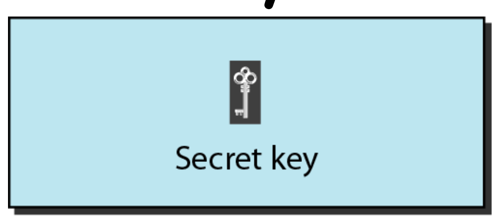
- The message must be encrypted at the sender site and decrypted at the receiver site

- This can be done using either
  - Symmetric-key cryptography or
  - Asymmetric-key cryptography

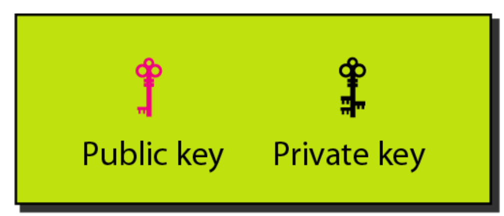
The Language of Cryptography



- **Symmetric key cryptography: private key**
- **Asymmetric (Public-key) cryptography: public key**
  - **Knowing the public key, Eve should not be able to calculate the private key**

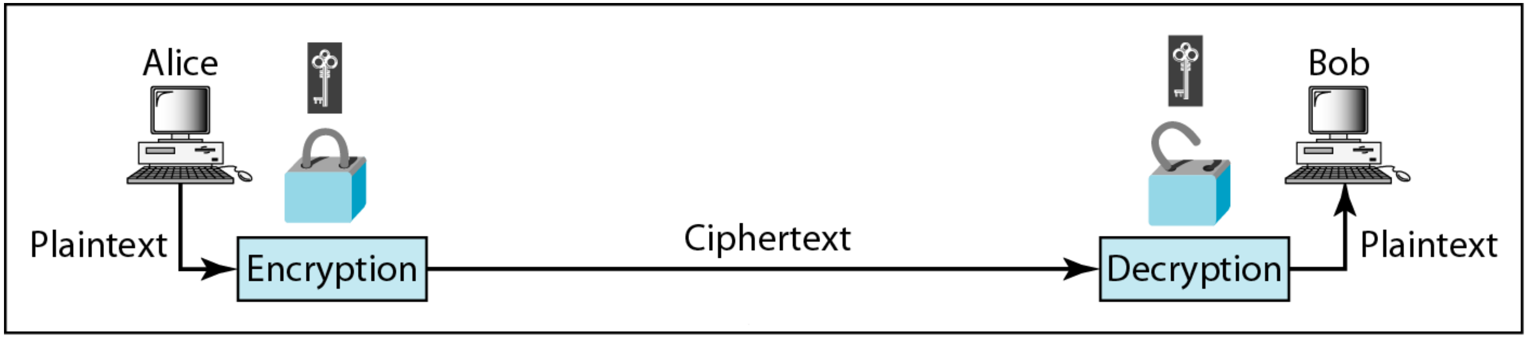


Symmetric-key cryptography

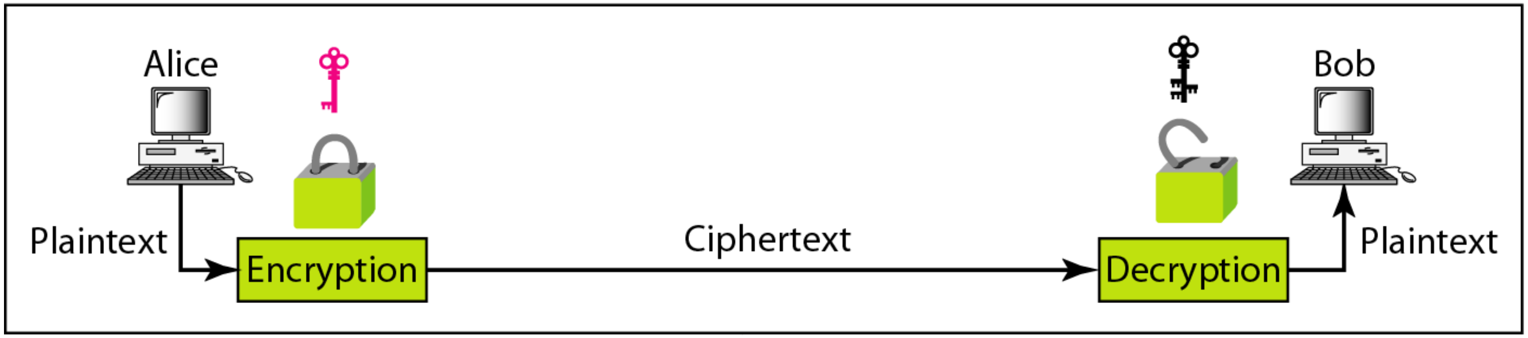


Asymmetric-key cryptography

Comparison between Two Categories of Cryptography



a. Symmetric-key cryptography

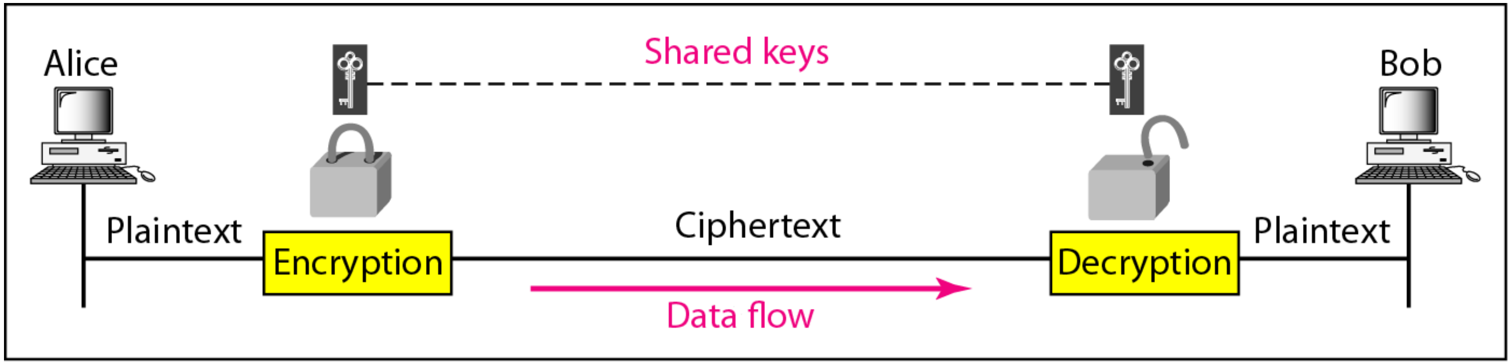


b. Asymmetric-key cryptography

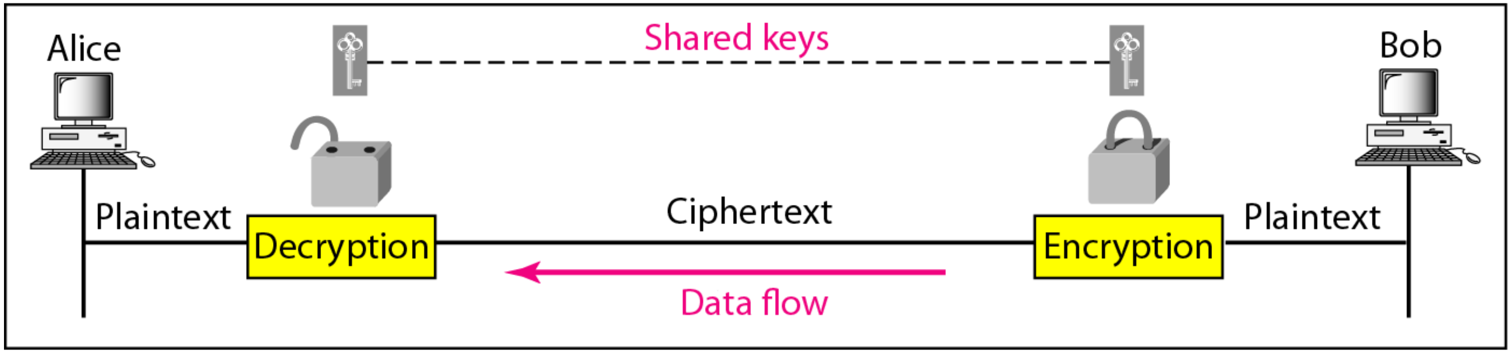
Symmetric-key cryptography

Asymmetric-key cryptography

Message Confidentiality Using Symmetric Keys in Two Directions

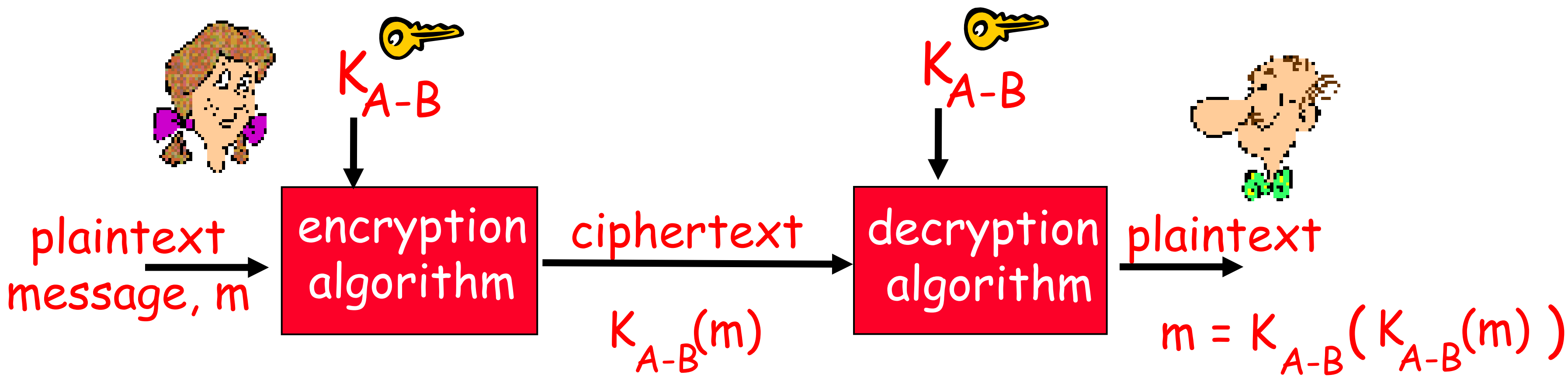


a. A shared secret key can be used in Alice-Bob communication



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

Symmetric Key Cryptography (Cont.)



Symmetric key crypto: Bob and Alice share know same (symmetric) key: K_{A-B}

- E.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Symmetric Key Cryptography – Example

Substitution cipher: Substituting one thing for another

- Monoalphabetic cipher: substitute one letter for another

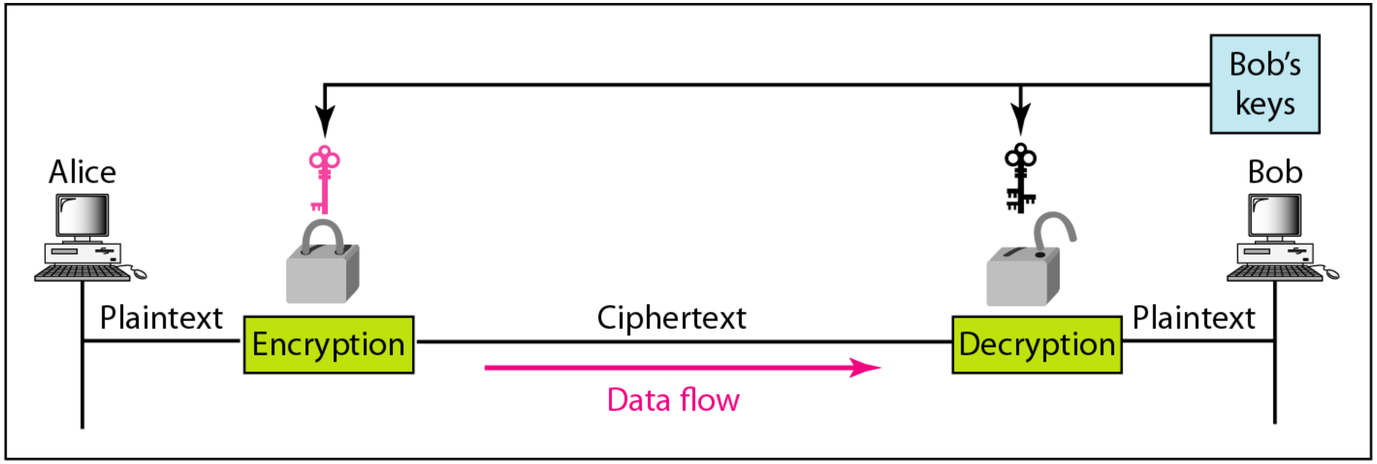
```
 Plaintext:  abcdefghijklmnopqrstuvwxyz

Ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

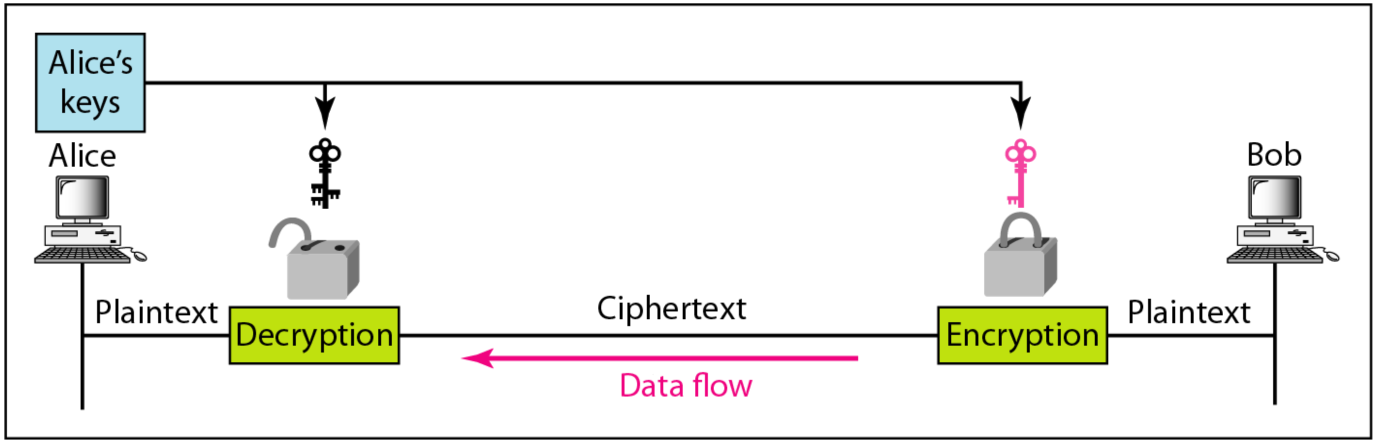
```
 Plaintext: bob. i love you. alice
Ciphertext: nkn. s gktc wky. mgsbc
```

- Q: How hard to break this simple cipher?
  - Can be very simple; e.g., if Trudy knows for certain that both names appear in the above ciphertext, he can immediately determine seven of 26 letters requiring 10^9 fewer possibilities to be checked by brute-force method

Message Confidentiality Using Asymmetric Keys

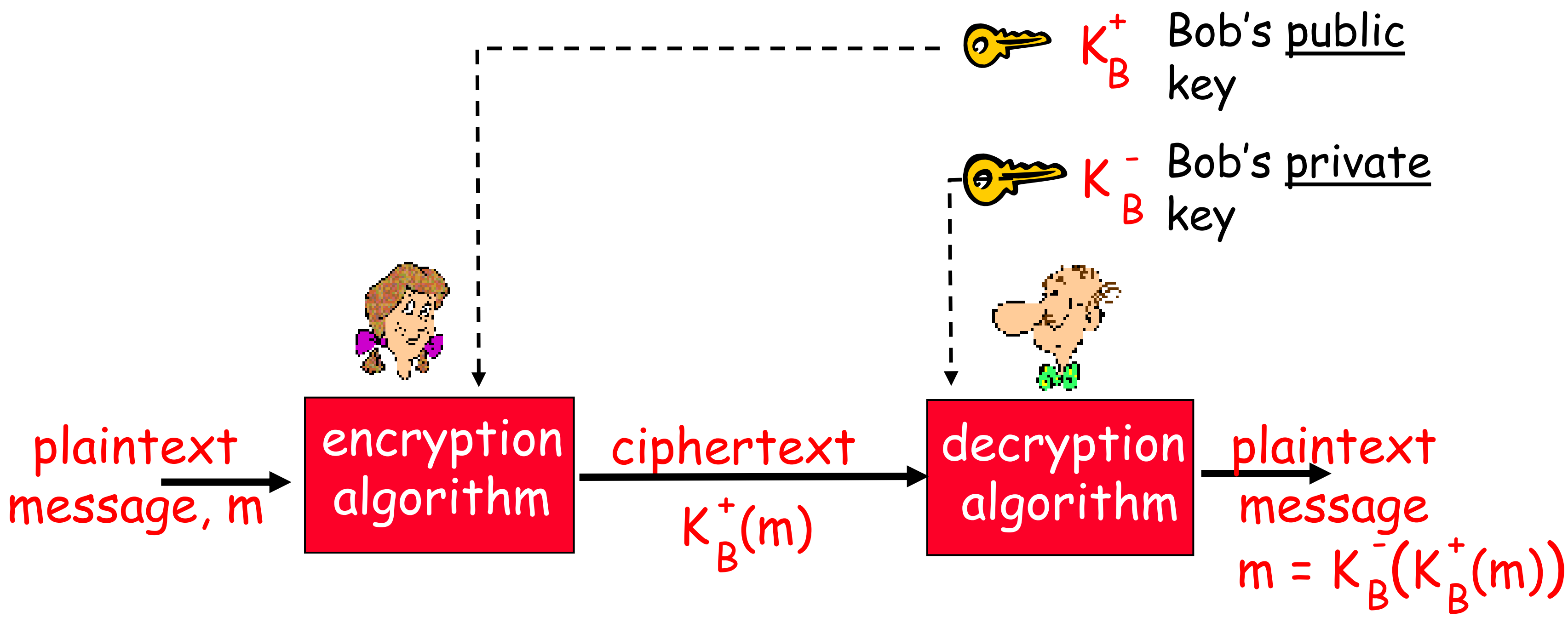


a. Bob's keys are used in Alice-Bob communication



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

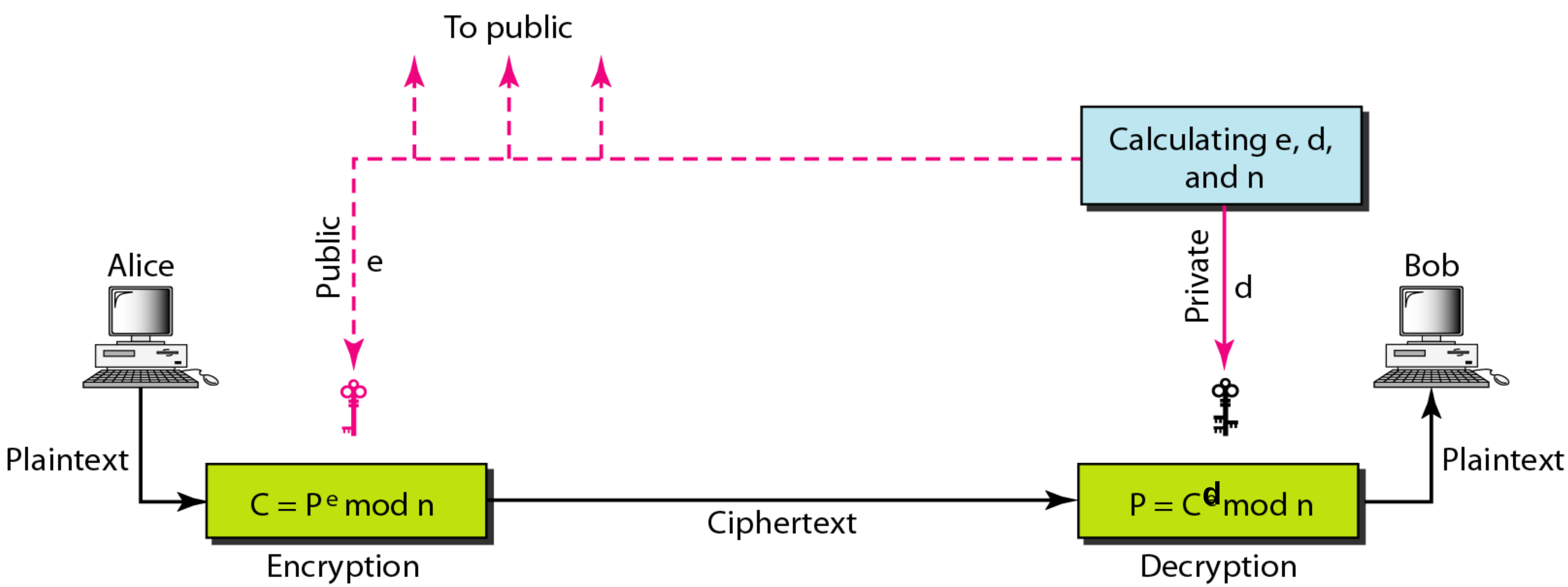
Public (Asymmetric) Key Cryptography (Cont.)



- $K^-_B(K^+_B(m)) = K^+_B(K^-_B(m)) = m$

The most famous example is the RSA (Rivest, Shamir, Adleman) algorithm

RSA


To public
Calculating e, d, and n
Public
e
Private
d
Alice
Bob
Plaintext
C = P e mod n
Encryption
Ciphertext
P = C d mod n
Decryption
Plaintext

RSA: Choosing Keys

1. Choose two large prime numbers p, q (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime")

4. Choose d such that $ed-1$ is exactly divisible by z (in other words: ed mod $z = 1$)

5. *Public* key is the pair of numbers $\underbrace{(n, e)}_{K_B^+}$
Private key is the pair $\underbrace{(n, d)}_{K_B^-}$

* z is also kept secret

RSA: Encryption, Decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$

$e=5$ (so e, z relatively prime)

$d=29$ (so $ed-1$ exactly divisible by z

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	1524832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	481968572106750915091411825223071697	12	l

RSA: Why is that

$$m = (m^e \bmod n)^d \bmod n$$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: Another Important Property

The following property will be *very* useful:

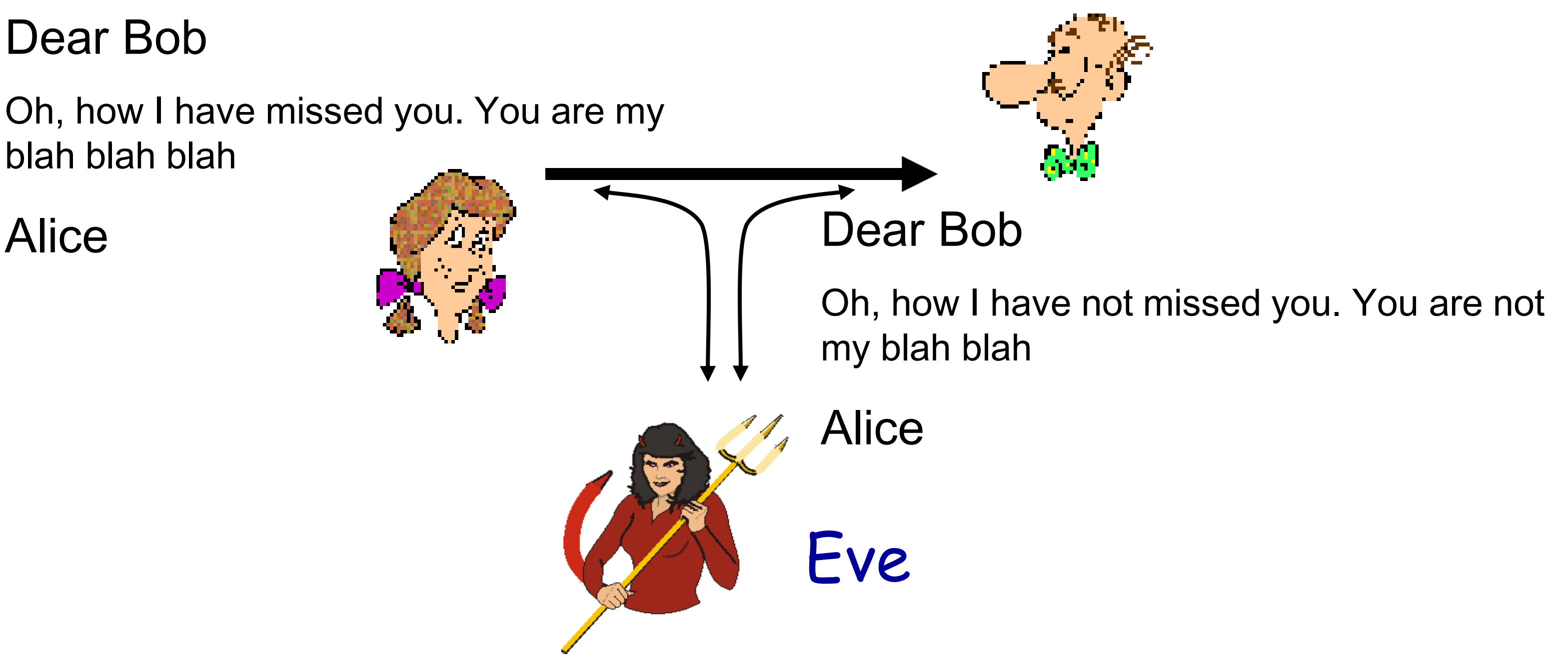
$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

Result is the same!

Message Integrity and Authentication

Bob receives msg from Alice, wants to ensure:

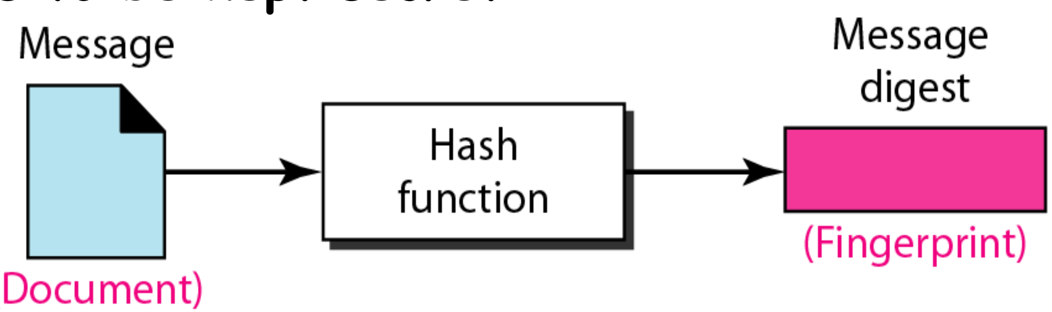
- **Message originally came from Alice**
- **Message not changed since sent by Alice**



- **Use a simple example to show that checksum is not a powerful function to apply for message integrity**

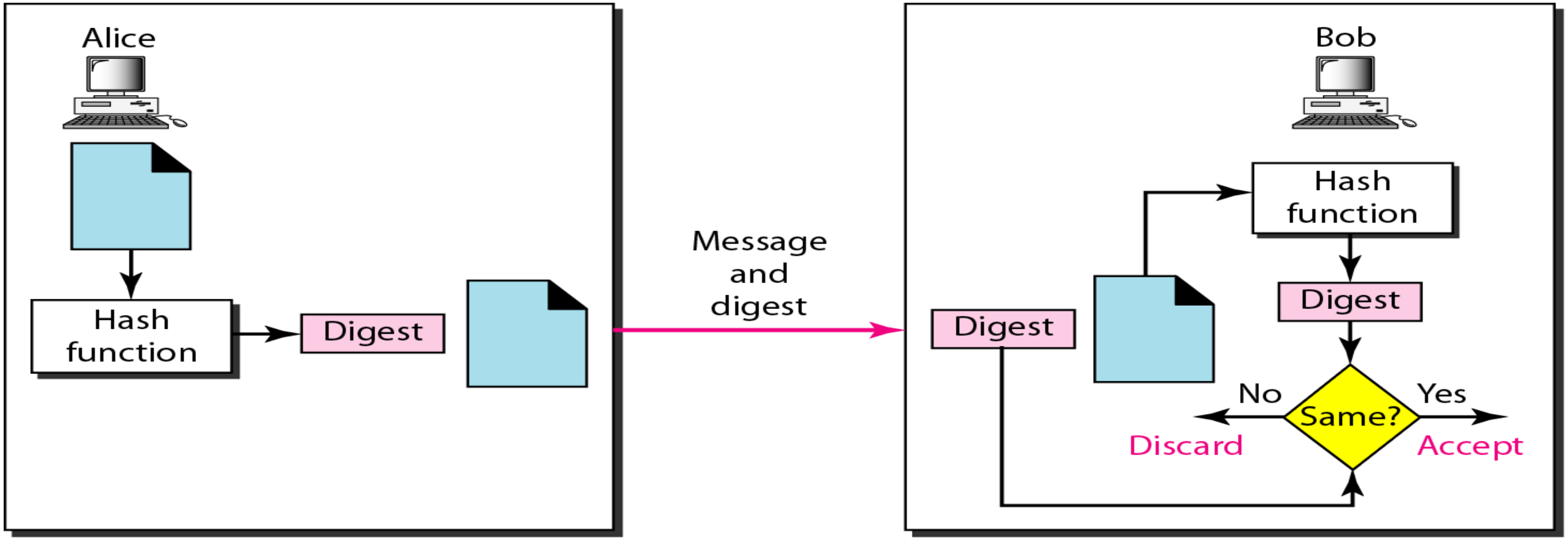
Message Integrity

- Encryption and decryption provide secrecy, or confidentiality, but not integrity
- E.g., Alice may write a will to distribute her estate upon her death. The will does not need to be encrypted. After her death anyone can examine her will. The integrity of the will however needs to be preserved. Alice does not want the content of the will to be changed (note that here in this example, we do not even need secrecy, but instead must have integrity)
- To preserve the integrity of a document, both the document and the fingerprint are needed
- A digest created by a hash function is normally called a Modification Detection Code (MDC)
- The message digest needs to be kept secret



Checking Integrity

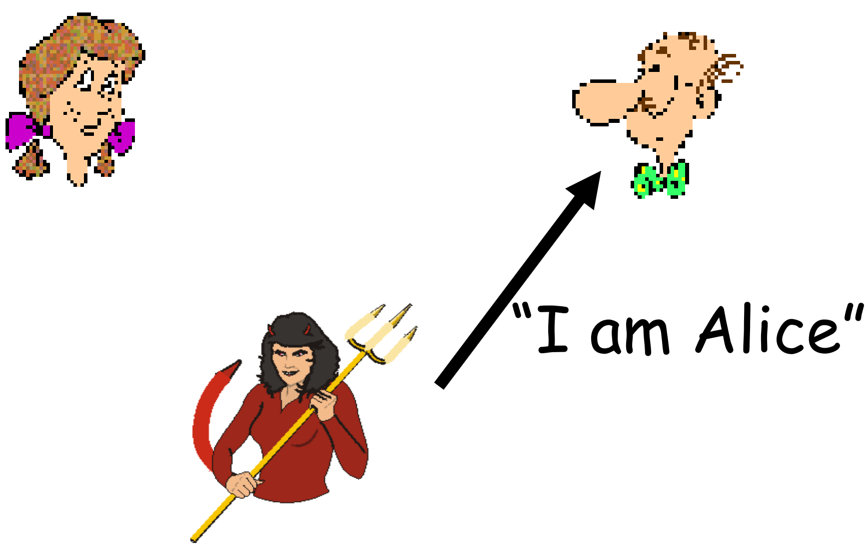
- It is computationally infeasible to find any two different messages x and y such that $H(x) = H(y)$
- The message digest ($H(m)$) is created at the sender and is sent with the message to the receiver (i.e., $\{m, H(m)\}$ sent)
- Receiver creates the hash function again and compares the new message digest with the one received. If both are the same the receiver is sure the original message has not been changed



Authentication

Goal: **Bob wants Alice to "prove" her identity to him**

Protocol ap1.0: Alice says "I am Alice"



in a network,
Bob can not "see"
Alice, so Trudy simply
declares
herself to be Alice

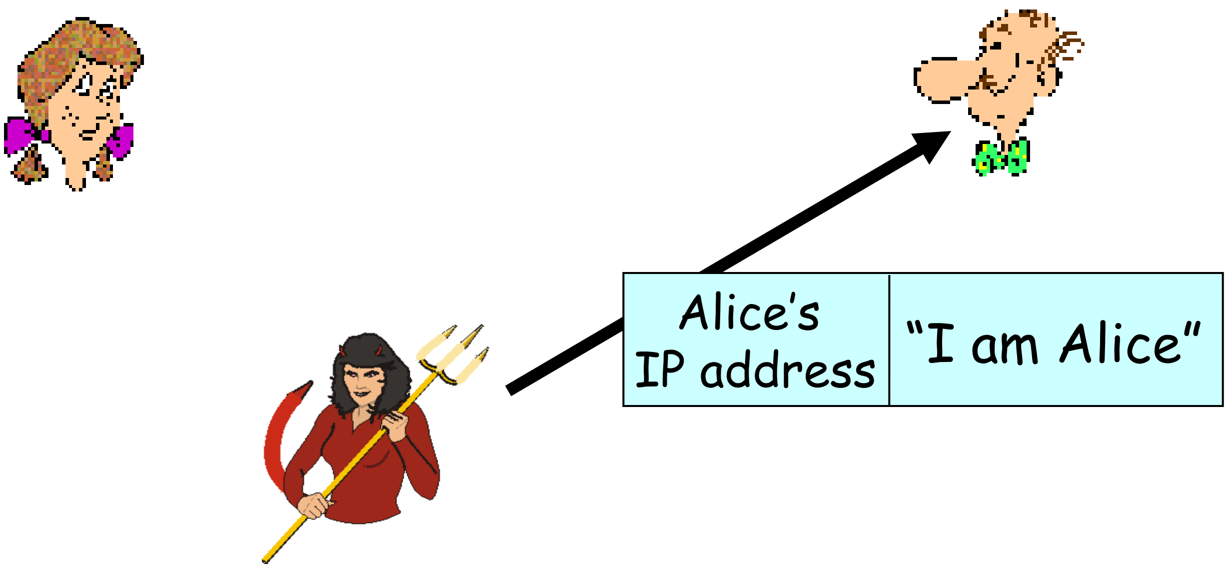
Authentication: Another Try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Failure scenario??

Authentication: Another Try (Cont.)

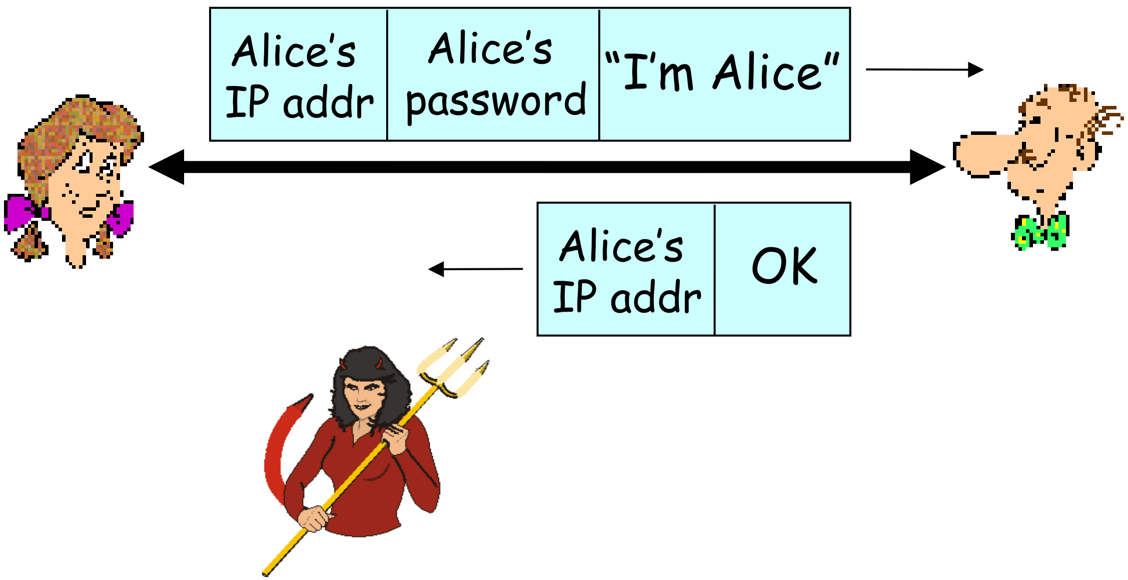
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: Another Try (Cont.)

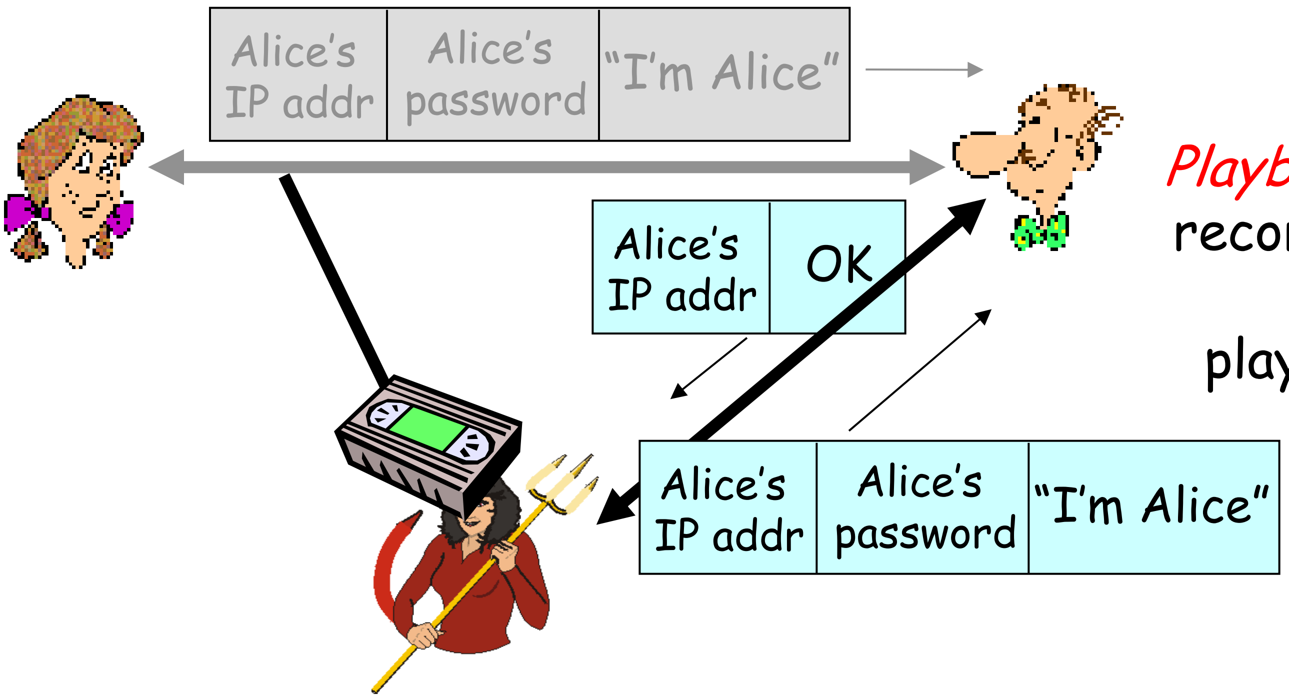
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it



Failure scenario??

Authentication: Another Try (Cont.)

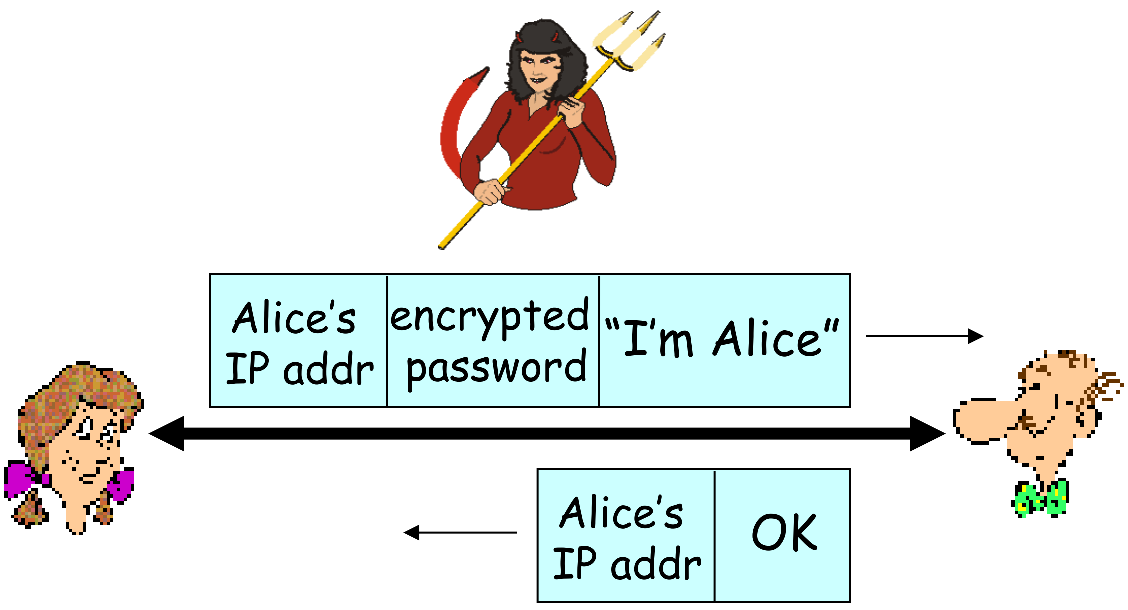
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it



Playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: Another Try (Cont.)

<u>Protocol ap3.1:</u> Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it



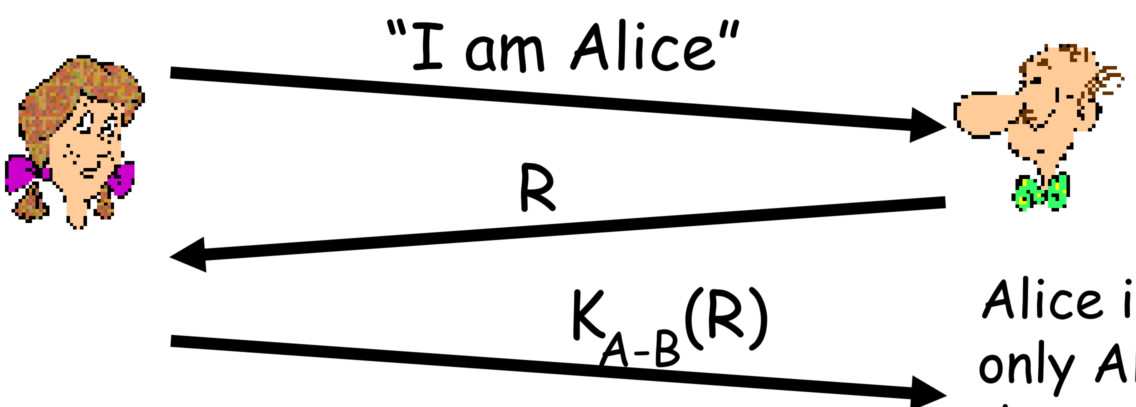
Failure scenario??

Authentication: Yet Another Try

<u>Goal:</u> avoid playback attack

<u>Nonce:</u> number (R) used only *once –in-a-lifetime*

<u>ap4.0:</u> to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

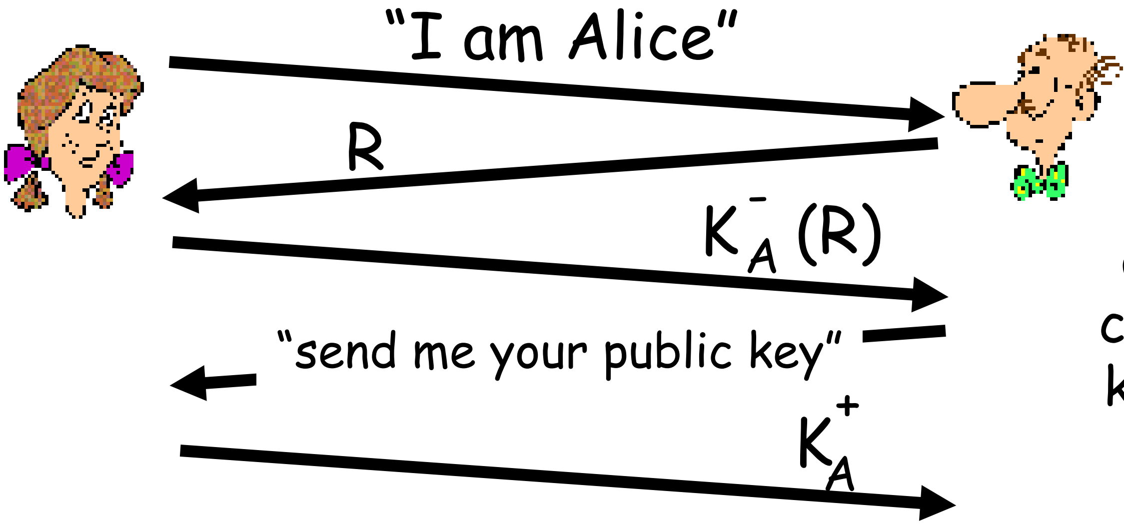
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

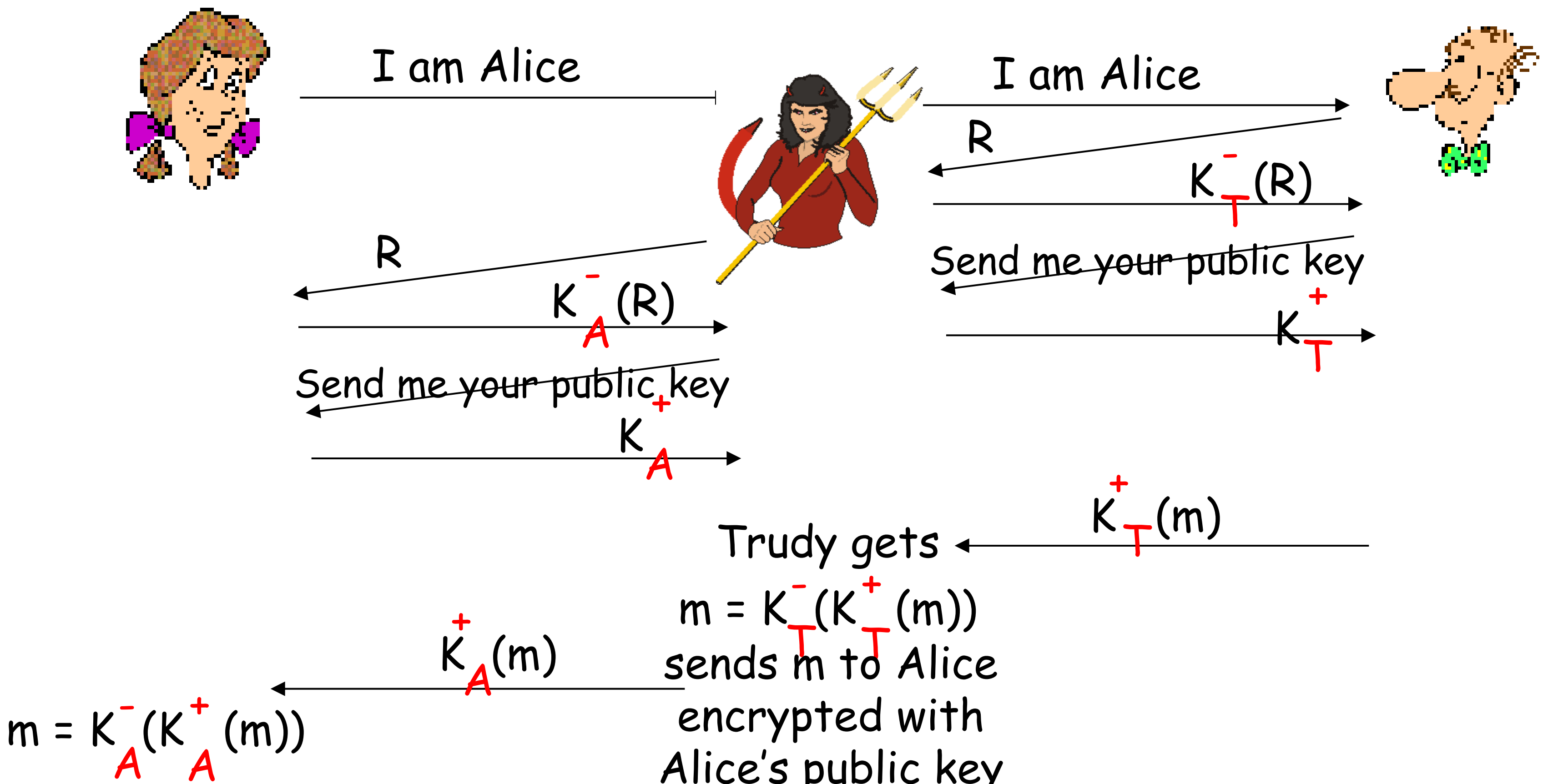
$K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key, that encrypted R such that

$K_A^+(K_A^-(R)) = R$

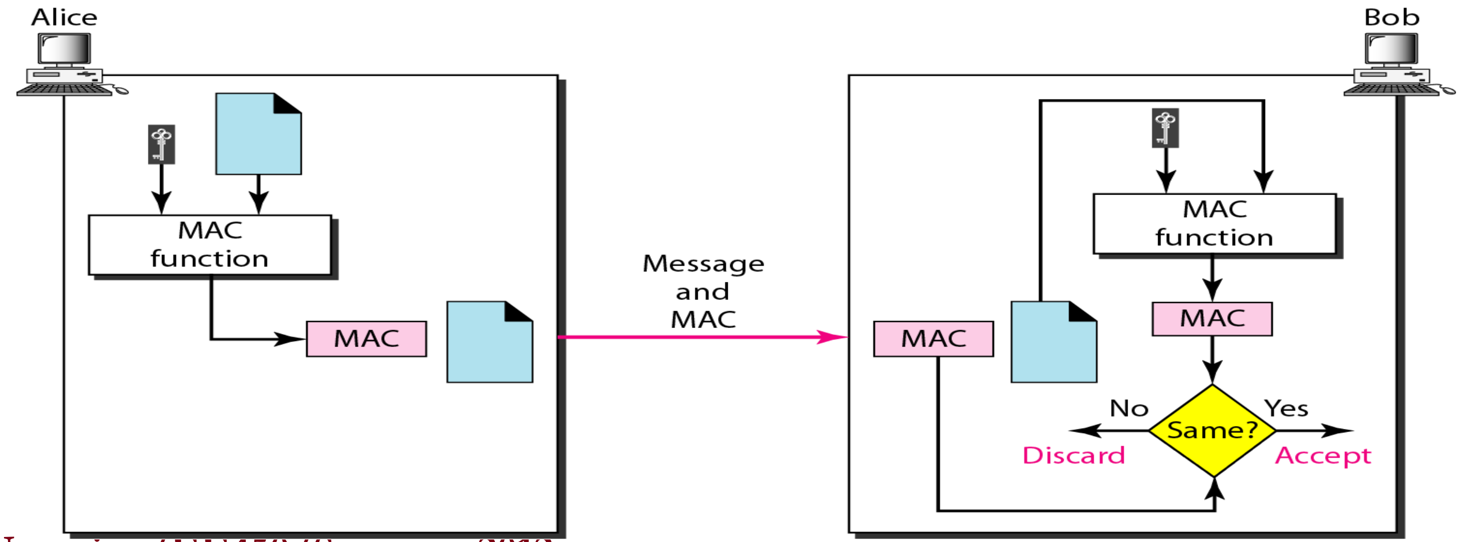
ap5.0: Security Hole

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

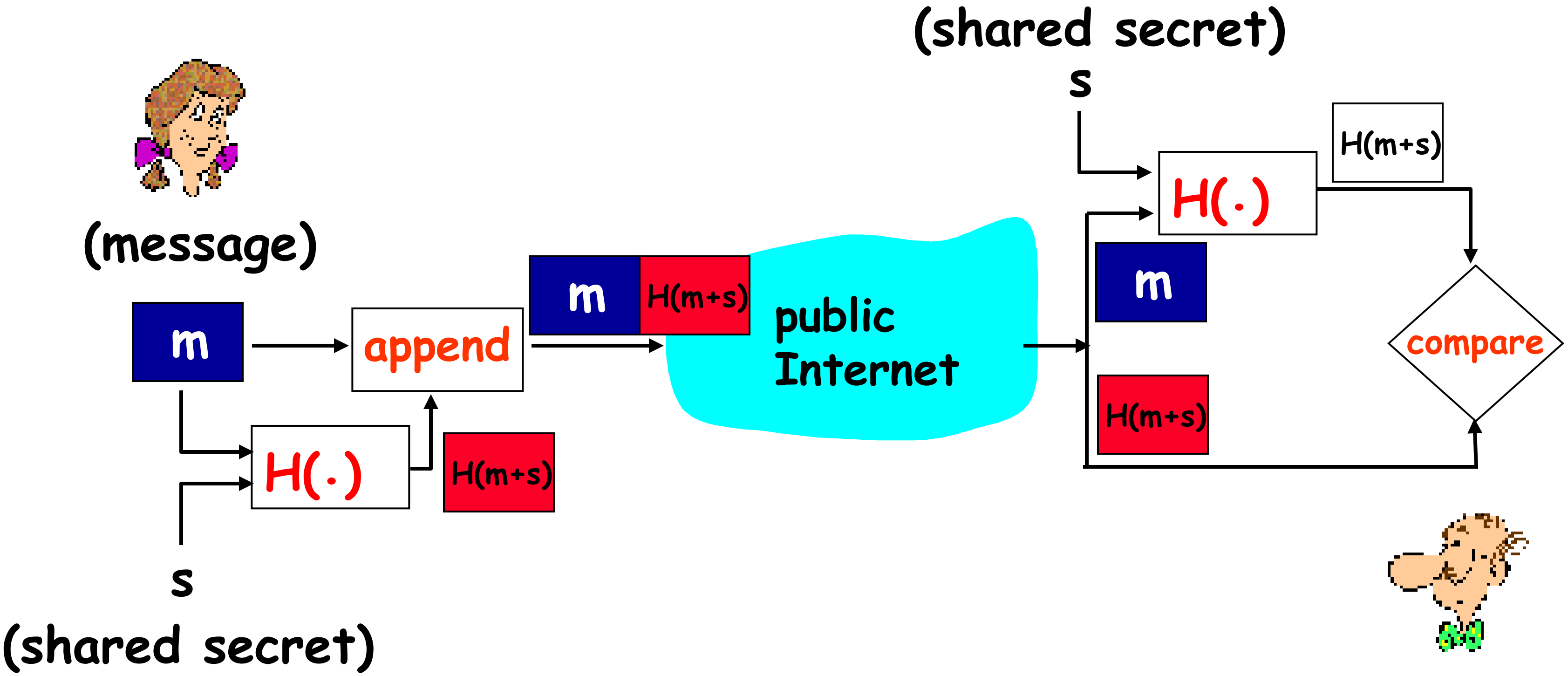


Checking Authentication

- A hash function guarantees the integrity of message (that it's not changed.) however it does not authenticate the sender of the message
- To provide message authentication we need to change a MDC to a Message Authentication Code (MAC)
- MDC uses a keyless hash function. MAC uses a keyed hash function; which includes the symmetric key between Alice and Bob (K_{A-B})



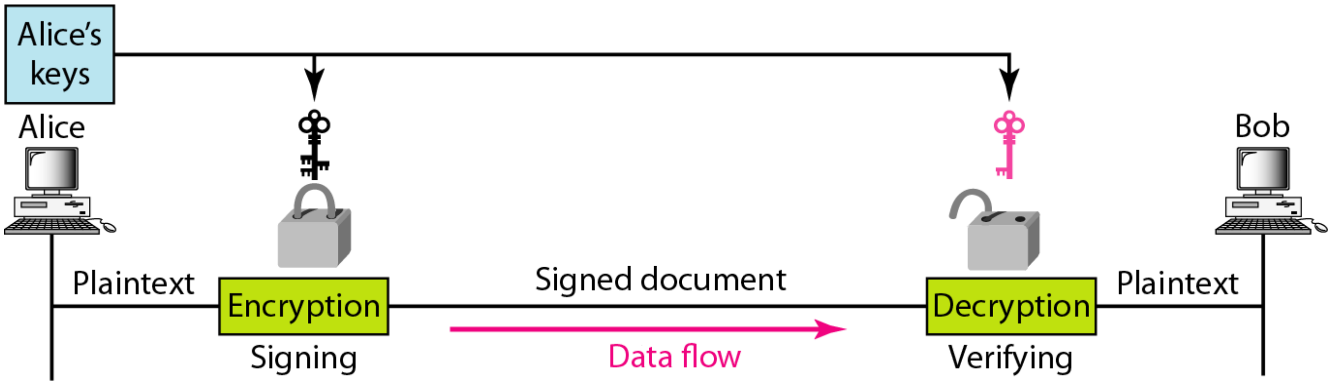
Digital Signature



- Although a MAC can provide message integrity and authentication, it has a drawback. It needs a symmetric key that must be established between Alice and Bob. A digital signature can use a pair of asymmetric keys (a public one and a private one)

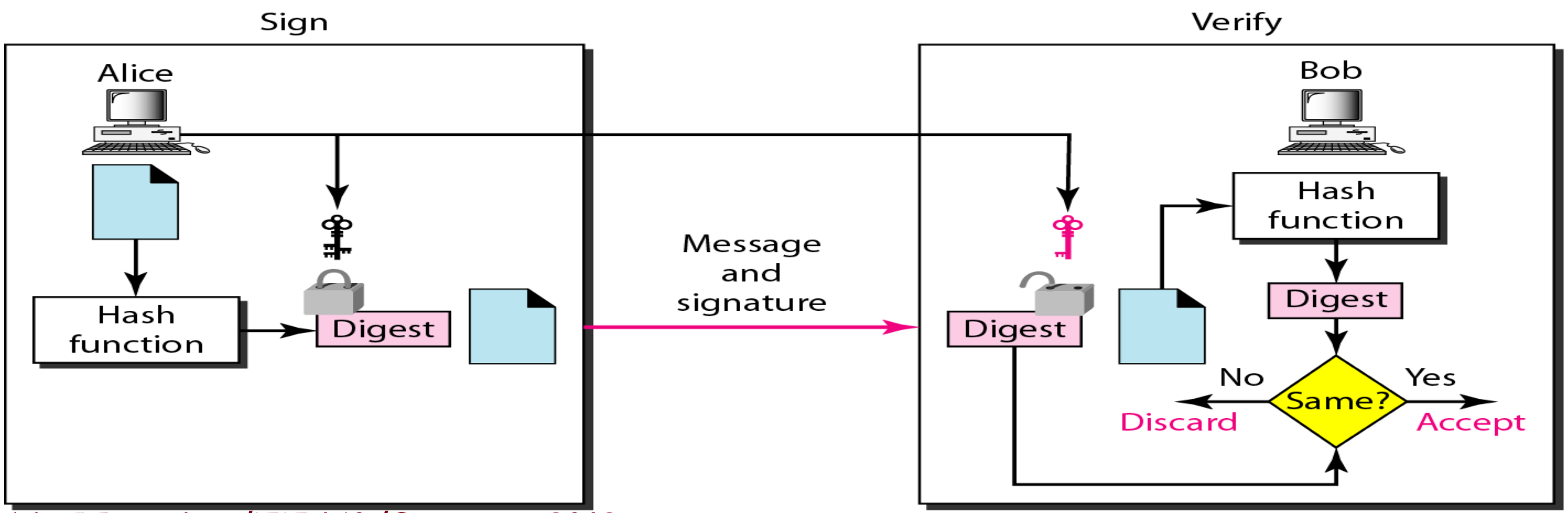
Digital Signature (Cont.)

- Approach one: Signing the message itself



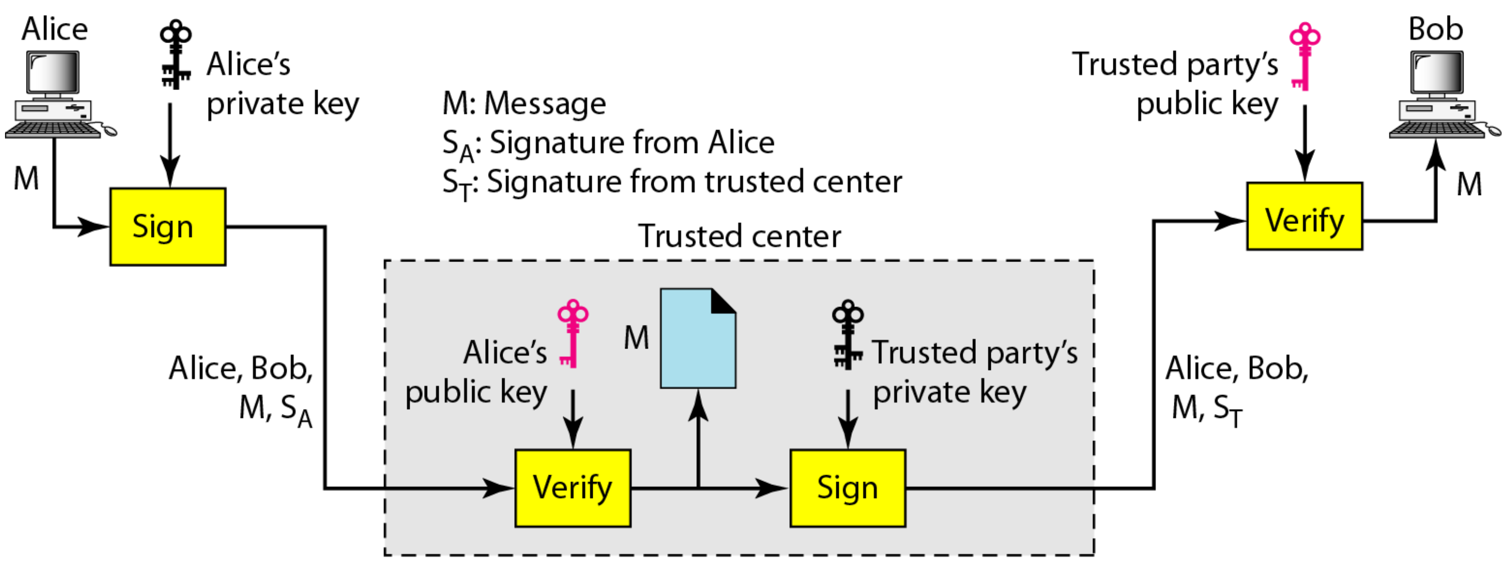
Drawback: messages are normally long

- Approach two: Signing the digest



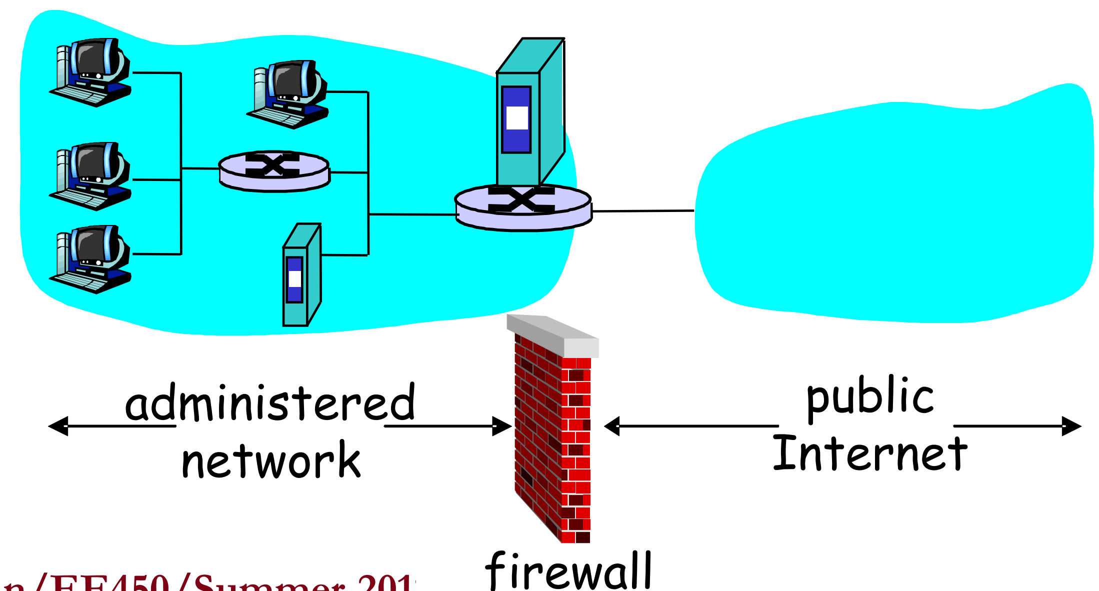
Using a Trusted Center for Nonrepudiation

- If Alice signs a message and then denies it can Bob later prove that Alice actually signed it? No; one solution is a trusted 3rd party. A trusted party can also solve many other problems concerning security services and key exchange



Firewalls

- All previous security measures cannot prevent Eve from sending harmful messages to a system. To control access to a system we need firewalls. A firewall is a device (usually a router or a computer) installed btn the internal network of an organization and the rest of the Internet to isolate organization's internal net from larger Internet
- It is designed to forward some packets (allows them to pass) and block (i.e., filter or not forward) others



Firewalls: Why

Prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections

Prevent illegal modification/access of internal data.

- E.g., attacker replaces CIA's homepage with something else

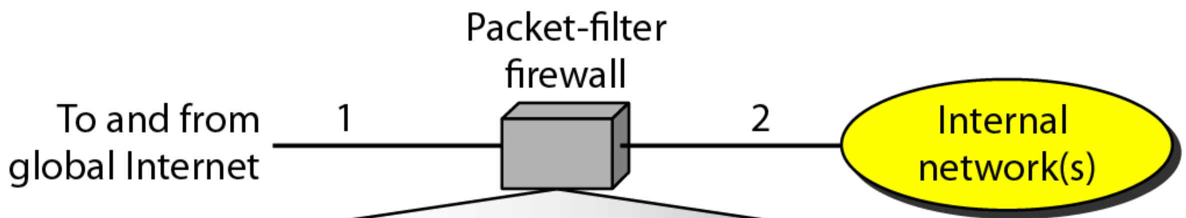
Allow only authorized access to inside network (set of authenticated users/hosts)

Firewall types:

- Packet-filter firewall
- Proxy (application layer) firewall (aka the application gateway

Packet-Filter Firewall

- A packet-filter firewall filters at the network or transport layer based on their header info (source and destination IP addresses and port addresses and type of protocol (TCP or UDP)



Interface	Source IP	Source port	Destination IP	Destination port
1	131.34.0.0	*	*	*
1	*	*	*	23
1	*	*	194.78.20.8	*
2	*	80	*	*

No web surfing at work dude ;)

Proxy Firewall

- Sometimes we need to filter a message based on the information available in the message itself (at the application layer)
- One solution is a proxy computer (sometimes called an application gateway)
- When the user client process sends a message, proxy firewall runs a server process to receive the request. The server opens the packet at the application layer and checks whether the request is legitimate. If it is, the server acts as a client process and sends the message to the real server in the corporation. If it is not, the message is dropped and an error message is sent to the external user. In this way, the requests of the external users are filtered based on the content at the application layer