

# Network Security

## A note on the use of these ppt slides:

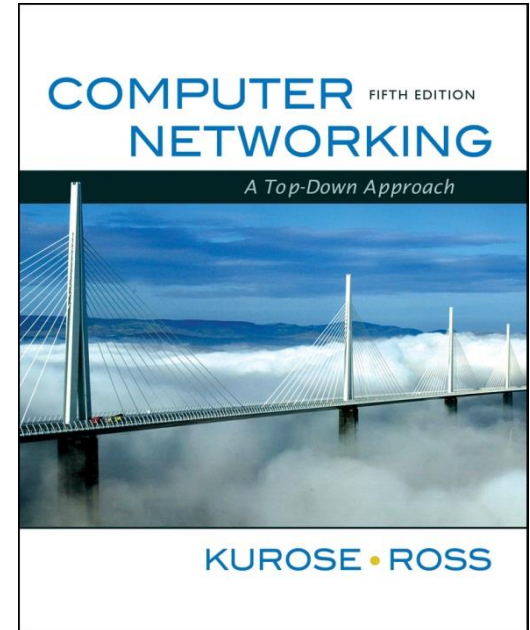
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- ☐ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- ☐ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2009

J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking:  
A Top Down Approach ,  
5<sup>th</sup> edition.*

*Jim Kurose, Keith Ross  
Addison-Wesley, April  
2009.*

# Network Security

## Chapter goals:

- ❑ understand principles of network security:
  - cryptography and its *many* uses beyond "confidentiality"
  - authentication
  - message integrity
- ❑ security in practice:
  - firewalls and intrusion detection systems
  - security in application, transport, network, link layers

# Roadmap

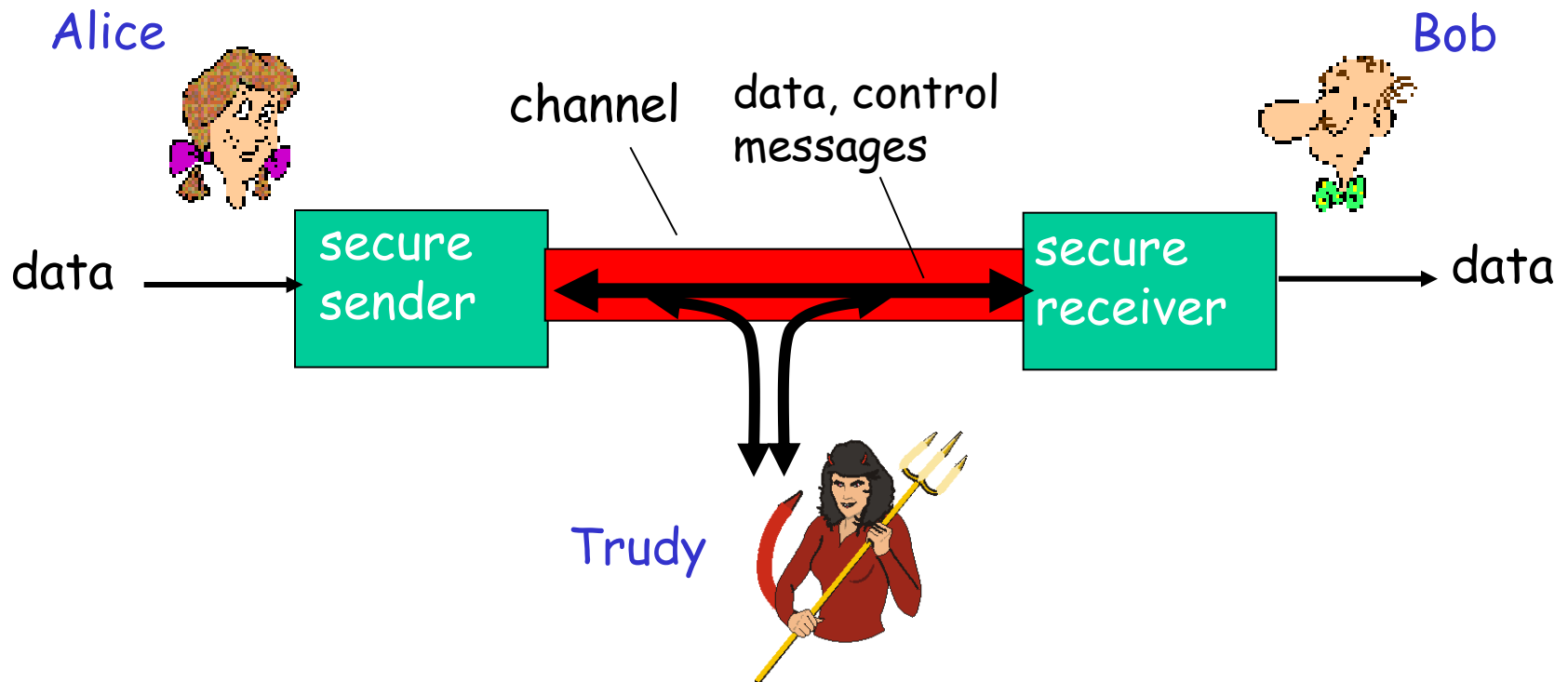
1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# What is network security?

- ❑ **Confidentiality:** only sender, intended receiver should “understand” message contents
  - sender encrypts message
  - receiver decrypts message
- ❑ **Authentication:** sender, receiver want to confirm identity of each other
- ❑ **Message integrity:** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- ❑ **Access and availability:** services must be accessible and available to users

# Friends and enemies: Alice, Bob, Trudy

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



# Who might Bob, 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
- ❑ other examples?

# There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: A lot! See section 1.6

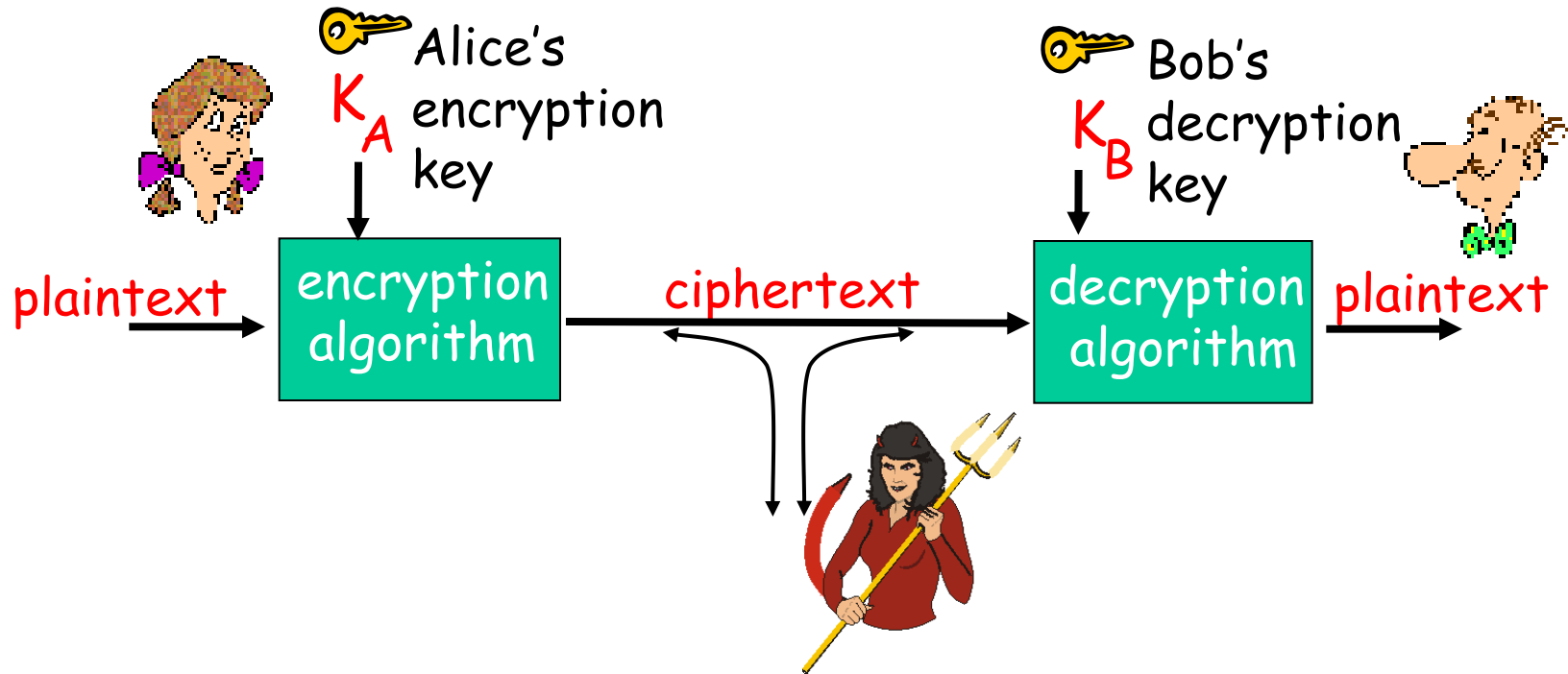
- *eavesdrop*: intercept messages
- actively *insert* messages into connection
- *impersonation*: can fake (spoof) source address in packet (or any field in packet)
- *hijacking*: “take over” ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent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

# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS



# The language of cryptography



$m$  plaintext message

$K_A(m)$  ciphertext, encrypted with key  $K_A$

$m = K_B(K_A(m))$

# Simple encryption scheme

**substitution cipher:** substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

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:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

E.g.: Plaintext: bob. i love you. alice  
ciphertext: nkn. s gktc wky. mgsbc

**Key:** the mapping from the set of 26 letters to the set of 26 letters

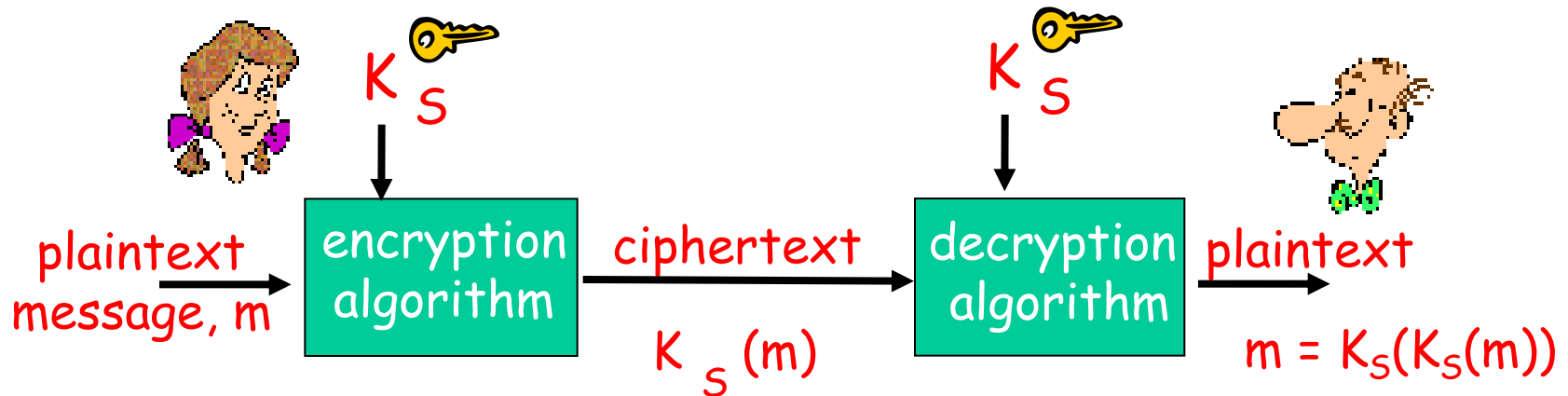
# Polyalphabetic encryption

- n monoalphabetic cyphers,  $M_1, M_2, \dots, M_n$
- Cycling pattern:
  - e.g.,  $n=4$ ,  $M_1, M_3, M_4, M_3, M_2$ ;  $M_1, M_3, M_4, M_3, M_2$ ;
- For each new plaintext symbol, use subsequent monoalphabetic pattern in cyclic pattern
  - dog: d from  $M_1$ , o from  $M_3$ , g from  $M_4$
- Key: the n ciphers and the cyclic pattern

# Types of Cryptography

- ❑ Crypto often uses keys:
  - Algorithm is known to everyone
  - Only "keys" are secret
- ❑ Public key cryptography
  - Involves the use of two keys
- ❑ Symmetric key cryptography
  - Involves the use one key
- ❑ Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

# Symmetric key cryptography



**symmetric key** crypto: Bob and Alice share same (symmetric) key:  $K_S$

□ e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

# Two types of symmetric ciphers

## ❑ Stream ciphers

- encrypt one bit at time

## ❑ Block ciphers

- Break plaintext message in equal-size blocks
- Encrypt each block as a unit

# Block ciphers

- ❑ Message to be encrypted is processed in blocks of  $k$  bits (e.g., 64-bit blocks).
- ❑ 1-to-1 mapping is used to map  $k$ -bit block of plaintext to  $k$ -bit block of ciphertext

## Example with $k=3$ :

<u>input</u>	<u>output</u>
000	110
001	111
010	101
011	100

<u>input</u>	<u>output</u>
100	011
101	010
110	000
111	001

What is the ciphertext for 010110001111 ?

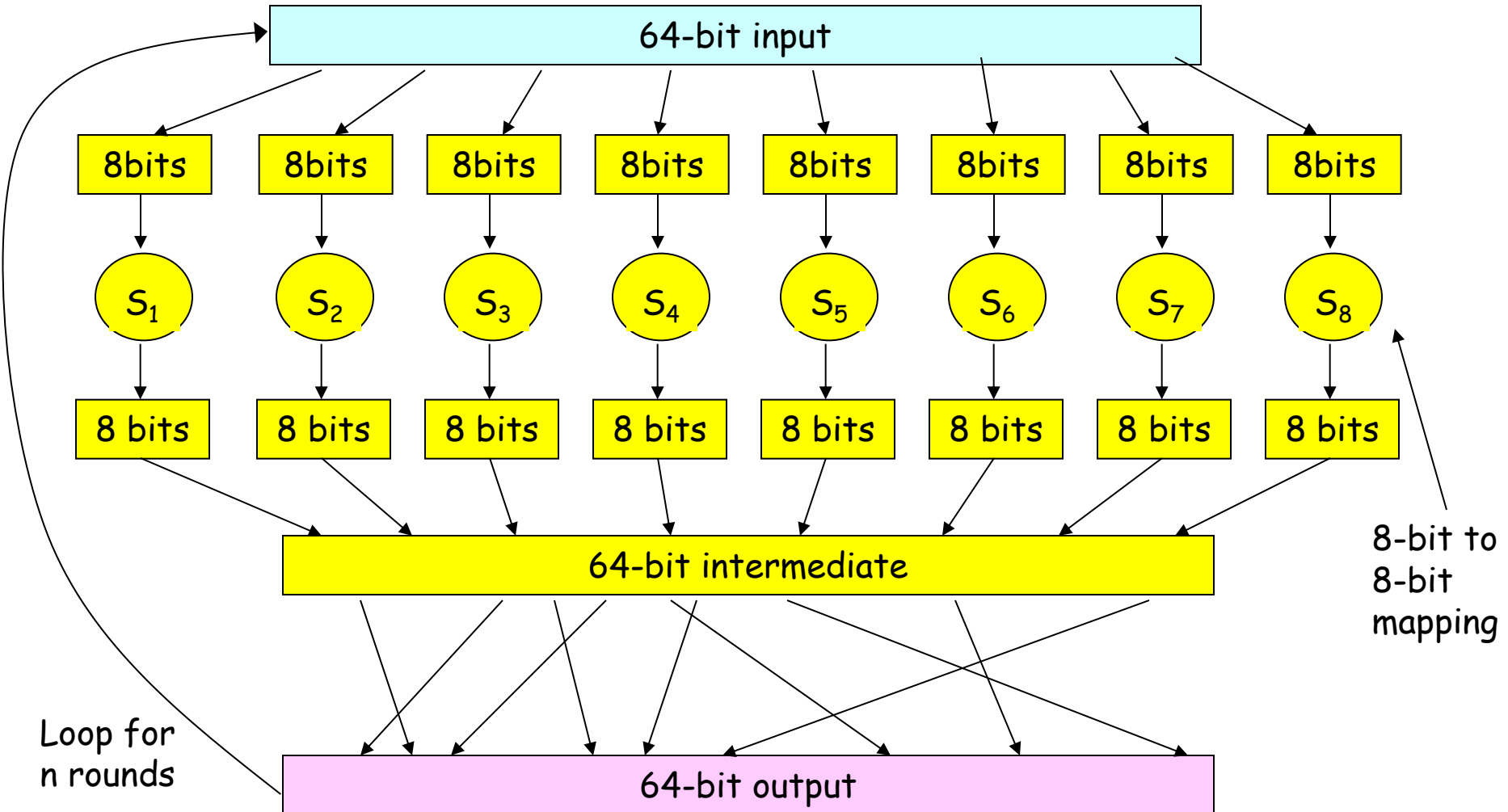
101000111001

# Block ciphers

- ❑ How many possible mappings are there for  $k=3$ ?
  - How many 3-bit inputs?
  - How many permutations of the 3-bit inputs?
  - Answer: 40,320 ; not very many!
- ❑ In general,  $2^k!$  mappings; huge for  $k=64$
- ❑ Problem:
  - Table approach requires table with  $2^{64}$  entries, each entry with 64 bits
- ❑ Table too big: instead use function that simulates a randomly permuted table

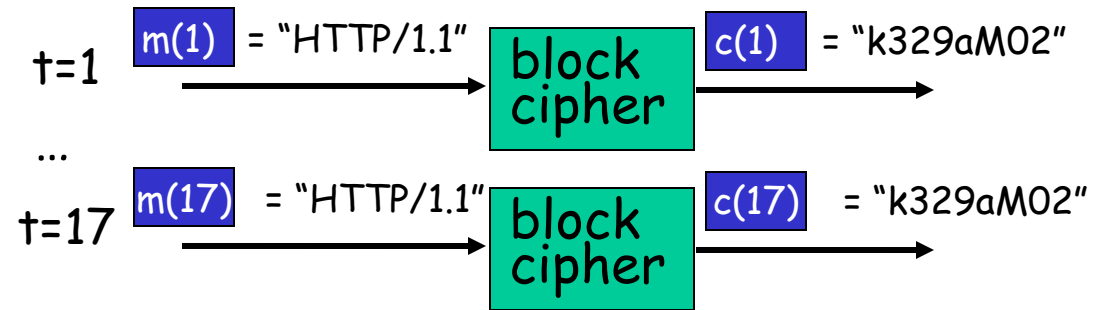


# Prototype function

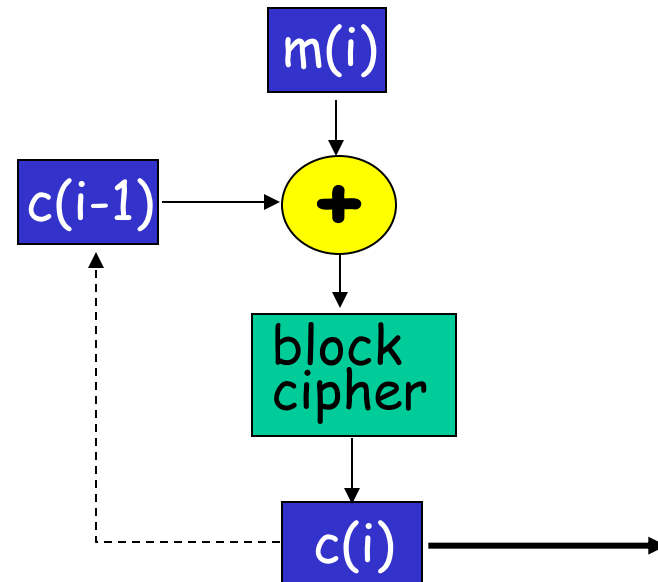


# Cipher Block Chaining (CBC)

- ❑ cipher block: if input block repeated, will produce same cipher text:



- ❑ *cipher block chaining:*  
XOR ith input block,  $m(i)$ , with previous block of cipher text,  $c(i-1)$ 
  - $c(0)$  transmitted to receiver in clear
  - what happens in "HTTP/1.1" scenario from above?



# Symmetric key crypto: DES

## DES: Data Encryption Standard

- ❑ US encryption standard [NIST 1993]
- ❑ 56-bit symmetric key, 64-bit plaintext input
- ❑ Block cipher with cipher block chaining
- ❑ How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - No known good analytic attack
- ❑ making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys (actually encrypt, decrypt, encrypt)

# AES: Advanced Encryption Standard

- ❑ new (Nov. 2001) symmetric-key NIST standard, replacing DES
- ❑ processes data in 128 bit blocks
- ❑ 128, 192, or 256 bit keys
- ❑ brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

# Public Key Cryptography

## symmetric key crypto

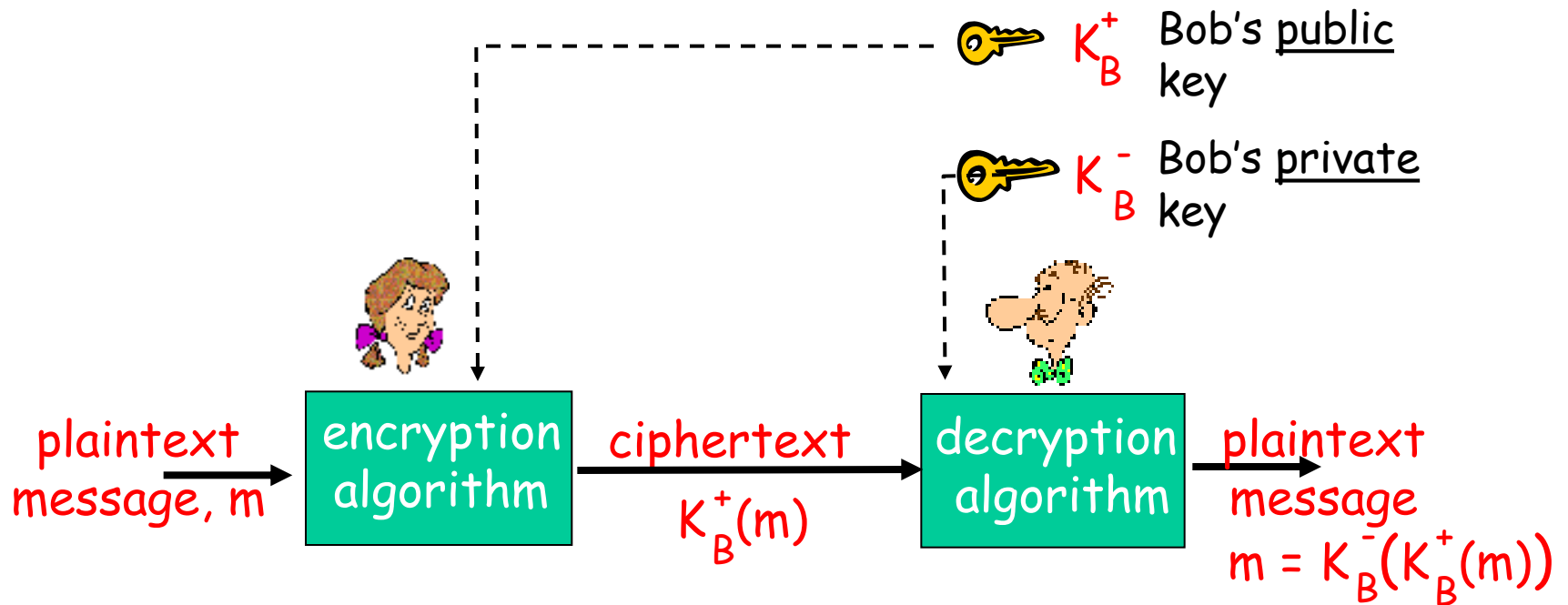
- ❑ requires sender, receiver know shared secret key
- ❑ Q: how to agree on key in first place (particularly if never "met")?

## public key cryptography

- ❑ radically different approach [Diffie-Hellman76, RSA78]
- ❑ sender, receiver do *not* share secret key
- ❑ *public* encryption key known to *all*
- ❑ *private* decryption key known only to receiver



# Public key cryptography



# Public key encryption algorithms

Requirements:

① need  $K_B^+(\cdot)$  and  $K_B^-(\cdot)$  such that

$$K_B^-(K_B^+(m)) = m$$

② given public key  $K_B^+$ , it should be impossible to compute private key  $K_B^-$

**RSA:** Rivest, Shamir, Adelson algorithm

# Prerequisite: modular arithmetic

□  $x \bmod n$  = remainder of  $x$  when divide by  $n$

□ Facts:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$$

□ Thus

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

□ Example:  $x=14$ ,  $n=10$ ,  $d=2$ :

$$(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$$

$$x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$$



# RSA: getting ready

- ❑ A message is a bit pattern.
- ❑ A bit pattern can be uniquely represented by an integer number.
- ❑ Thus encrypting a message is equivalent to encrypting a number.

## Example

- ❑  $m = 10010001$ . This message is uniquely represented by the decimal number 145.
- ❑ To encrypt  $m$ , we encrypt the corresponding number, which gives a new number (the cyphertext).

# RSA: Creating public/private key pair

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:  $ed \bmod z = 1$ ).
5. Public key is  $(n, e)$ . Private key is  $(n, d)$ .  

$\underbrace{\hspace{1.5cm}}_{K_B^+}$

$\underbrace{\hspace{1.5cm}}_{K_B^-}$

# RSA: Encryption, decryption

0. Given  $(n,e)$  and  $(n,d)$  as computed above
1. To encrypt message  $m (<n)$ , compute
$$c = m^e \bmod n$$
2. To decrypt received bit pattern,  $c$ , compute
$$m = c^d \bmod n$$

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  $z$ ).

Encrypting 8-bit messages.

encrypt:	<u>bit pattern</u>	<u><math>m</math></u>	<u><math>m^e</math></u>	<u><math>c = m^e \bmod n</math></u>
	00001000	12	24832	17

decrypt:	<u><math>c</math></u>	<u><math>c^d</math></u>	<u><math>m = c^d \bmod n</math></u>
	17	481968572106750915091411825223071697	12

# Why does RSA work?

- ❑ Must show that  $c^d \bmod n = m$   
where  $c = m^e \bmod n$
- ❑ Fact: for any  $x$  and  $y$ :  $x^y \bmod n = x^{(y \bmod z)} \bmod n$ 
  - where  $n = pq$  and  $z = (p-1)(q-1)$
- ❑ Thus,
$$\begin{aligned}c^d \bmod n &= (m^e \bmod n)^d \bmod n \\&= m^{ed} \bmod n \\&= m^{(ed \bmod z)} \bmod n \\&= m^1 \bmod n \\&= m\end{aligned}$$

# RSA: another important property

The following property will be *very* useful later:

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

use public key  
first, followed  
by private key

use private key  
first, followed  
by public key

*Result is the same!*

Why  $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$  ?

Follows directly from modular arithmetic:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

# Why is RSA Secure?

- ❑ Suppose you know Bob's public key  $(n,e)$ . How hard is it to determine  $d$ ?
- ❑ Essentially need to find factors of  $n$  without knowing the two factors  $p$  and  $q$ .
- ❑ Fact: factoring a big number is hard.

# Generating RSA keys

- ❑ Have to find big primes  $p$  and  $q$
- ❑ Approach: make good guess then apply testing rules (see Kaufman)



# Session keys

- ❑ Exponentiation is computationally intensive
- ❑ DES is at least 100 times faster than RSA

## Session key, $K_S$

- ❑ Bob and Alice use RSA to exchange a symmetric key  $K_S$
- ❑ Once both have  $K_S$ , they use symmetric key cryptography

# Roadmap

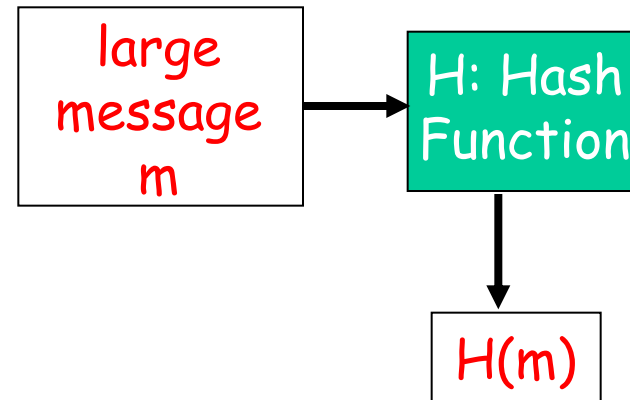
1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# Message Integrity

- ❑ Allows communicating parties to verify that received messages are authentic.
  - Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been replayed
  - Sequence of messages is maintained
- ❑ Let's first talk about message digests

# Message Digests

- ❑ Function  $H()$  that takes as input an arbitrary length message and outputs a fixed-length string:  
"message signature"
- ❑ Note that  $H()$  is a many-to-1 function
- ❑  $H()$  is often called a "hash function"



- ❑ Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine  $m$  from  $H(m)$
  - Collision resistance: Computationally difficult to produce  $m$  and  $m'$  such that  $H(m) = H(m')$
  - Seemingly random output

# Internet checksum: poor message digest

Internet checksum has some properties of hash function:

- ✓ produces fixed length digest (16-bit sum) of input
- ✓ is many-to-one
- ❑ But given message with given hash value, it is easy to find another message with same hash value.
- ❑ Example: Simplified checksum: add 4-byte chunks at a time:

<u>message</u>	<u>ASCII format</u>
----------------	---------------------

I O U 1	49 4F 55 31
---------	-------------

0 0 . 9	30 30 2E 39
---------	-------------

9 B O B	39 42 D2 42
---------	-------------

B2 C1 D2 AC
-------------

<u>message</u>	<u>ASCII format</u>
----------------	---------------------

I O U <u>9</u>	49 4F 55 <u>39</u>
----------------	--------------------

0 0 . <u>1</u>	30 30 2E <u>31</u>
----------------	--------------------

9 B O B	39 42 D2 42
---------	-------------

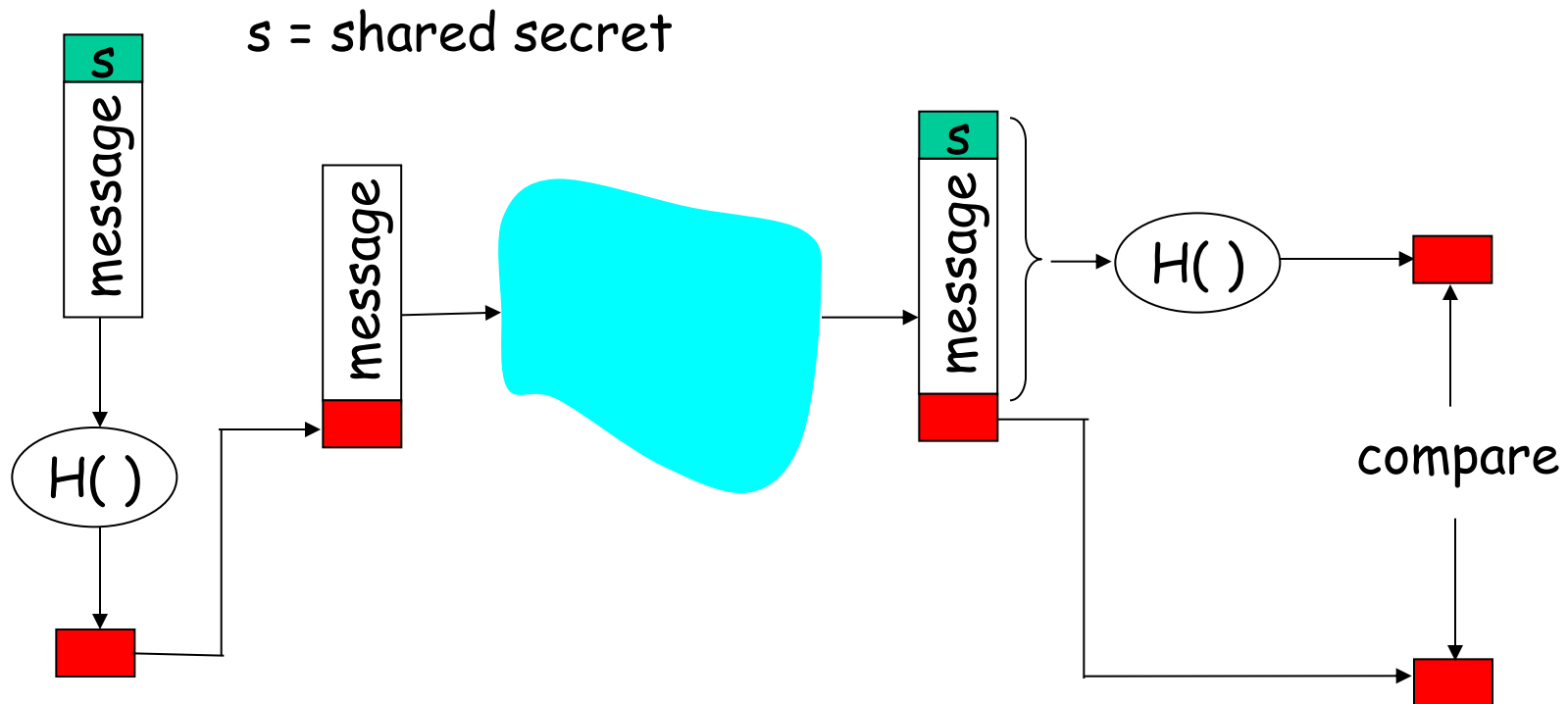
B2 C1 D2 AC
-------------

different messages  
but identical checksums!

# Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

# Message Authentication Code (MAC)



- ❑ *Authenticates sender*
- ❑ *Verifies message integrity*
- ❑ No encryption !
- ❑ Also called "keyed hash"
- ❑ Notation:  $MD_m = H(s || m)$  ; send  $m || MD_m$

# Digital Signatures

Cryptographic technique analogous to handwritten signatures.

- ❑ sender (Bob) digitally signs document, establishing he is document owner/creator.
- ❑ Goal is similar to that of a MAC, except now use public-key cryptography
- ❑ **verifiable, nonforgeable**: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document




# Digital Signatures

## Simple digital signature for message $m$ :

- Bob signs  $m$  by encrypting with his private key  $K_B^-$ , creating "signed" message,  $K_B^-(m)$

Bob's message,  $m$

Dear Alice  
Oh, how I have missed  
you. I think of you all the  
time! ... (blah blah blah)  
Bob

  $K_B^-$  Bob's private  
key

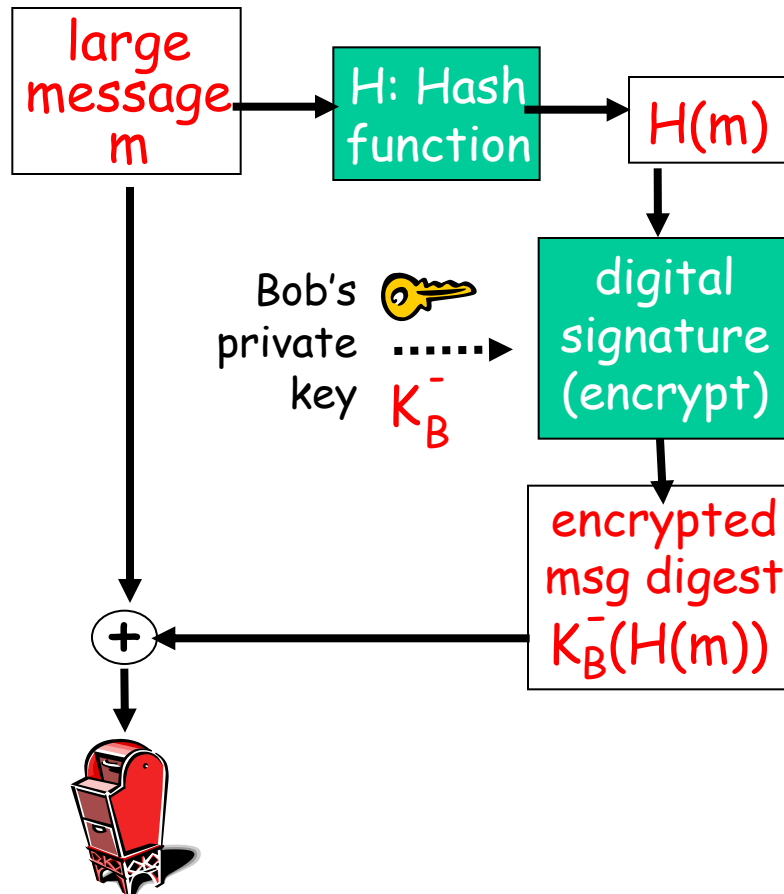
Public key  
encryption  
algorithm

$K_B^-(m)$

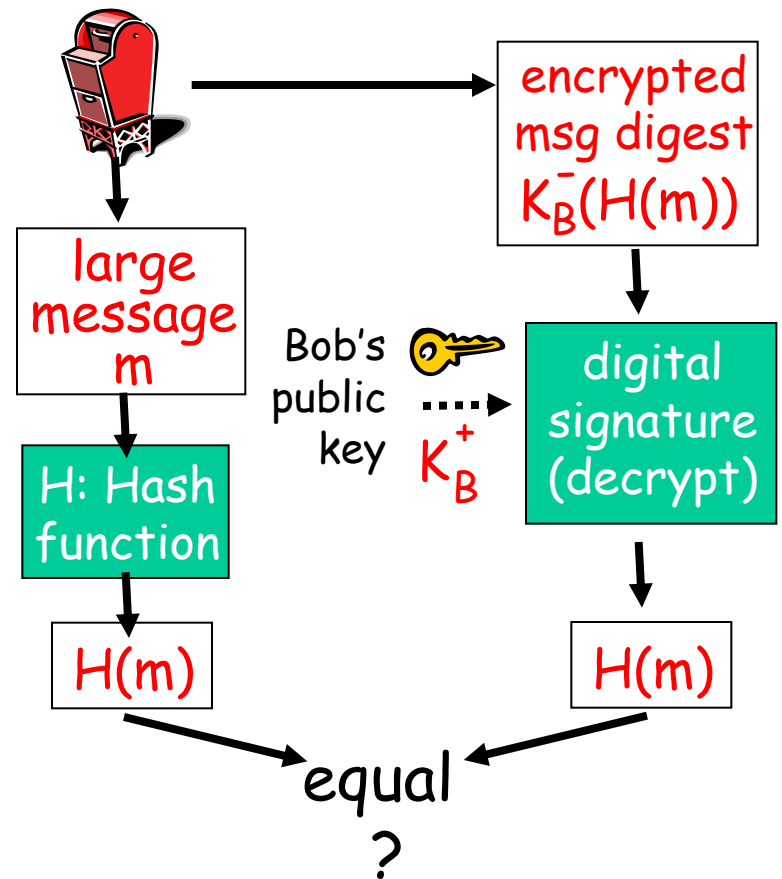
Bob's message,  
 $m$ , signed  
(encrypted) with  
his private key

# Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



# Digital Signatures (more)

- Suppose Alice receives msg  $m$ , digital signature  $K_B^-(m)$
- Alice verifies  $m$  signed by Bob by applying Bob's public key  $K_B^+$  to  $K_B^-(m)$  then checks  $K_B^+(K_B^-(m)) = m$ .
- If  $K_B^+(K_B^-(m)) = m$ , whoever signed  $m$  must have used Bob's private key.

Alice thus verifies that:

- ✓ Bob signed  $m$ .
- ✓ No one else signed  $m$ .
- ✓ Bob signed  $m$  and not  $m'$ .

Non-repudiation:

- ✓ Alice can take  $m$ , and signature  $K_B^-(m)$  to court and prove that Bob signed  $m$ .

# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: Ipsec
8. Operational security: firewalls and IDS

# Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”



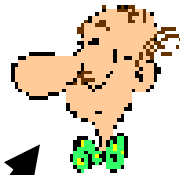
Failure scenario??



# Authentication

Goal: Bob wants Alice to “prove” her identity to him

Protocol ap1.0: Alice says “I am Alice”

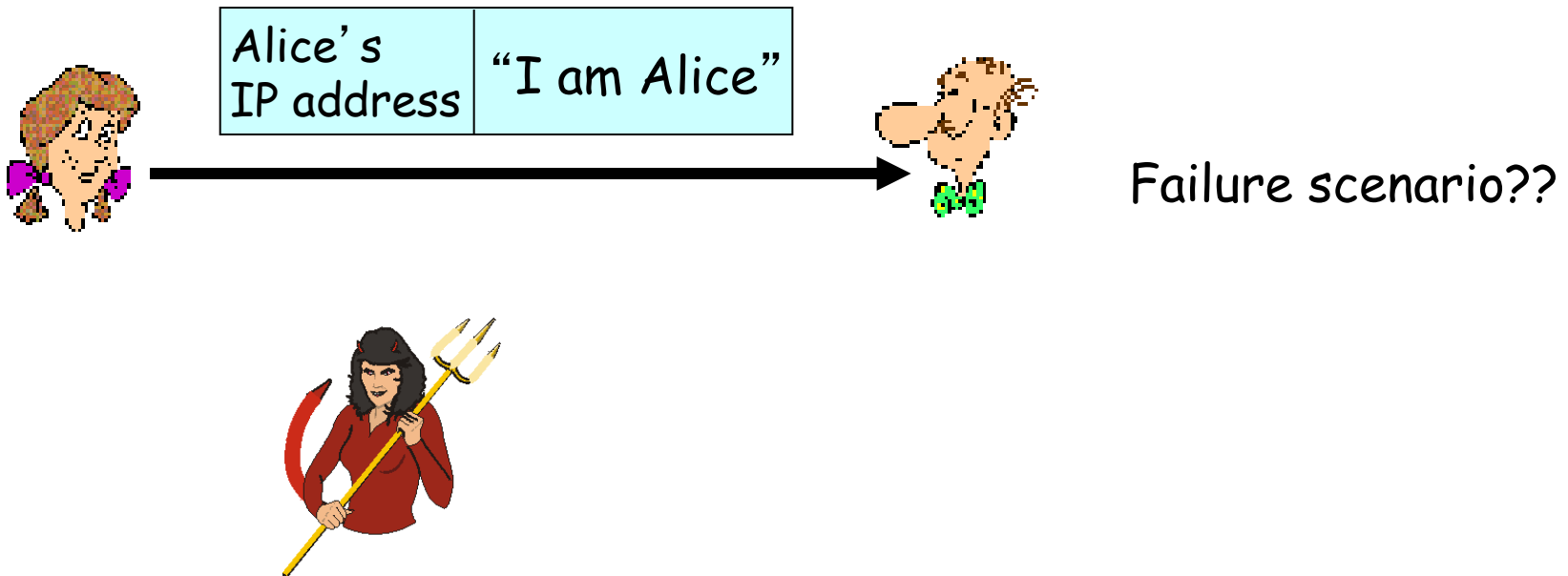


“I am Alice”

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

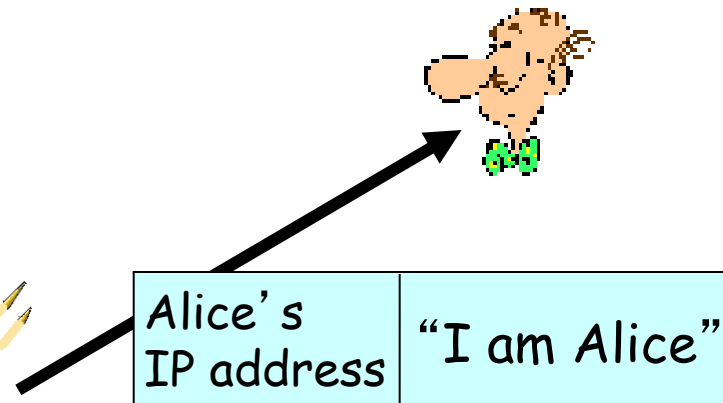
# Authentication: another try

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



# Authentication: another try

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

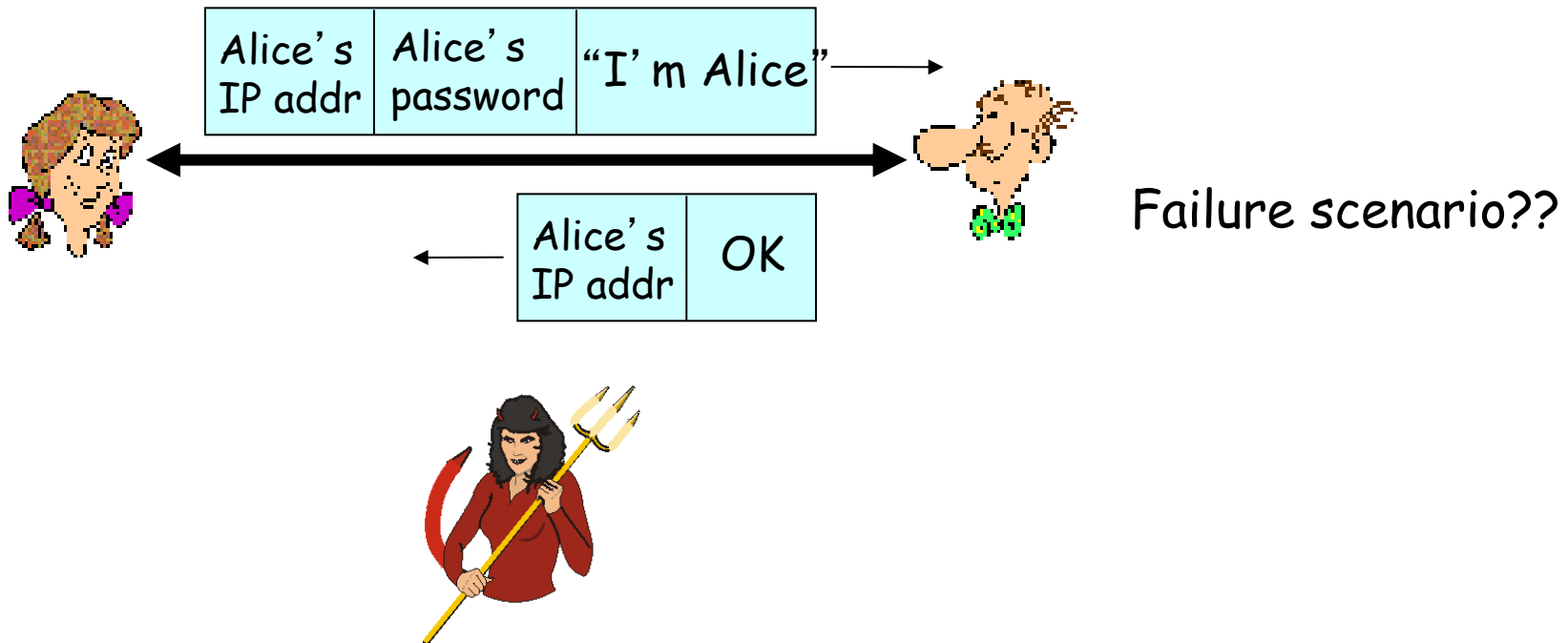


Trudy can create a packet “spoofing” Alice's address



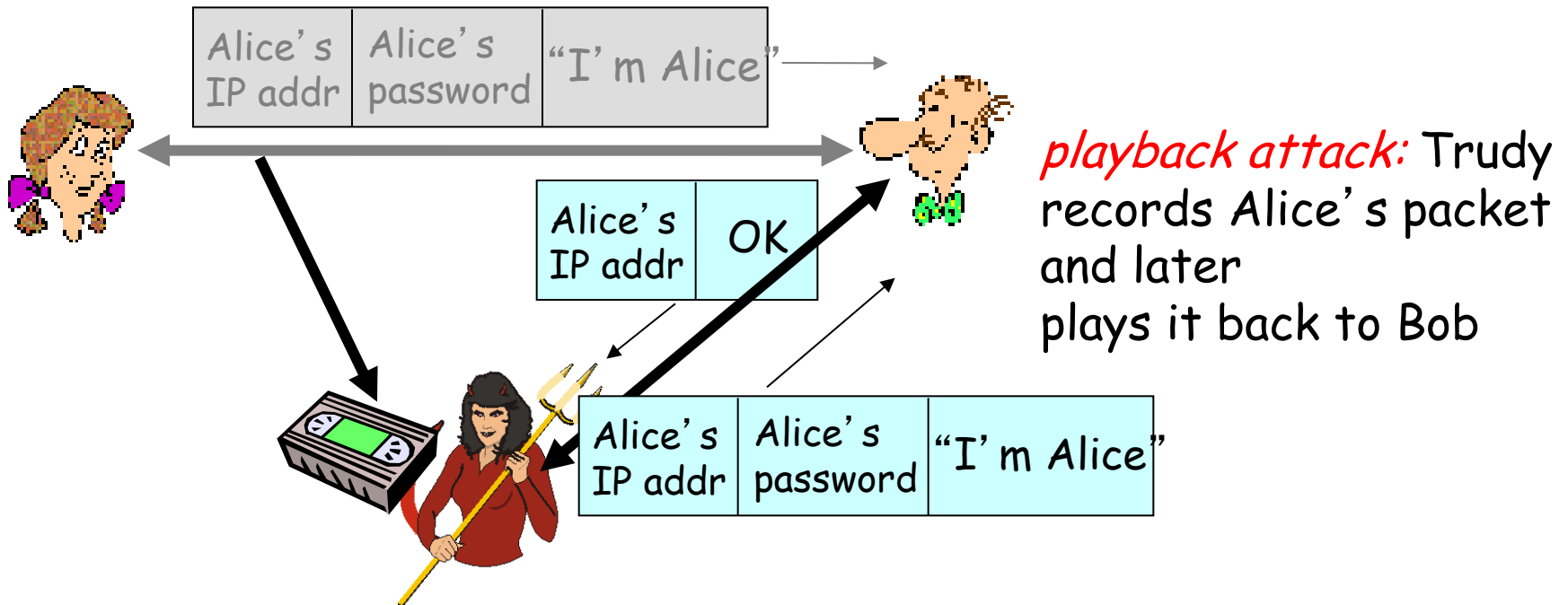
# Authentication: another try

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



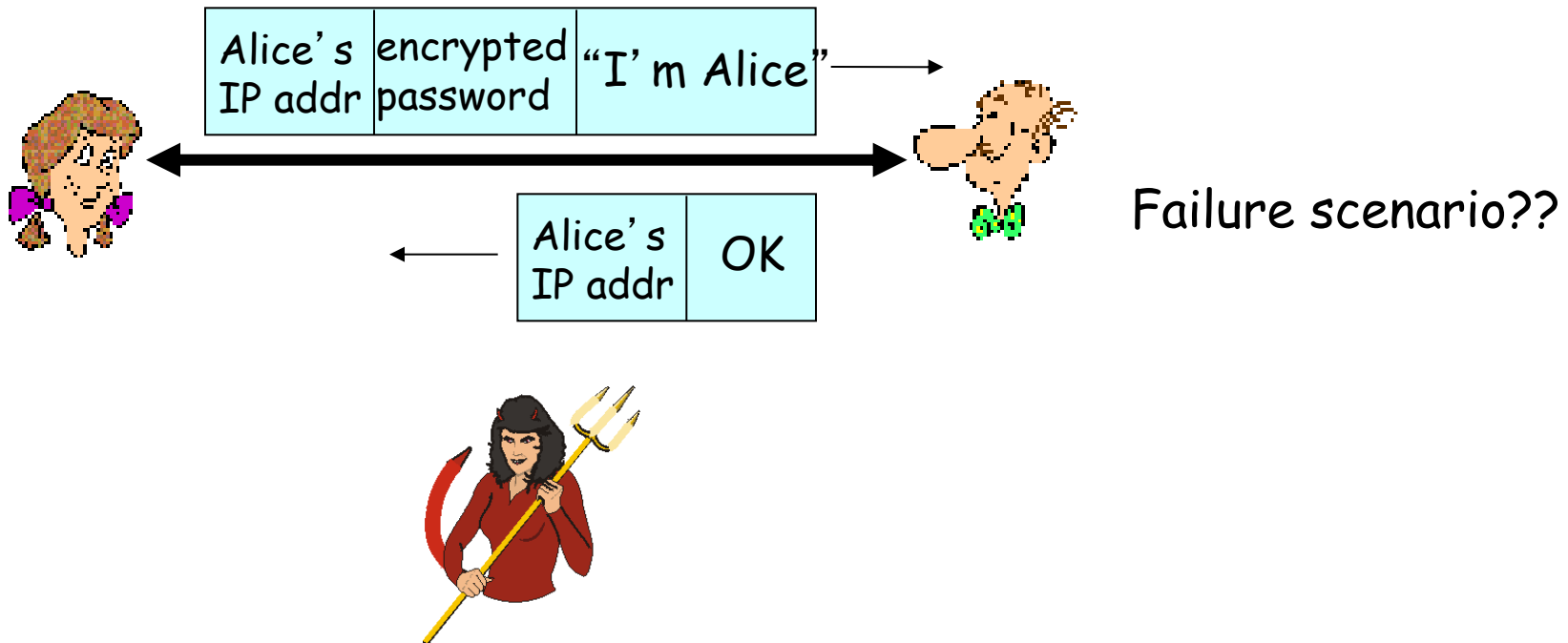
# Authentication: another try

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



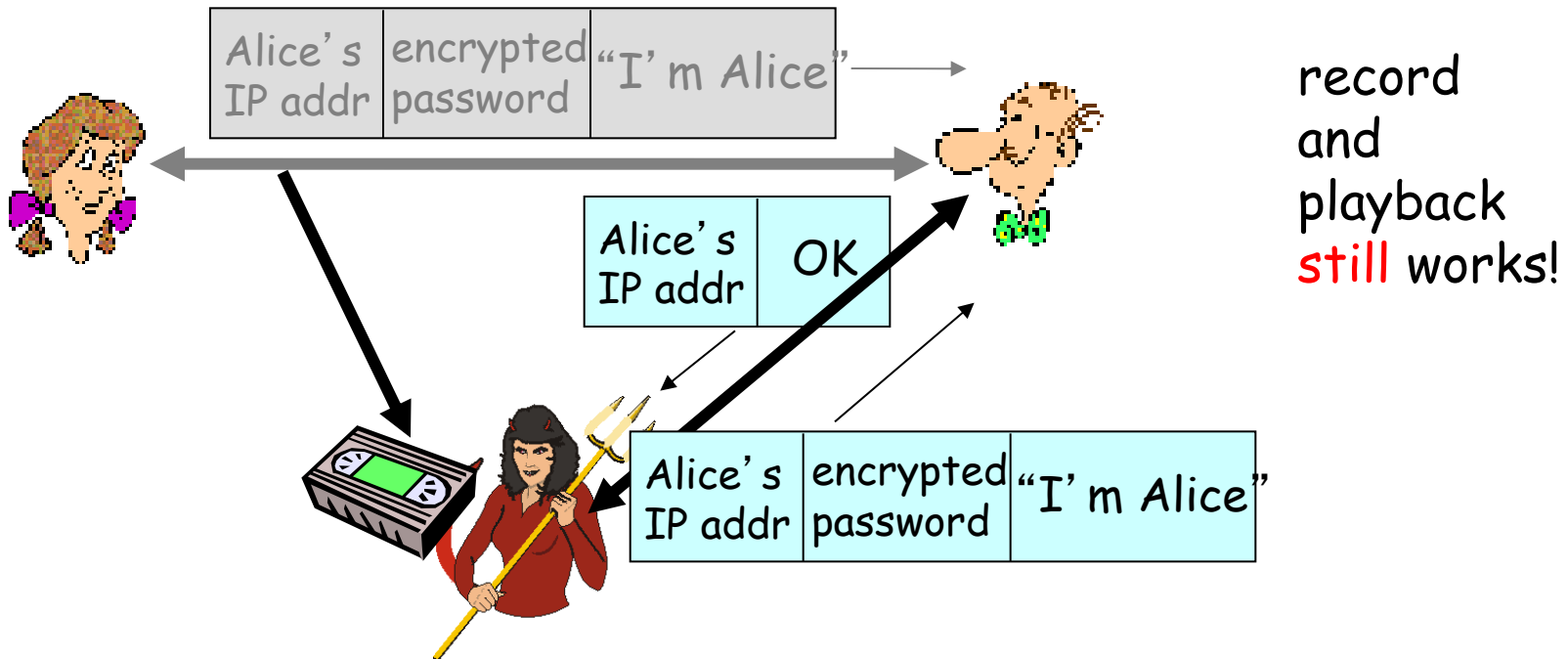
# Authentication: yet another try

Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.



# Authentication: another try

Protocol ap3.1: Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.



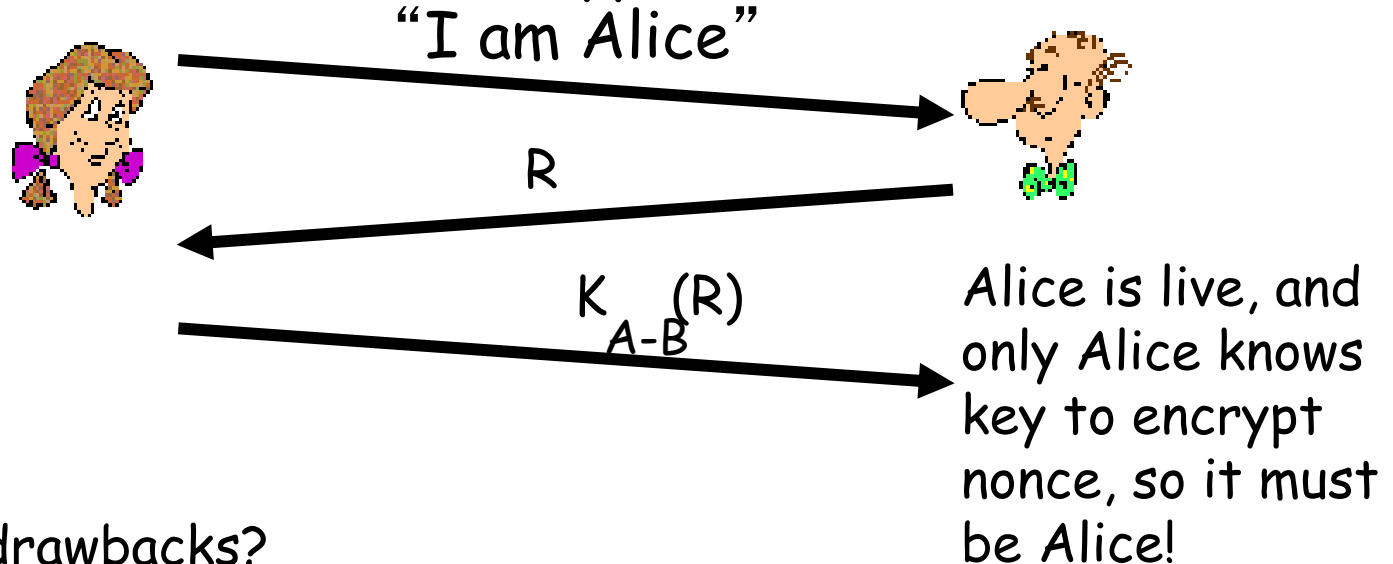
# Authentication: yet another try

Goal: avoid playback attack

Nonce: number (R) used only *once -in-a-lifetime*

ap4.0: to prove Alice “live”, Bob sends Alice **nonce**, R. Alice

must return R, encrypted with shared secret key



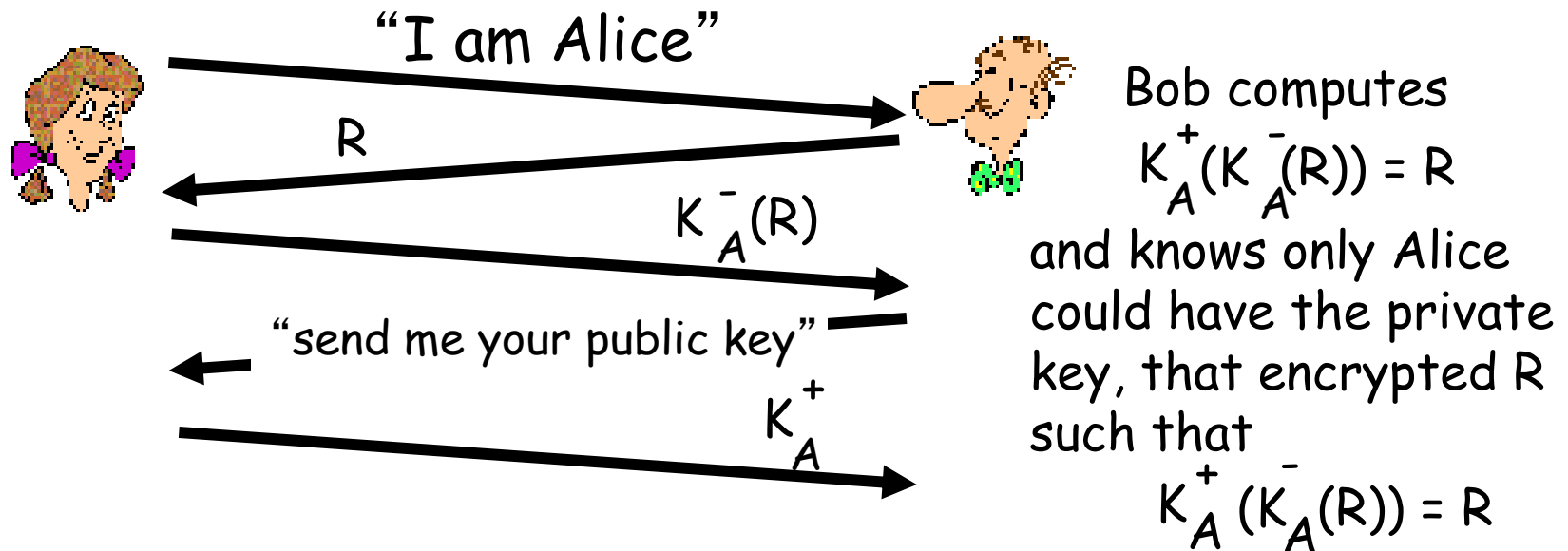
Failures, drawbacks?

# Authentication: ap5.0

ap4.0 requires shared symmetric key

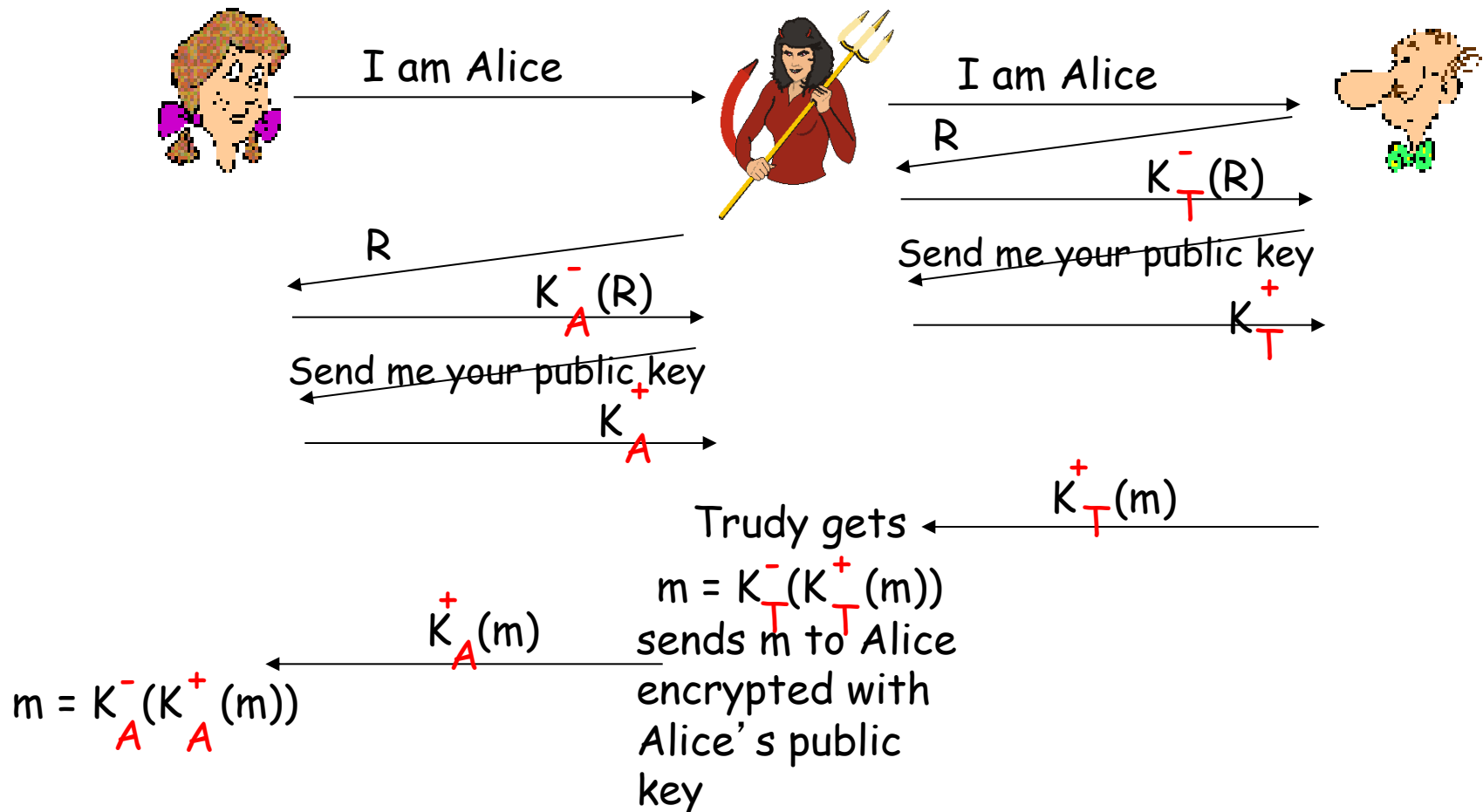
□ can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



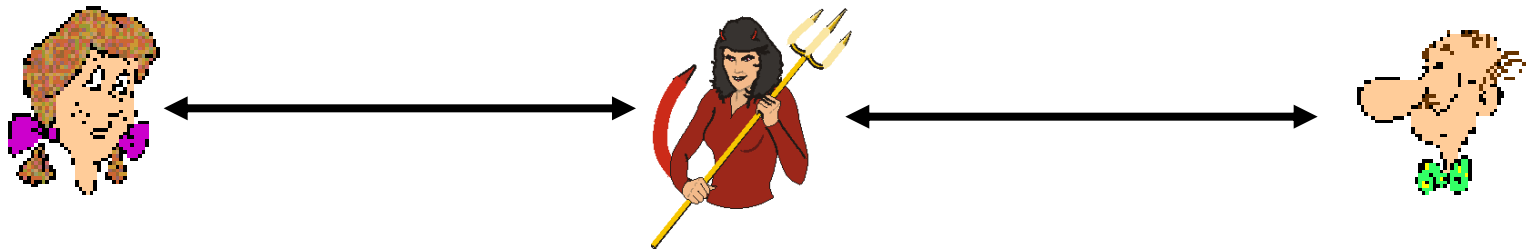
# ap5.0: security hole

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



## ap5.0: security hole

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



Difficult to detect:

- ❑ Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
- ❑ problem is that Trudy receives all messages as well!

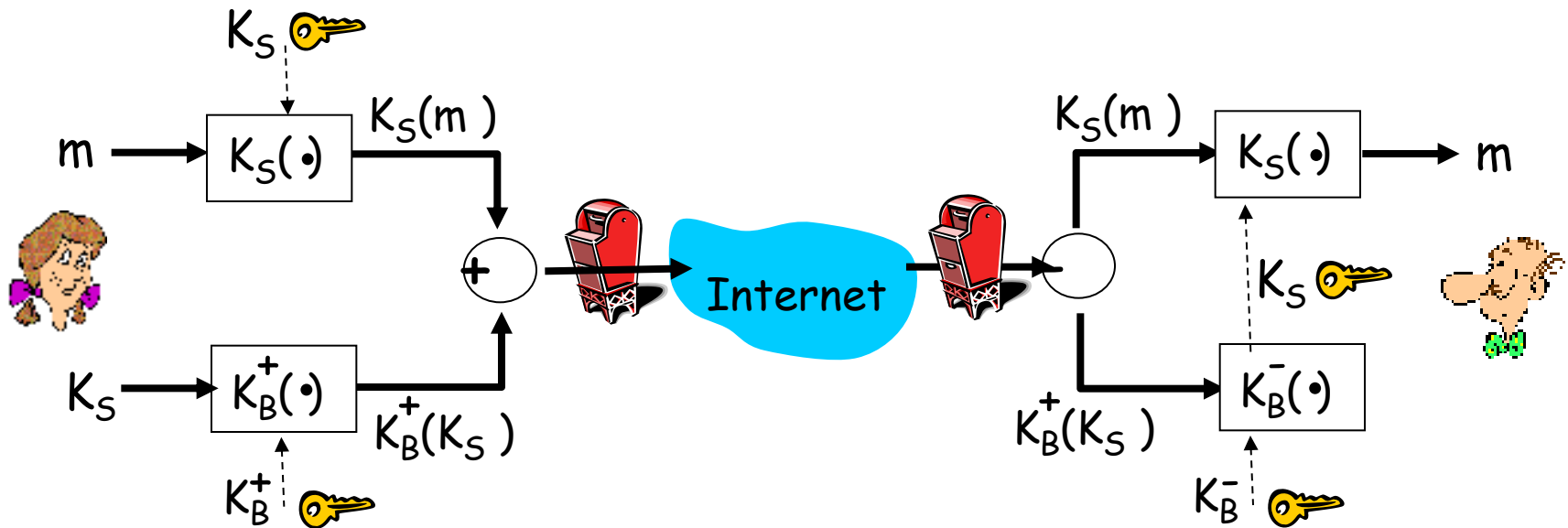


# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# Secure e-mail

- Alice wants to send confidential e-mail,  $m$ , to Bob.

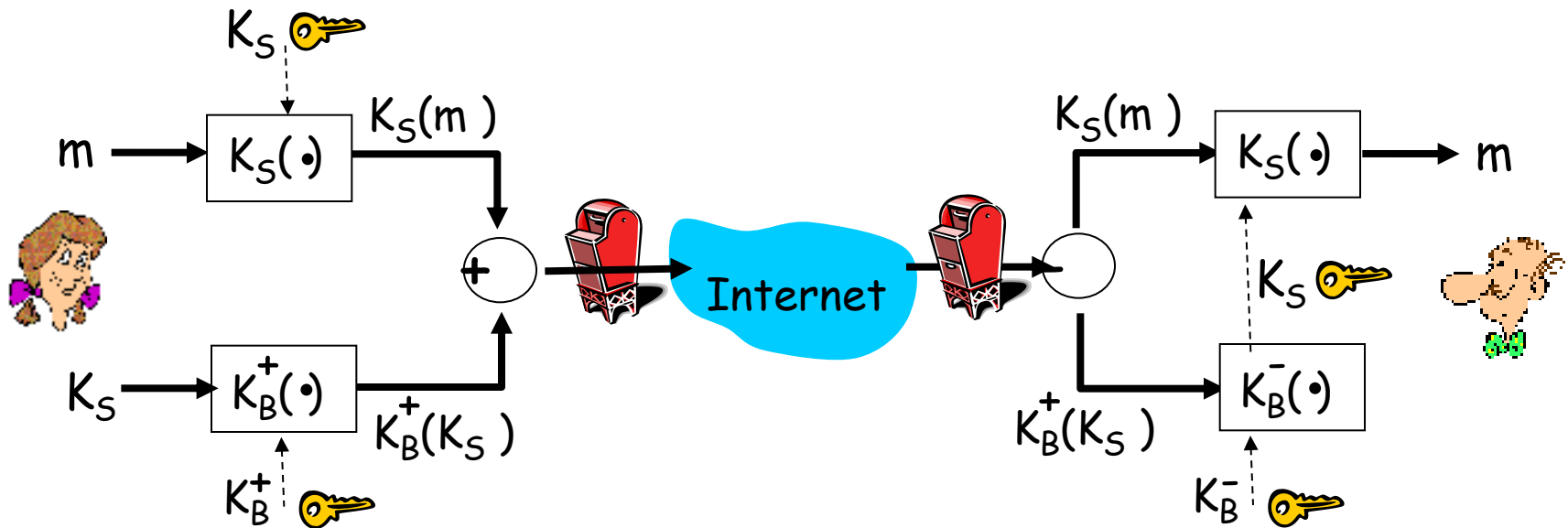


## Alice:

- generates random *symmetric* private key,  $K_S$ .
- encrypts message with  $K_S$  (for efficiency)
- also encrypts  $K_S$  with Bob's public key.
- sends both  $K_S(m)$  and  $K_B(K_S)$  to Bob.

# Secure e-mail

- Alice wants to send confidential e-mail,  $m$ , to Bob.

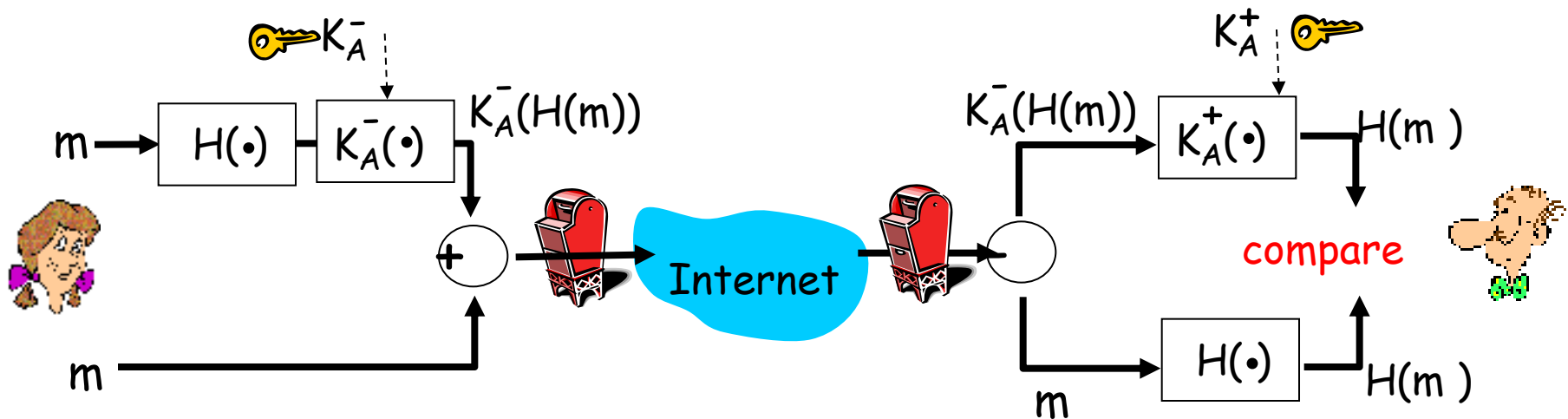


**Bob:**

- uses his private key to decrypt and recover  $K_S$
- uses  $K_S$  to decrypt  $K_S(m)$  to recover  $m$

# Secure e-mail (continued)

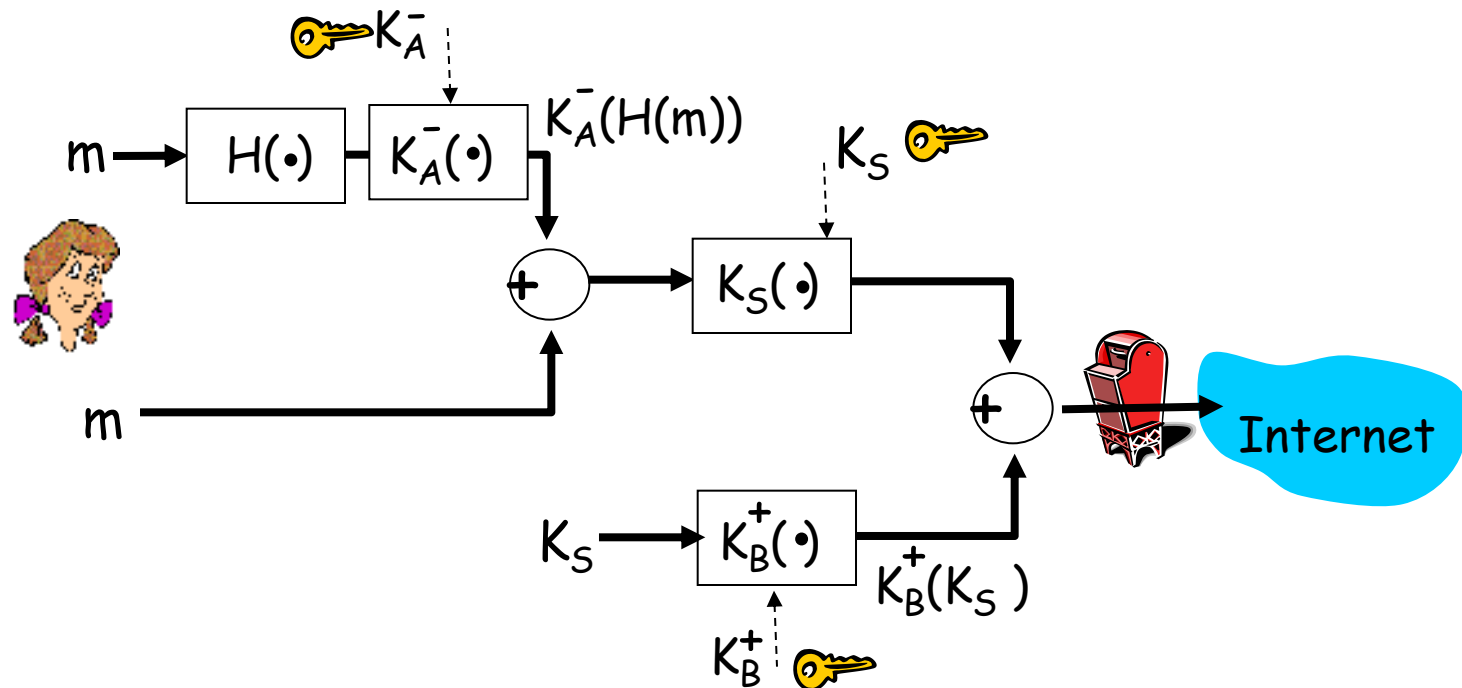
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

# Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



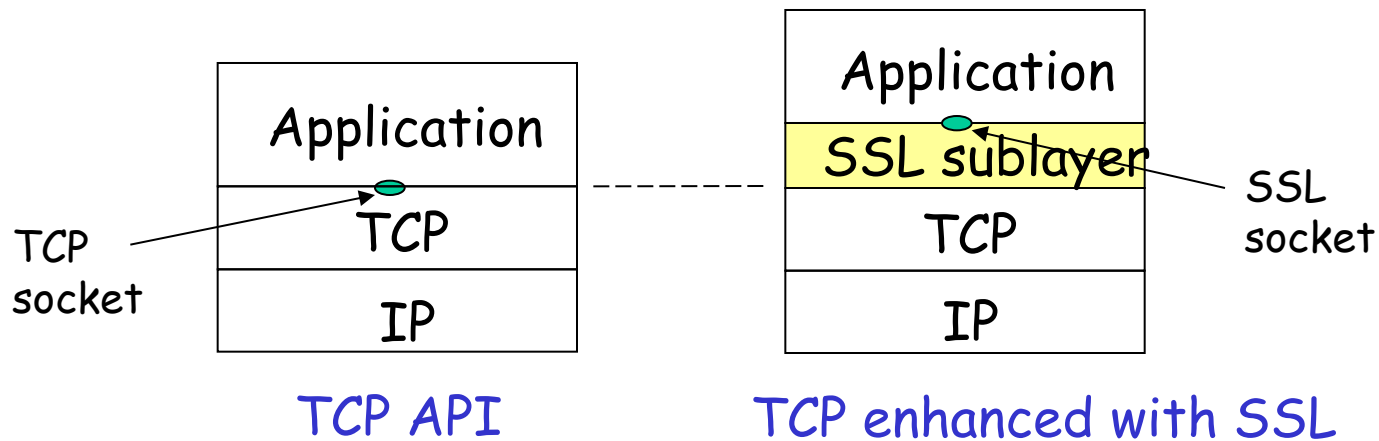
**Alice uses three keys:** her private key, Bob's public key, newly created symmetric key

# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# Secure sockets layer (SSL)

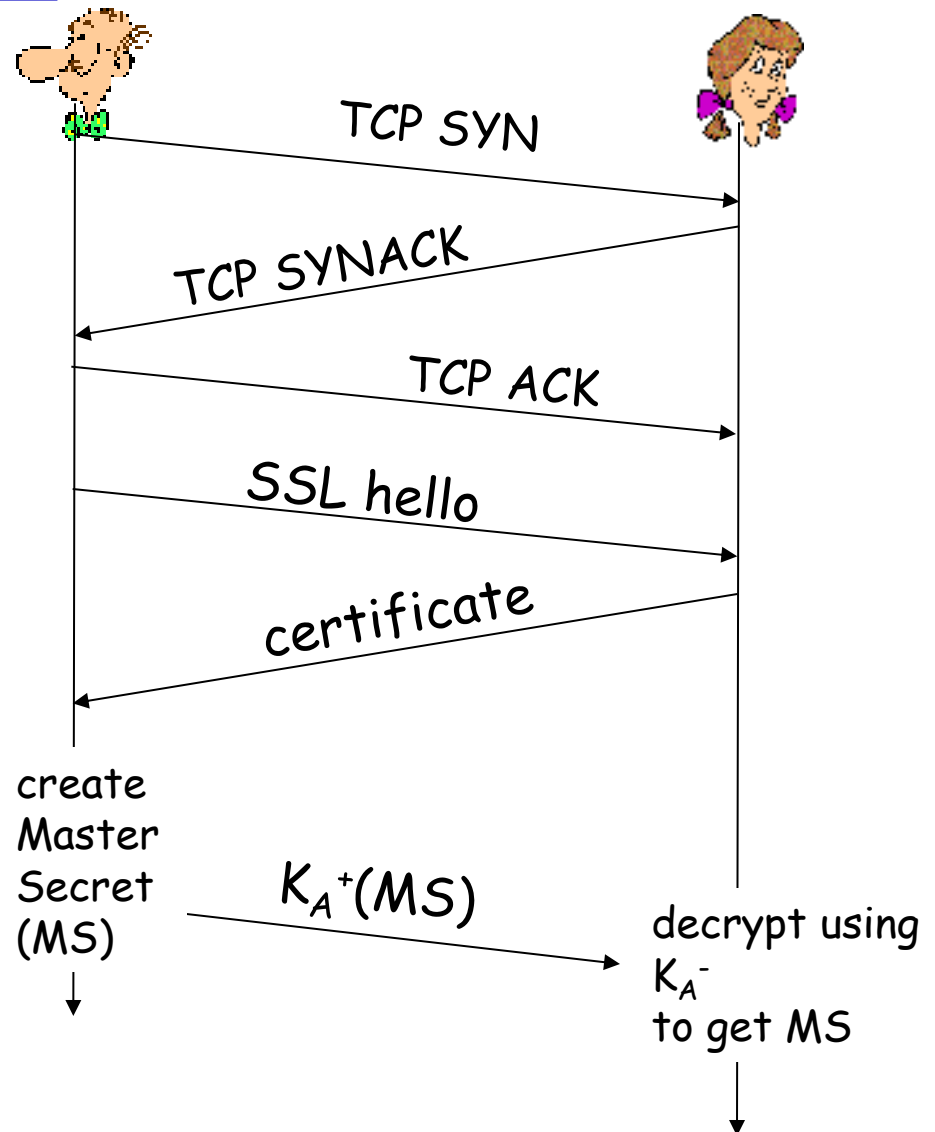
- ❑ provides transport layer security to any TCP-based application using SSL services.
  - e.g., between Web browsers, servers for e-commerce (shttp)
- ❑ security services:
  - server authentication, data encryption, client authentication (optional)



# SSL: three phases

## 1. Handshake:

- ❑ Bob establishes TCP connection to Alice
- ❑ authenticates Alice via CA signed certificate
- ❑ creates, encrypts (using Alice's public key), sends master secret key to Alice
  - nonce exchange not shown





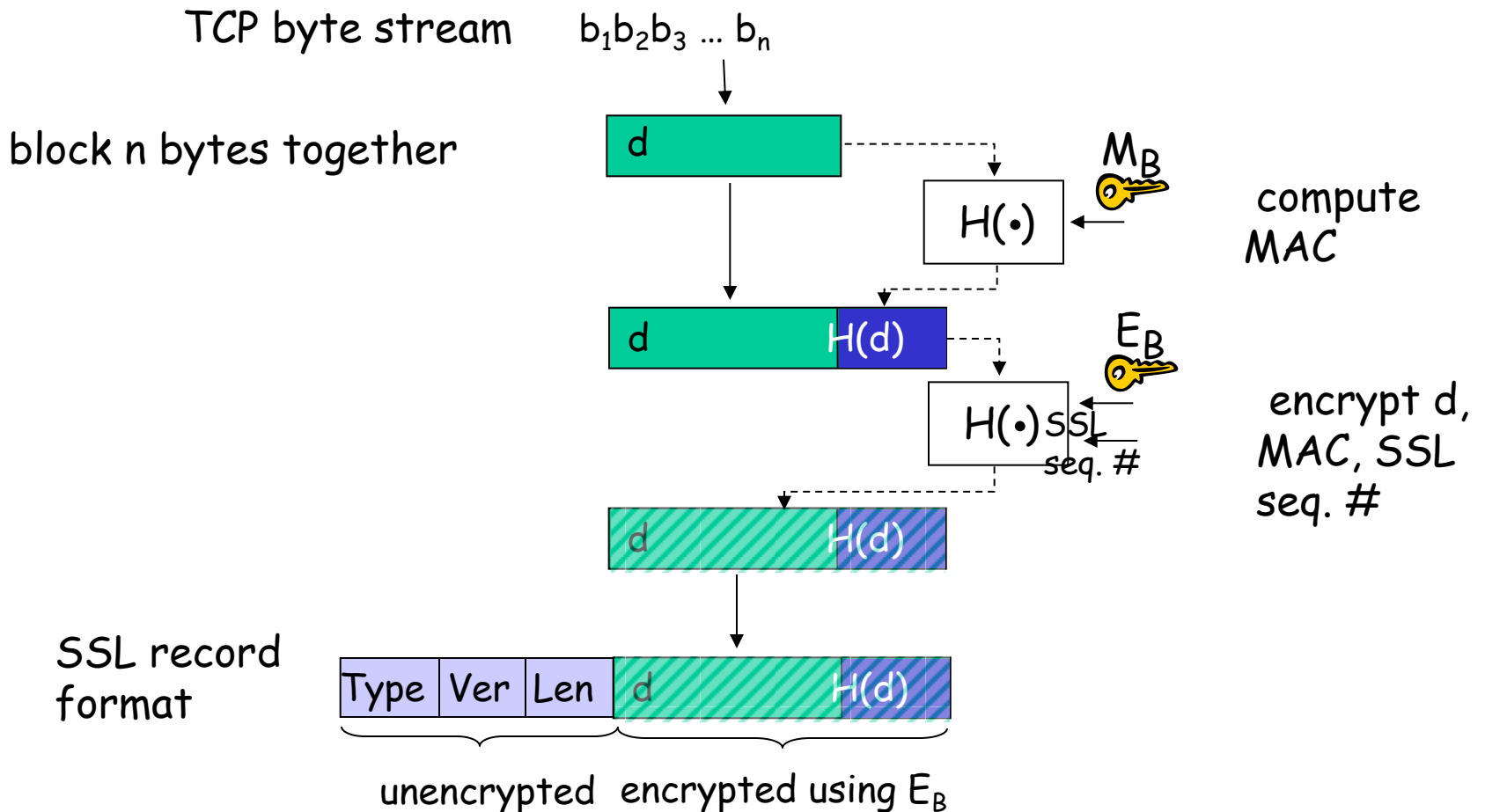
# SSL: three phases

## *2. Key Derivation:*

- ❑ Alice, Bob use shared secret (MS) to generate 4 keys:
  - $E_B$ : Bob→Alice data encryption key
  - $E_A$ : Alice→Bob data encryption key
  - $M_B$ : Bob→Alice MAC key
  - $M_A$ : Alice→Bob MAC key
- ❑ encryption and MAC algorithms negotiable between Bob, Alice
- ❑ why 4 keys?
  - ❑ 2 encryption keys will be used to encrypt data
  - ❑ 2 MAC keys will be used to verify the integrity of the data

# SSL: three phases

## 3. Data transfer



# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# IPsec: Network Layer Security

- ❑ network-layer secrecy:
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- ❑ network-layer authentication
  - destination host can authenticate source IP address (prevent IP address spoofing)
- ❑ two principal protocols:
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- ❑ for both AH and ESP, source, destination handshake:
  - create network-layer logical channel called a security association (SA)
- ❑ each SA unidirectional.
- ❑ uniquely determined by:
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID

# Authentication Header (AH) Protocol

- ❑ provides source authentication, data integrity, no confidentiality
- ❑ AH header inserted between IP header, data field.
- ❑ protocol field: 51
- ❑ intermediate routers process datagrams as usual

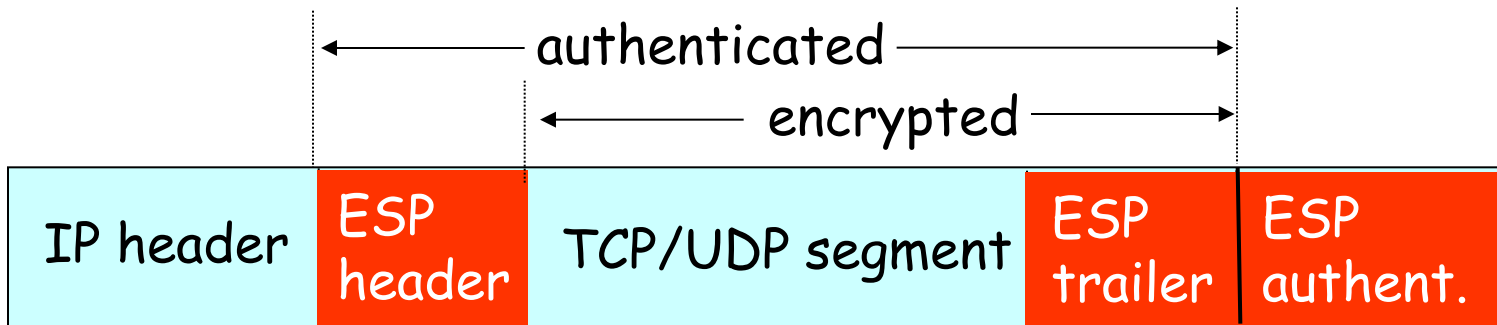
## AH header includes:

- ❑ connection identifier
- ❑ authentication data: source- signed message digest calculated over original IP datagram.
- ❑ next header field: specifies type of data (e.g., TCP, UDP, ICMP)



# ESP Protocol

- ❑ provides secrecy, host authentication, data integrity.
- ❑ data, ESP trailer encrypted.
- ❑ next header field is in ESP trailer.
- ❑ ESP authentication field is similar to AH authentication field.
- ❑ Protocol = 50.



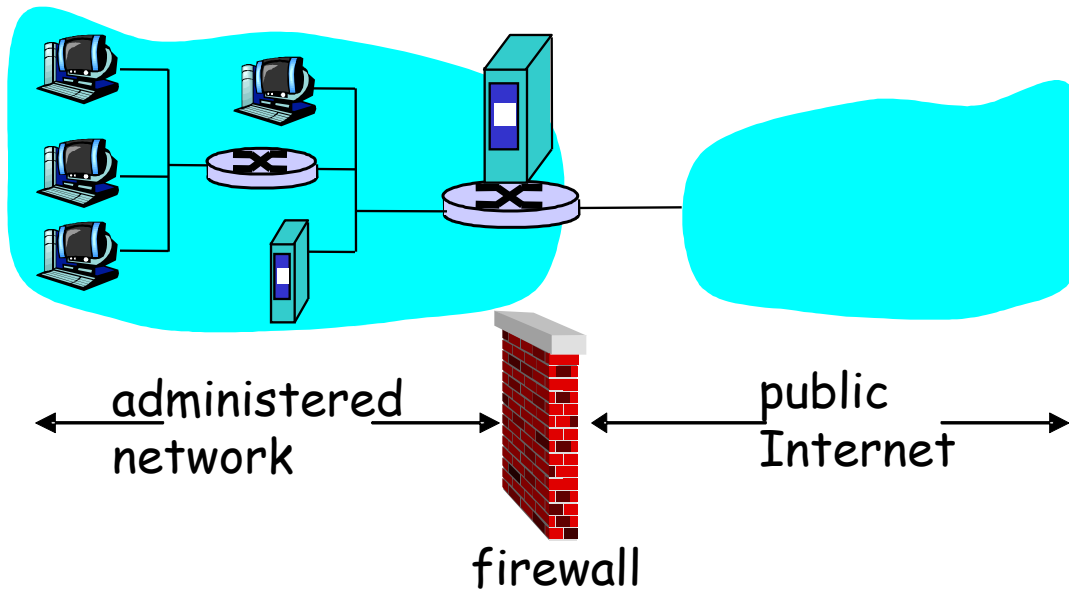
# Roadmap

1. What is network security?
2. Principles of cryptography
3. Message integrity
4. End point authentication
5. Securing e-mail
6. Securing TCP connections: SSL
7. Network layer security: IPsec
8. Operational security: firewalls and IDS

# Firewalls

## firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others.

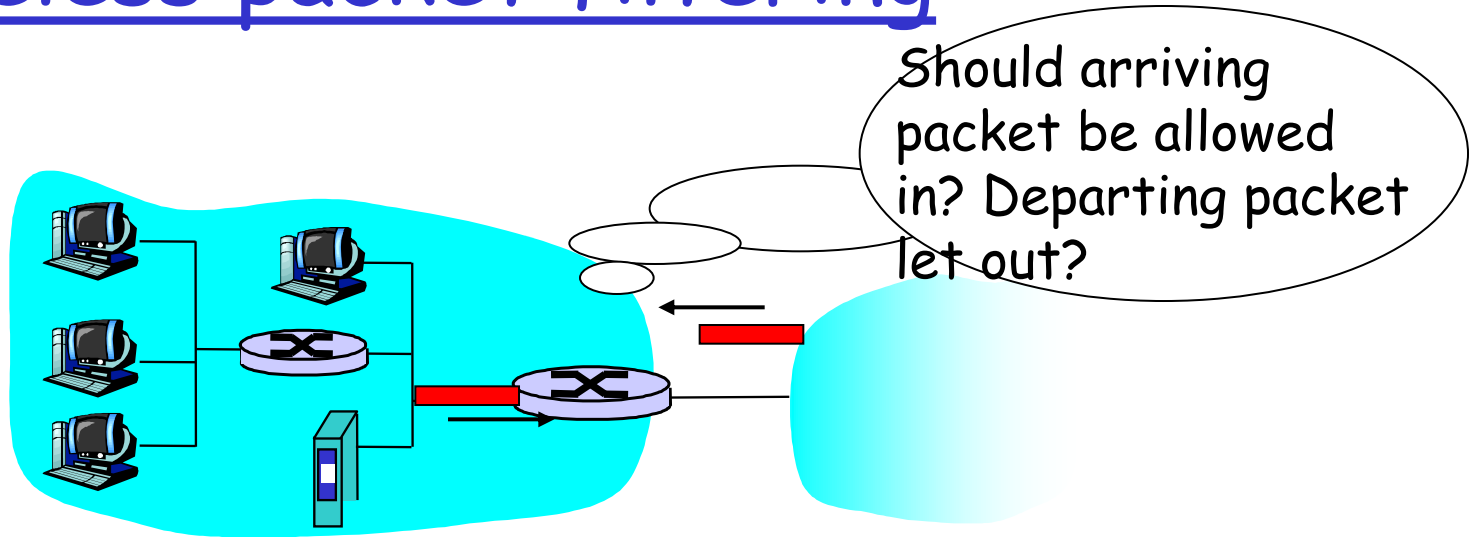




# Firewalls: Why

- ❑ 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)
- ❑ three types of firewalls:
  - stateless packet filters
  - stateful packet filters
  - application gateways

# Stateless packet filtering



- ❑ internal network connected to Internet via **router firewall**
- ❑ router **filters packet-by-packet**, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

# Stateless packet filtering: example

- ❑ example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - all incoming, outgoing UDP flows and telnet connections are blocked.
- ❑ example 2: Block inbound TCP segments with ACK=0.
  - prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

# Stateless packet filtering: more examples

<u>Policy</u>	<u>Firewall Setting</u>
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

# Access Control Lists

- **ACL**: table of rules, applied top to bottom to incoming packets: (action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

# Stateful packet filtering

- ❑ stateless packet filter: heavy handed tool
  - admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- ❑ *stateful packet filter*: track status of every TCP connection
  - track connection setup (SYN), teardown (FIN): can determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets

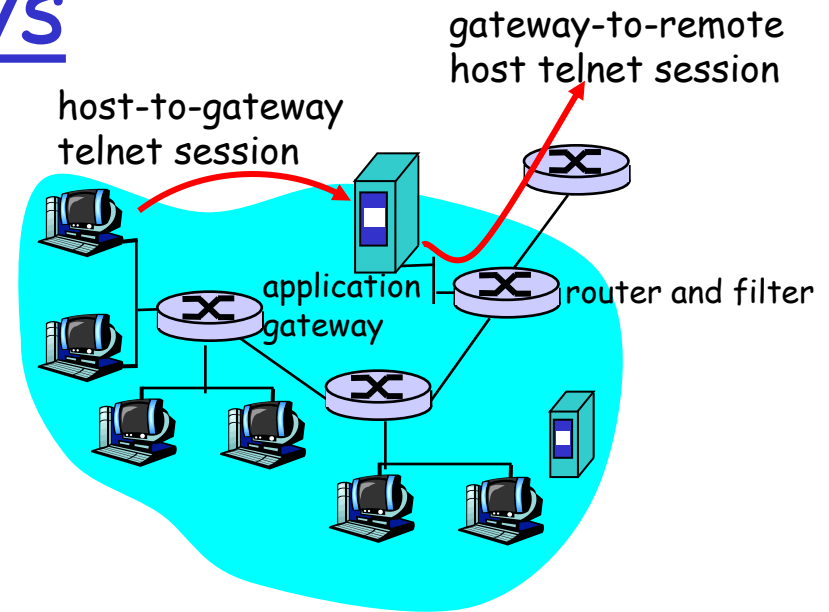
# Stateful packet filtering

- ACL augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	×
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	×
deny	all	all	all	all	all	all	

# Application gateways

- ❑ filters packets on application data as well as on IP/TCP/UDP fields.
- ❑ example: allow select internal users to telnet outside.



1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.



# Limitations of firewalls and gateways

- ❑ IP spoofing: router can't know if data "really" comes from claimed source
- ❑ if multiple app's. need special treatment, each has own app. gateway.
- ❑ client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- ❑ filters often use all or nothing policy for UDP.
- ❑ tradeoff: **degree of communication with outside world, level of security**
- ❑ many highly protected sites still suffer from attacks.

# Intrusion detection systems

## ❑ packet filtering:

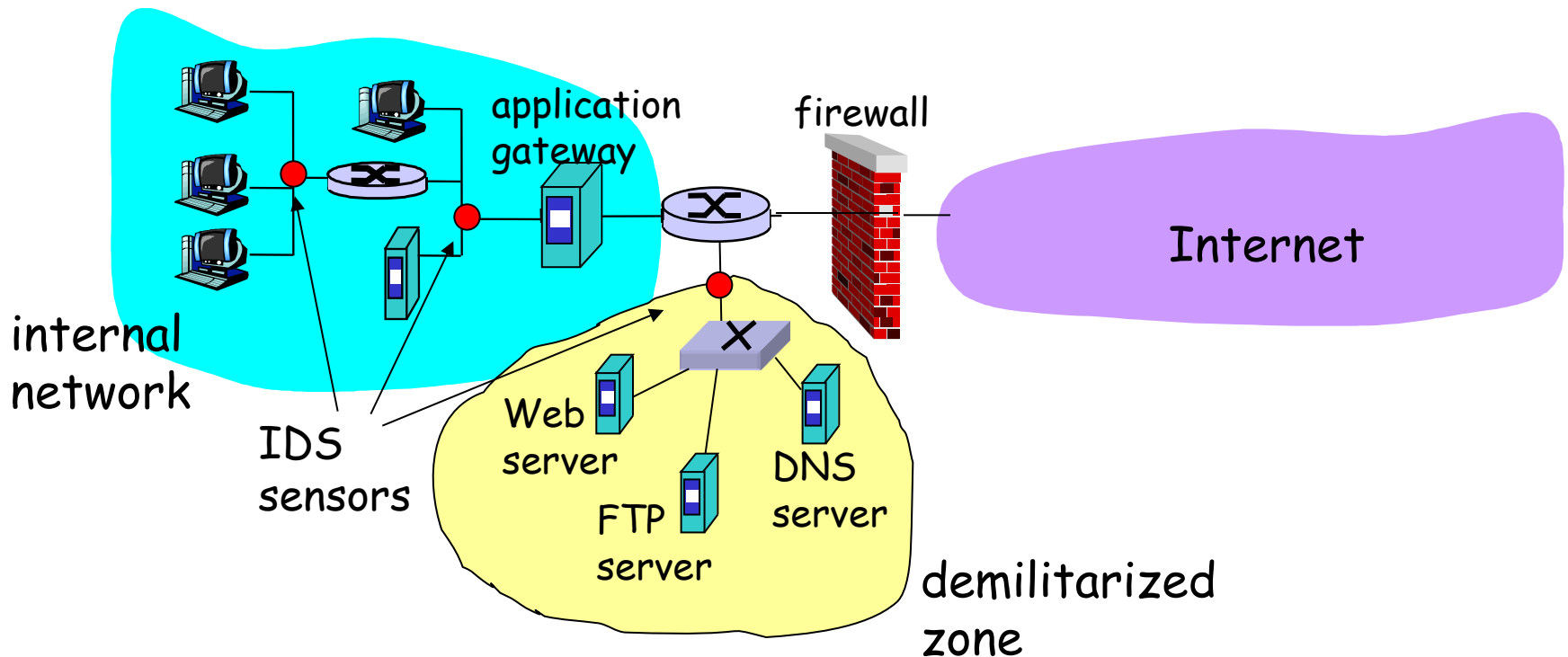
- operates on TCP/IP headers only
- no correlation check among sessions

## ❑ *IDS: intrusion detection system*

- *deep packet inspection*: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
- *examine correlation* among multiple packets
  - port scanning
  - network mapping
  - DoS attack

# Intrusion detection systems

- multiple IDSs: different types of checking at different locations



# Network Security (summary)

Basic techniques.....

- cryptography (symmetric and public)
- message integrity
- end-point authentication

.... used in many different security scenarios

- secure email
- secure transport (SSL)
- IP sec

Operational Security: firewalls and IDS