

# Chapter 7: Network Security

## Chapter goals:

- ❑ understand principles of network security:
  - cryptography and its *many* uses beyond “confidentiality”
  - authentication
  - message integrity
  - key distribution
- ❑ security in practice:
  - firewalls
  - security in application, transport, network, link layers

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

7.5 Key Distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# What is network security?

**Confidentiality:** only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

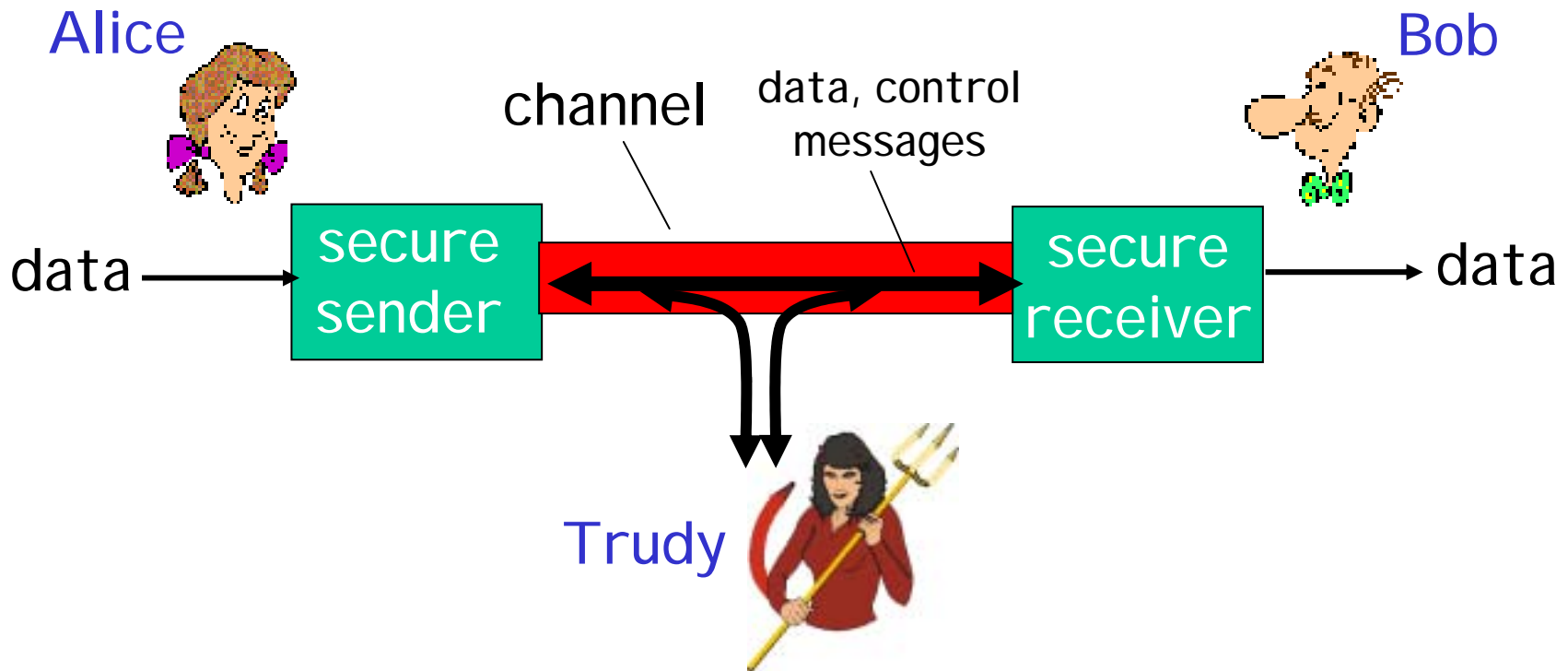
**Authentication:** sender, receiver want to confirm identity of each other

**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!

- *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)

*more on this later .....*

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

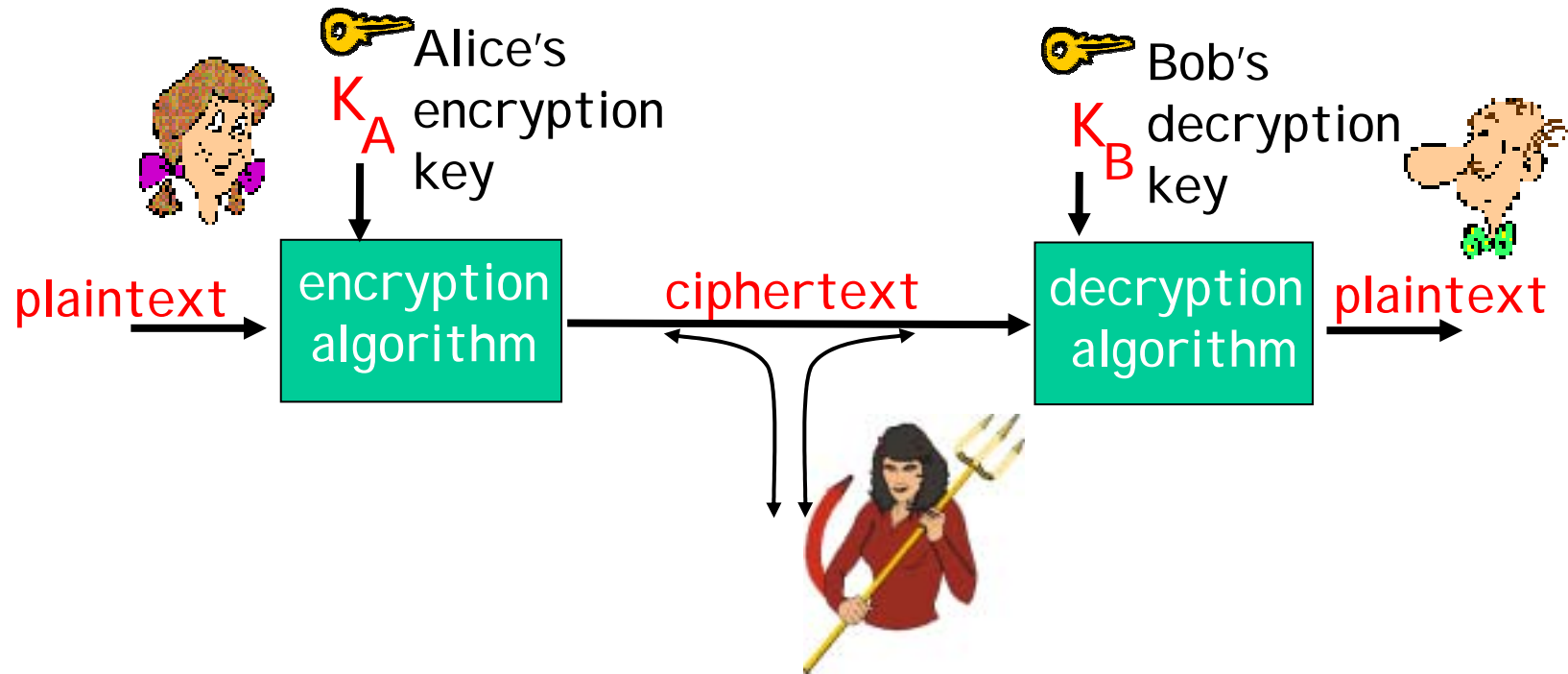
7.5 Key Distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# The language of cryptography



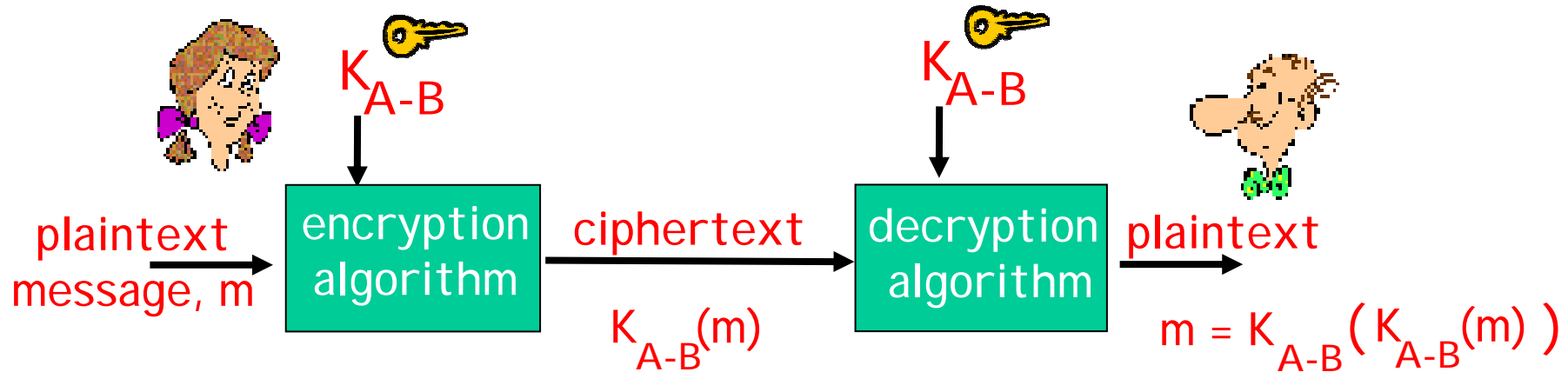
**symmetric key** crypto: sender, receiver keys *identical*

**public-key** crypto: encryption key *public*, decryption key *secret* (private)





# Symmetric key cryptography



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

# Symmetric key crypto: DES

## DES: Data Encryption Standard

- ❑ US encryption standard [NI ST 1993]
- ❑ 56-bit symmetric key, 64-bit plaintext input
- ❑ How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase (“Strong cryptography makes the world a safer place”) decrypted (brute force) in 4 months
  - no known “backdoor” decryption approach
- ❑ making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

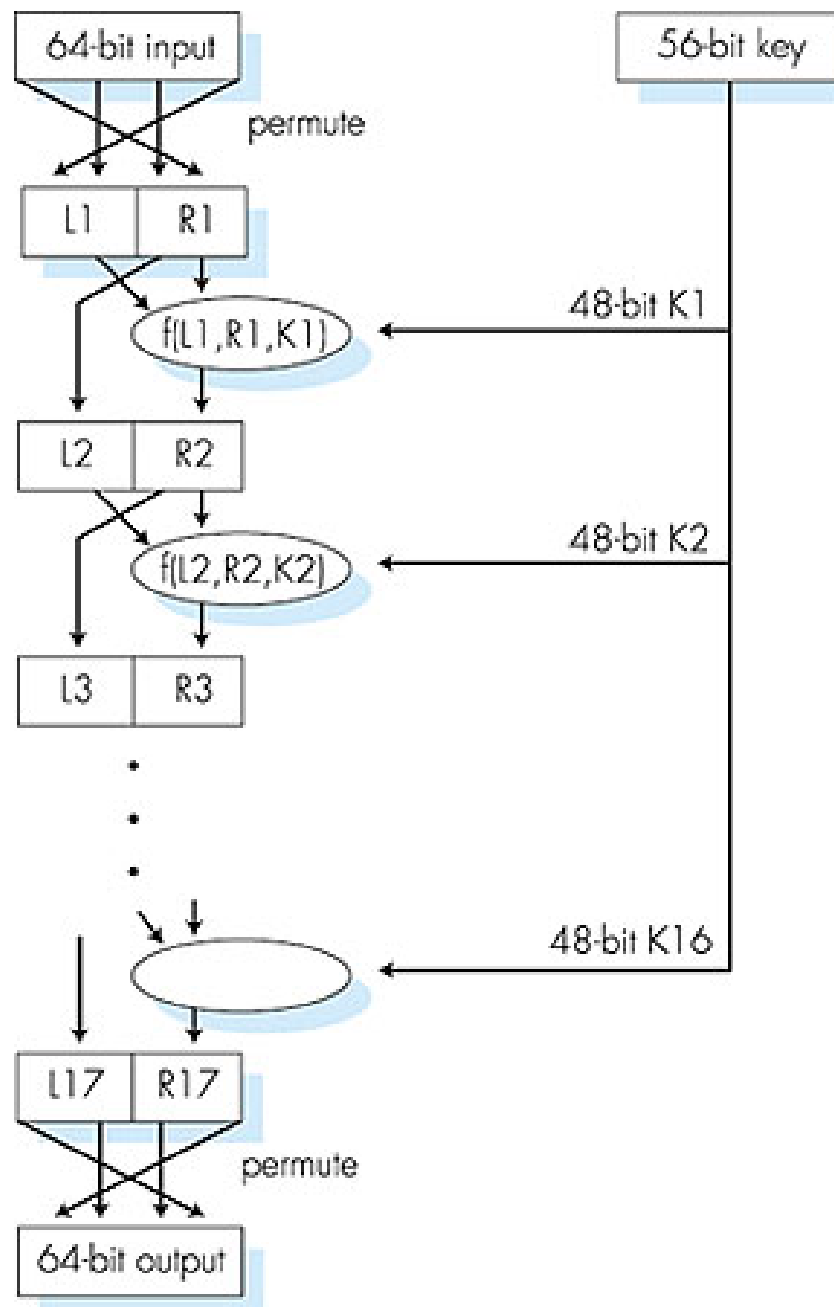
# Symmetric key crypto: DES

## DES operation

initial permutation

16 identical "rounds" of  
function application,  
each using different  
48 bits of key

final permutation



# 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

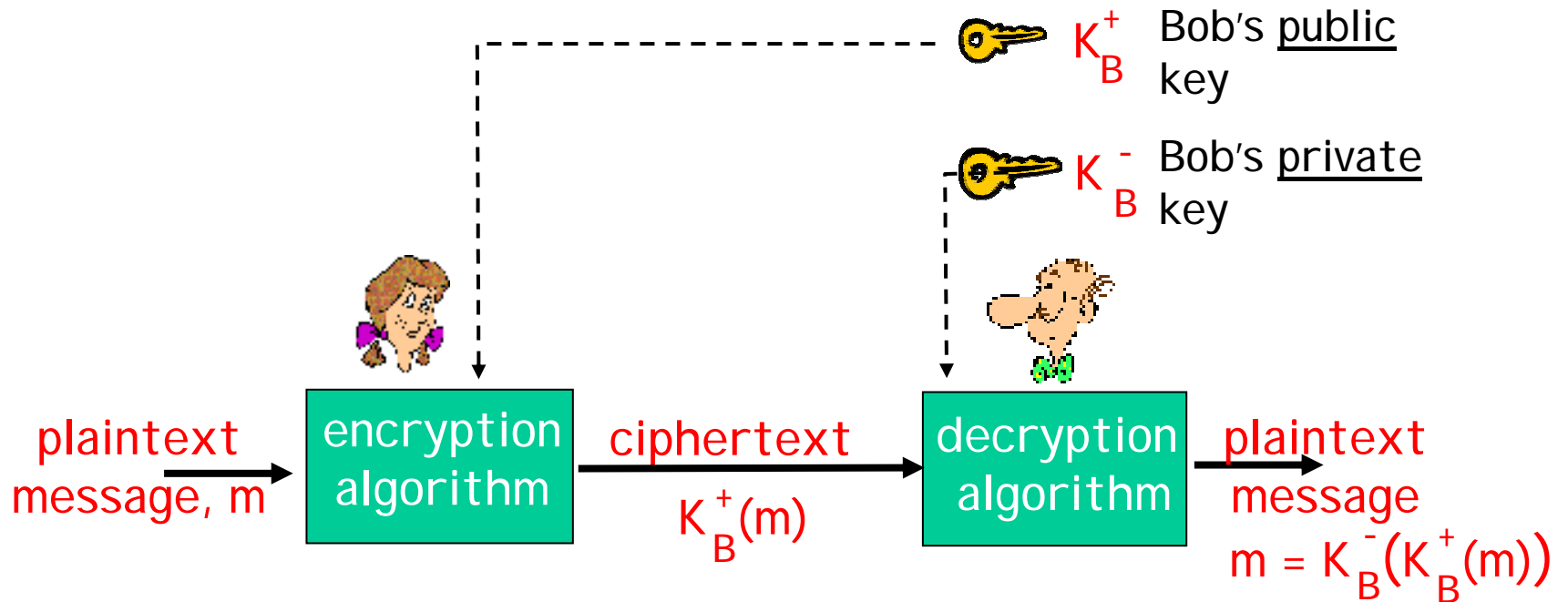
- ❑ requires sender, receiver know shared secret key
- ❑ Q: how to agree on key in first place (particularly if never "met")?

## 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



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


# 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:	<u>letter</u>	<u>m</u>	<u>m<sup>e</sup></u>	<u>c = m<sup>e</sup> mod n</u>
	I	12	1524832	17
decrypt:	<u>c</u>	<u>c<sup>d</sup></u>	<u>m = c<sup>d</sup> mod n</u>	<u>letter</u>
	17	481968572106750915091411825223071697	12	I

# RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

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

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

---

$$\begin{aligned} (m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{ed \bmod (p-1)(q-1)} \bmod n \\ &\quad \text{(using number theory result above)} \\ &= m^1 \bmod n \\ &\quad \text{(since we chose } ed \text{ to be divisible by} \\ &\quad \text{(} (p-1)(q-1) \text{ with remainder 1 )} \\ &= m \end{aligned}$$

# 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!*

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

7.5 Key Distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# Authentication

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

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



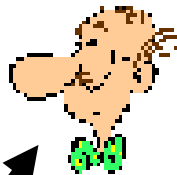
Failure scenario??



# Authentication

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

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



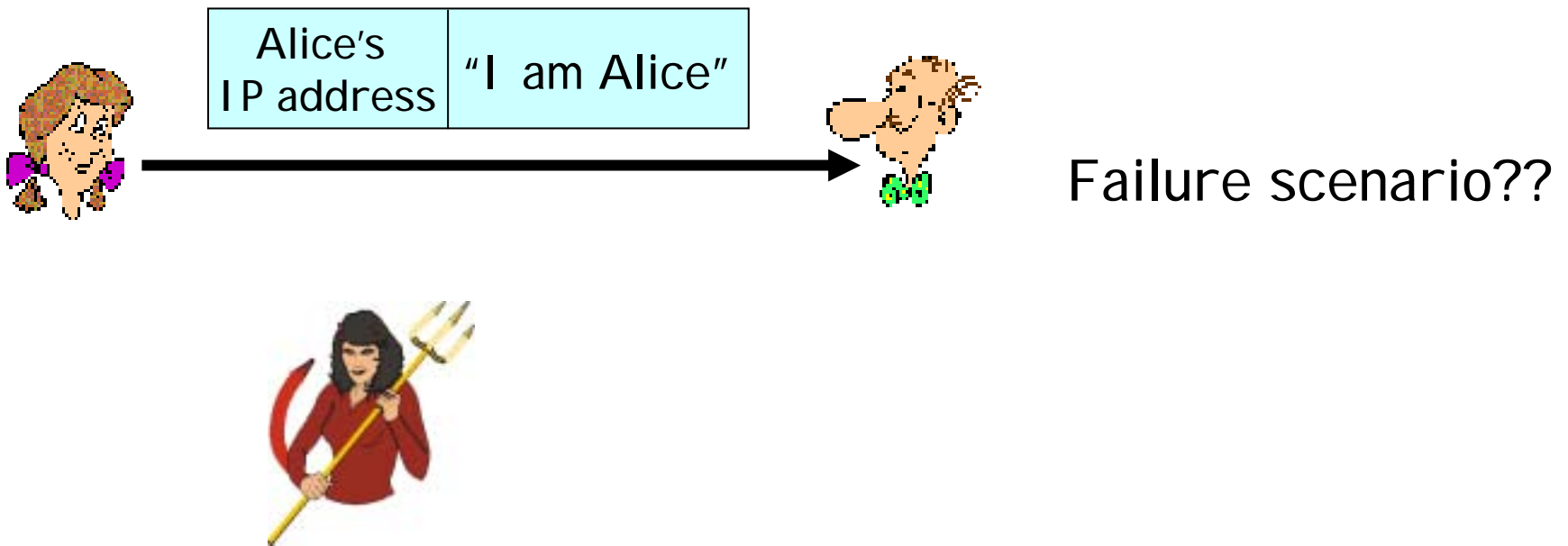
“I am Alice”

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



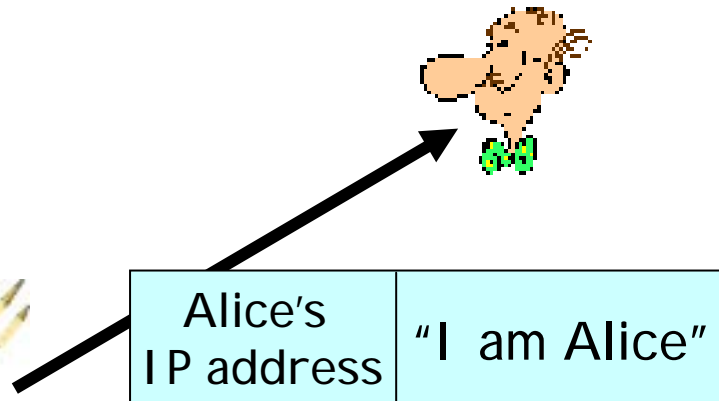
# Authentication: another try

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



# Authentication: another try

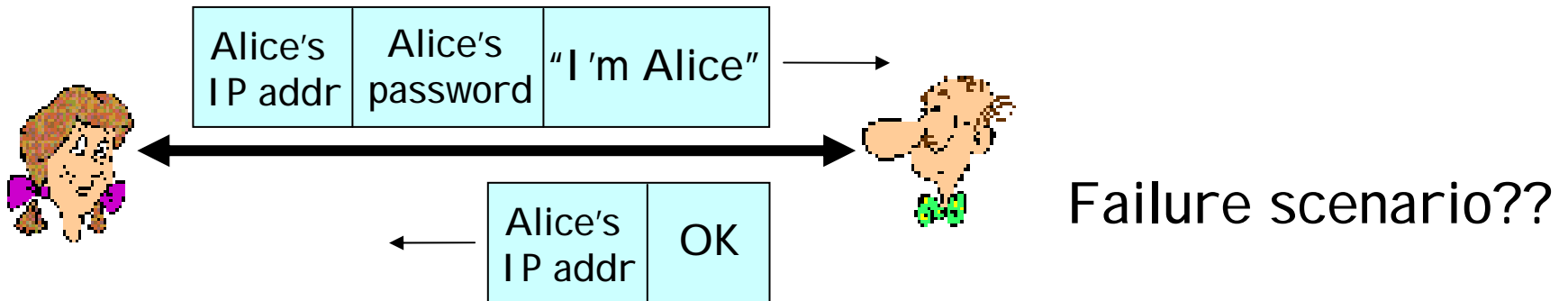
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



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

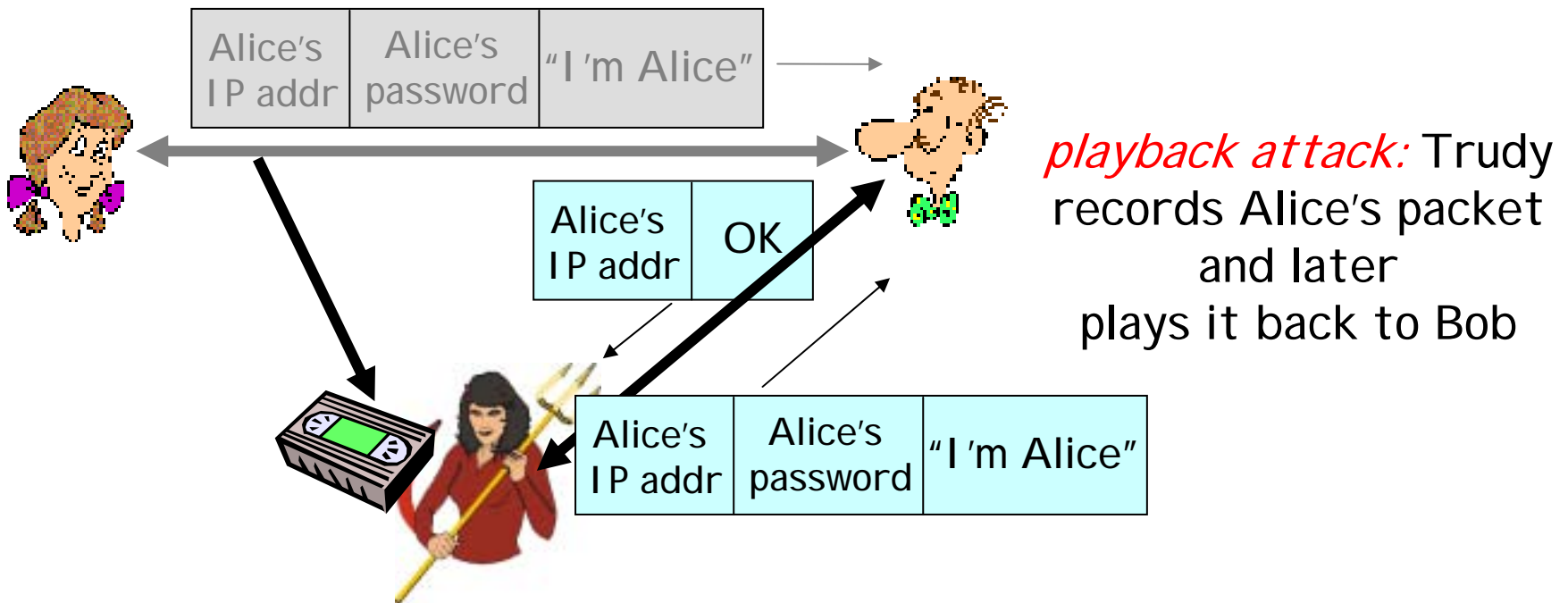
# Authentication: another try

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



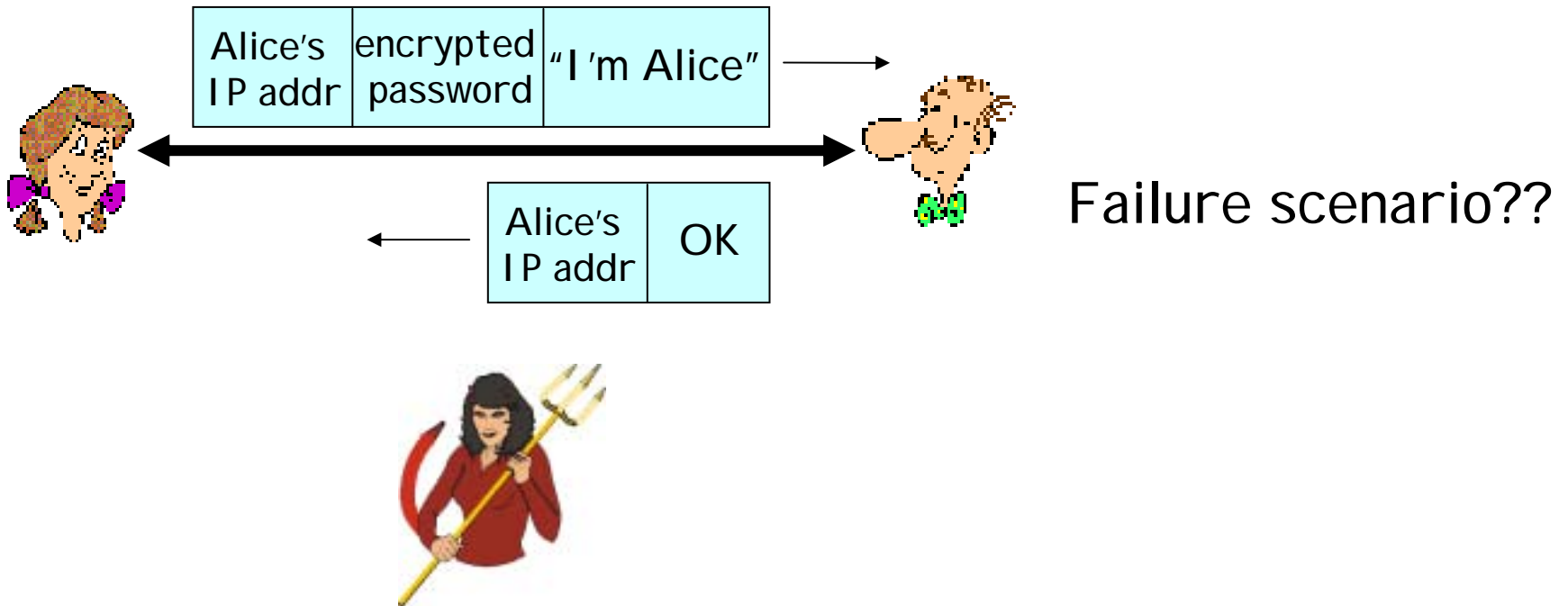
# Authentication: another try

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



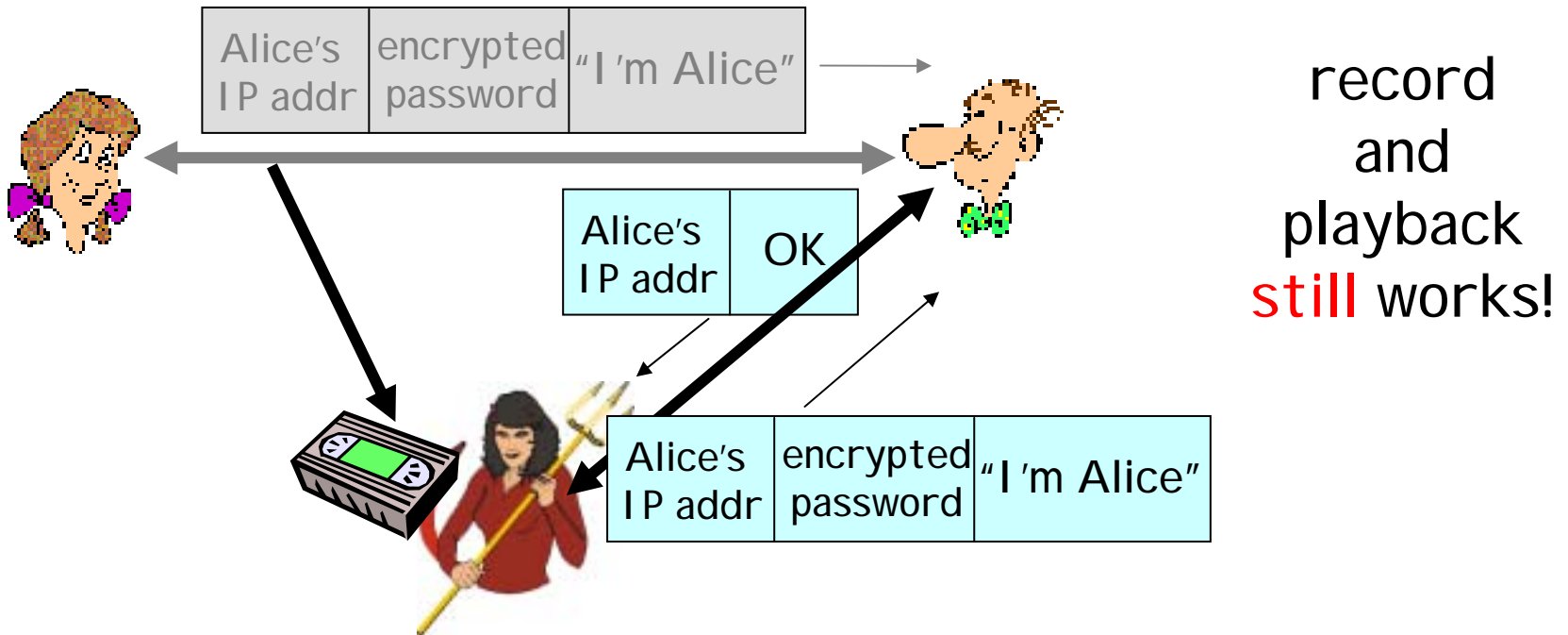
# Authentication: yet another try

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.

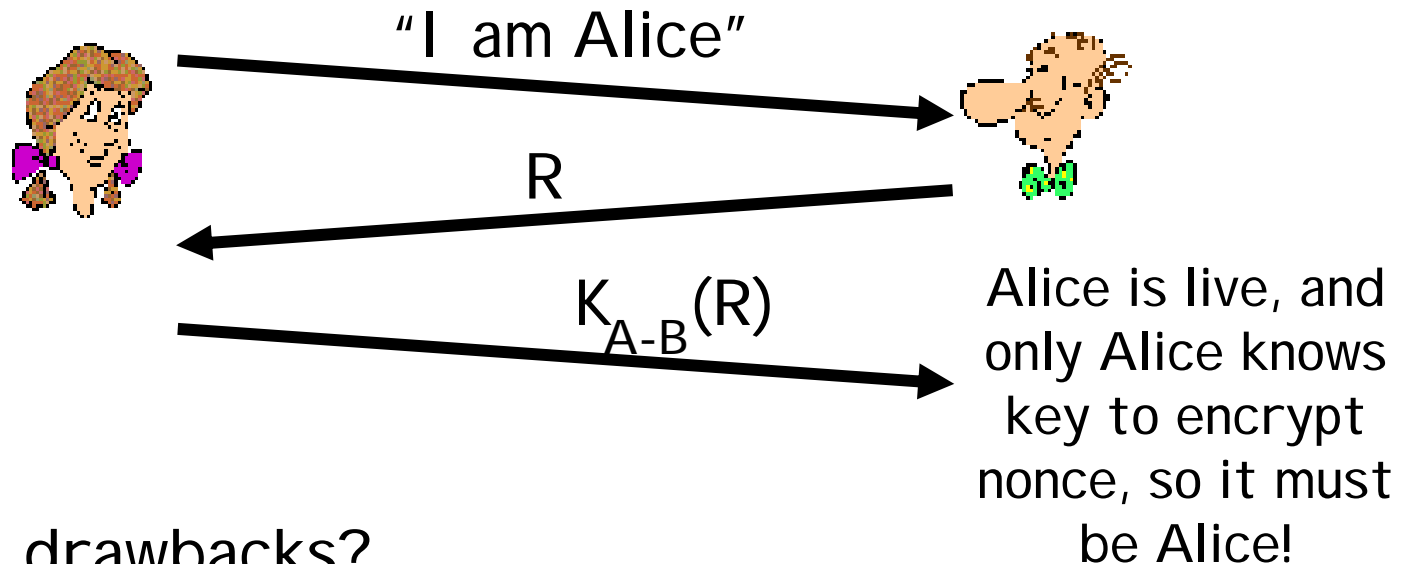


# Authentication: yet another try

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



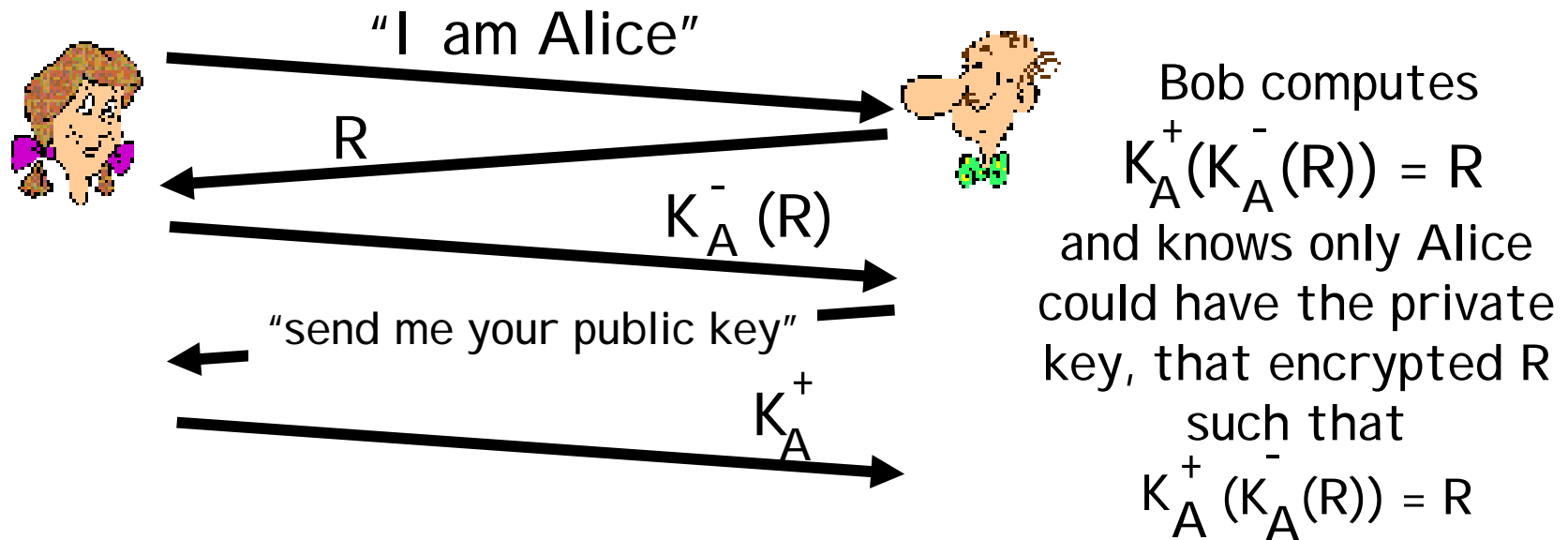
Failures, drawbacks?

# Authentication: ap5.0

ap4.0 requires shared symmetric key

□ can we authenticate using public key techniques?

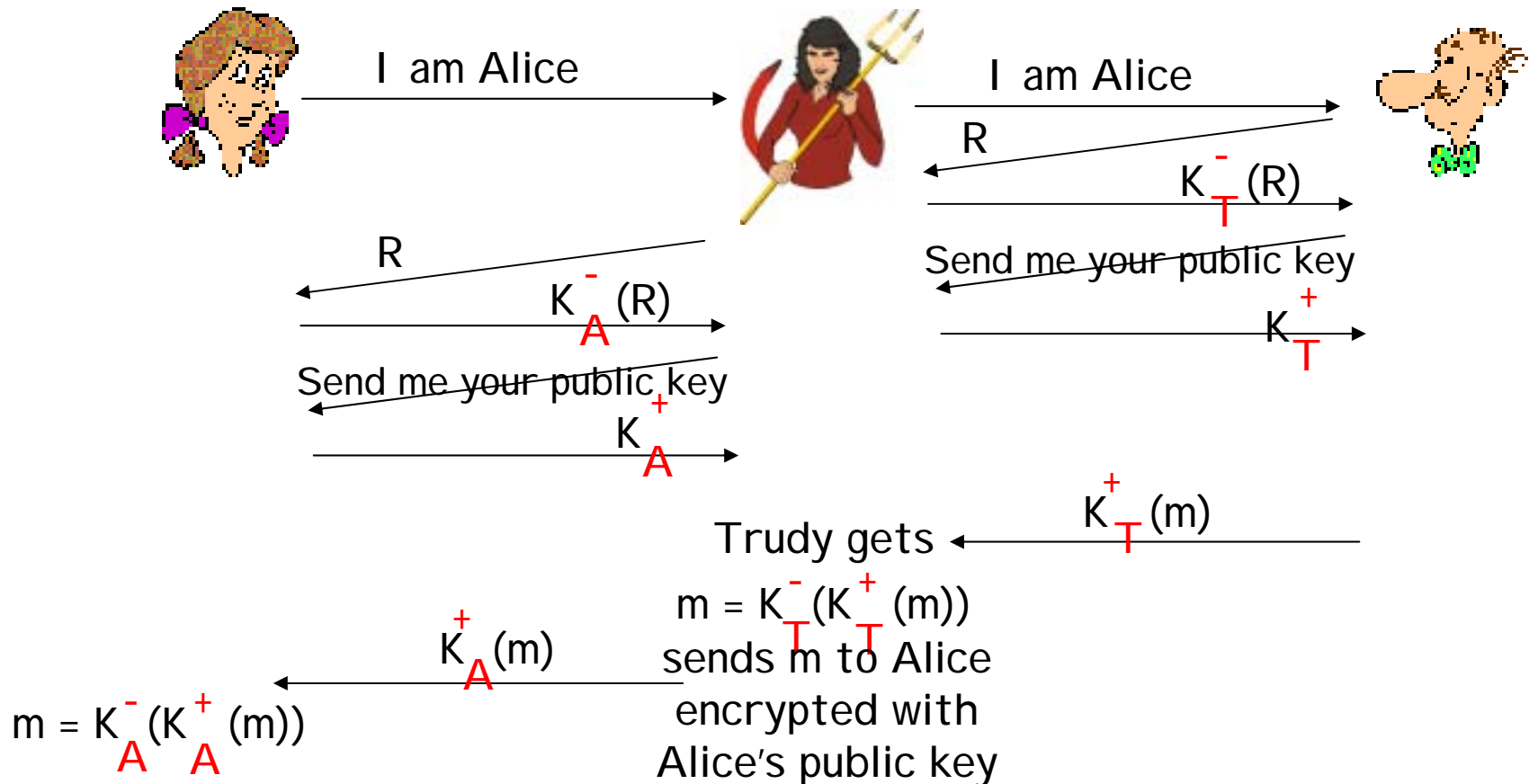
ap5.0: use nonce, public key cryptography





# 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!

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Message integrity

7.5 Key Distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# Digital Signatures

Cryptographic technique analogous to handwritten signatures.

- ❑ sender (Bob) digitally signs document, establishing he is document owner/creator.
- ❑ **verifiable, nonforgeable**: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

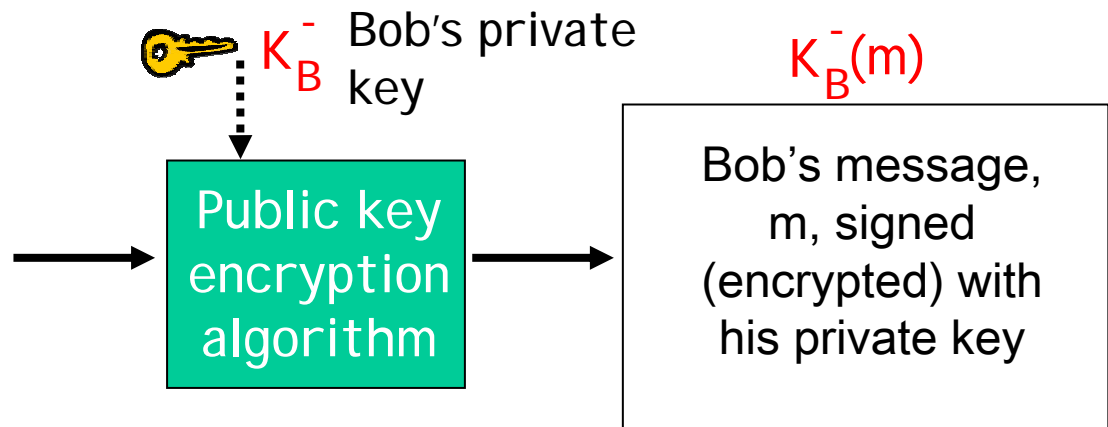
# Digital Signatures

## 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



# 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:

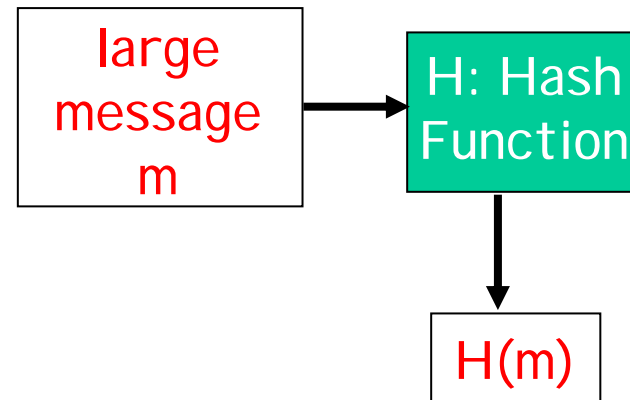
- ✓ Alice can take  $m$ , and signature  $K_B^-(m)$  to court and prove that Bob signed  $m$ .

# Message Digests

Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy-to-compute digital “fingerprint”

- apply hash function  $H$  to  $m$ , get fixed size message digest,  $H(m)$ .



**Hash function properties:**

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest  $x$ , computationally infeasible to find  $m$  such that  $x = H(m)$

# Internet checksum: poor crypto hash function

- Internet checksum has some properties of hash function:
- produces fixed length digest (16-bit sum) of message
  - is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

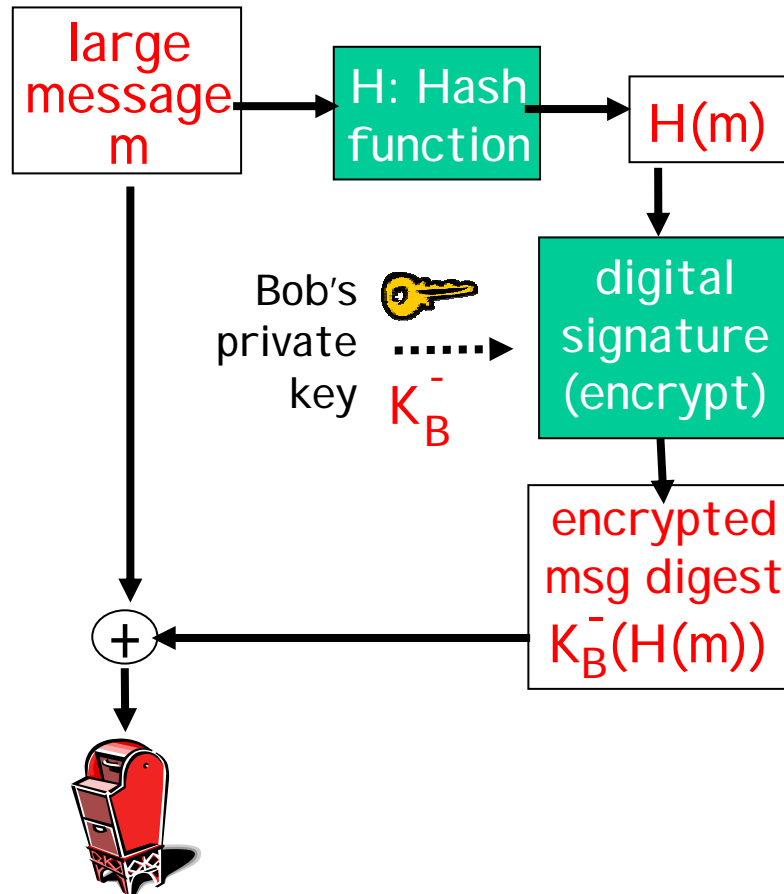
<u>message</u>	<u>ASCII format</u>	<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31	I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39	0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	<u>39 42 D2 42</u>	9 B O B	<u>39 42 D2 42</u>

B2 C1 D2 AC — different messages — B2 C1 D2 AC  
but identical checksums!

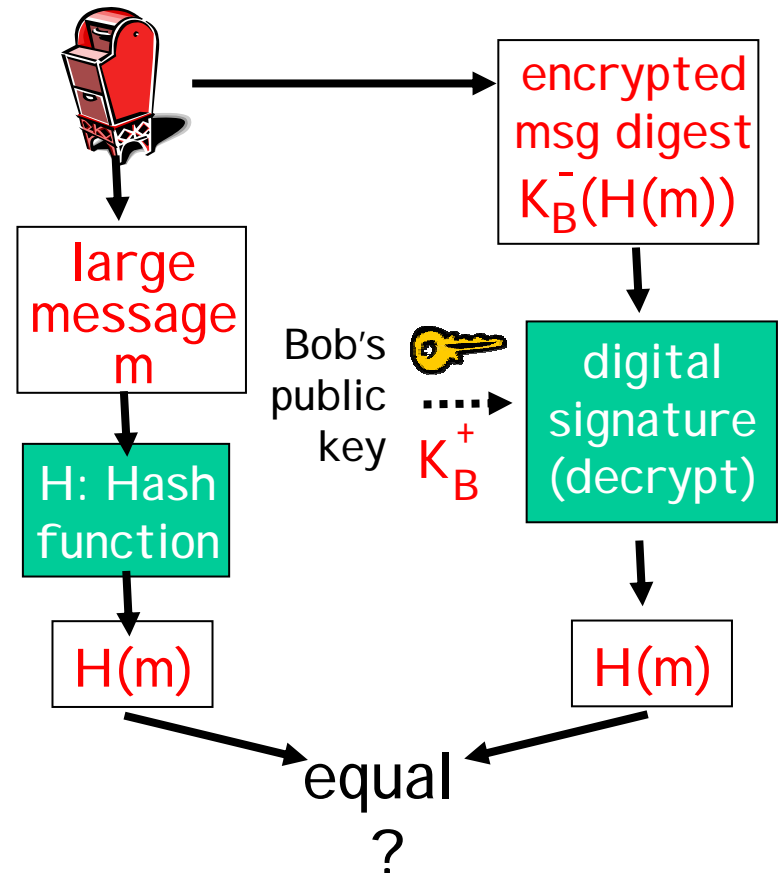


# Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



# Hash Function Algorithms

- ❑ MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string  $x$ , appears difficult to construct msg  $m$  whose MD5 hash is equal to  $x$ .
- ❑ SHA-1 is also used.
  - US standard [NI ST, FIPS PUB 180-1]
  - 160-bit message digest

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

7.5 Key distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# Trusted Intermediaries

## Symmetric key problem:

- ❑ How do two entities establish shared secret key over network?

## **Solution:**

- ❑ trusted key distribution center (KDC) acting as intermediary between entities

## Public key problem:

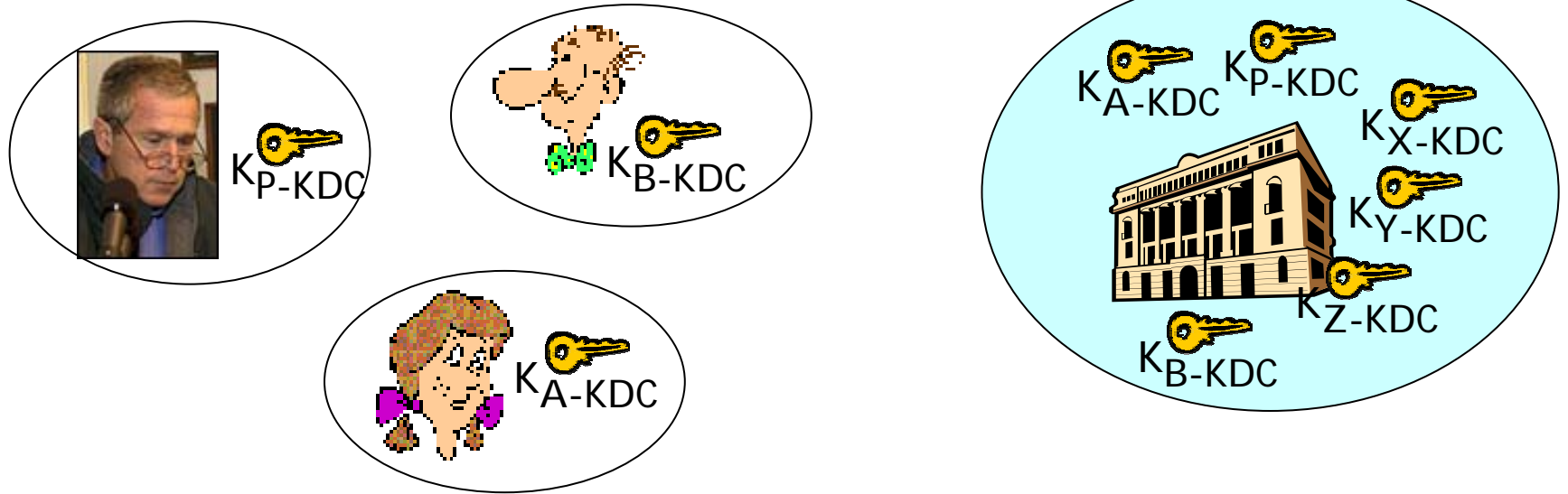
- ❑ When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

## **Solution:**

- ❑ trusted certification authority (CA)

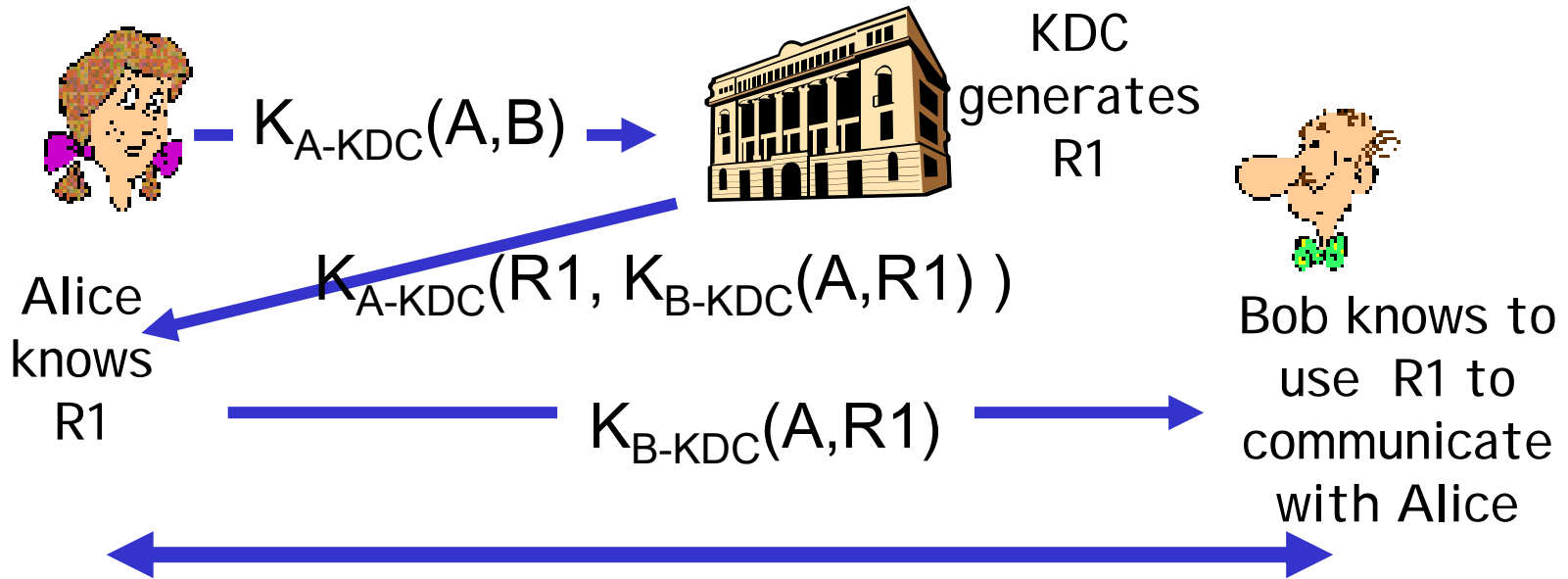
# Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- **KDC**: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys,  $K_{A-KDC}$   $K_{B-KDC}$ , for communicating with KDC.



# Key Distribution Center (KDC)

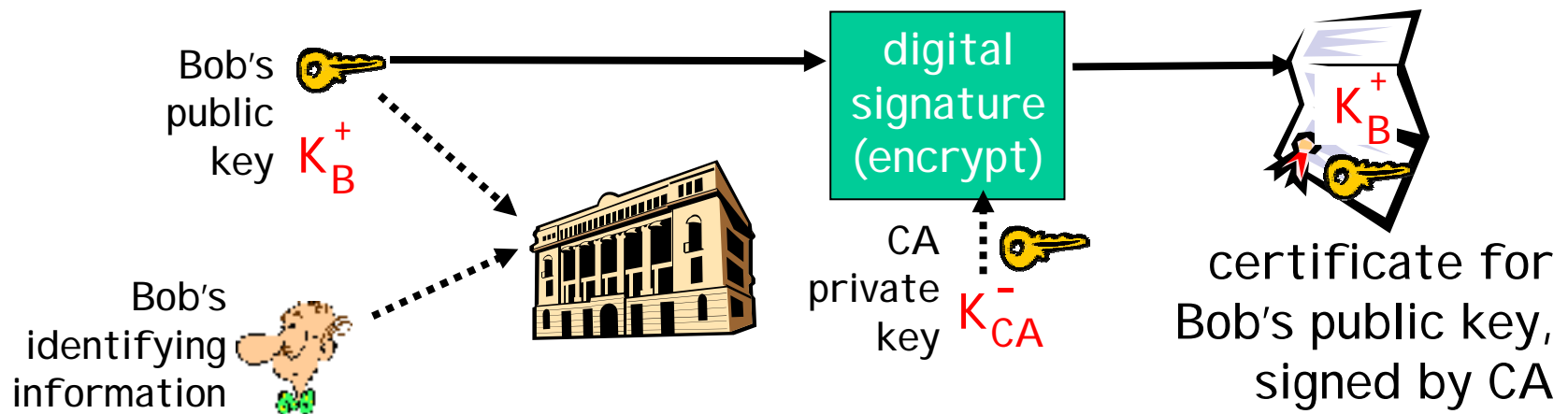
Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using  $R1$  as *session key* for shared symmetric encryption

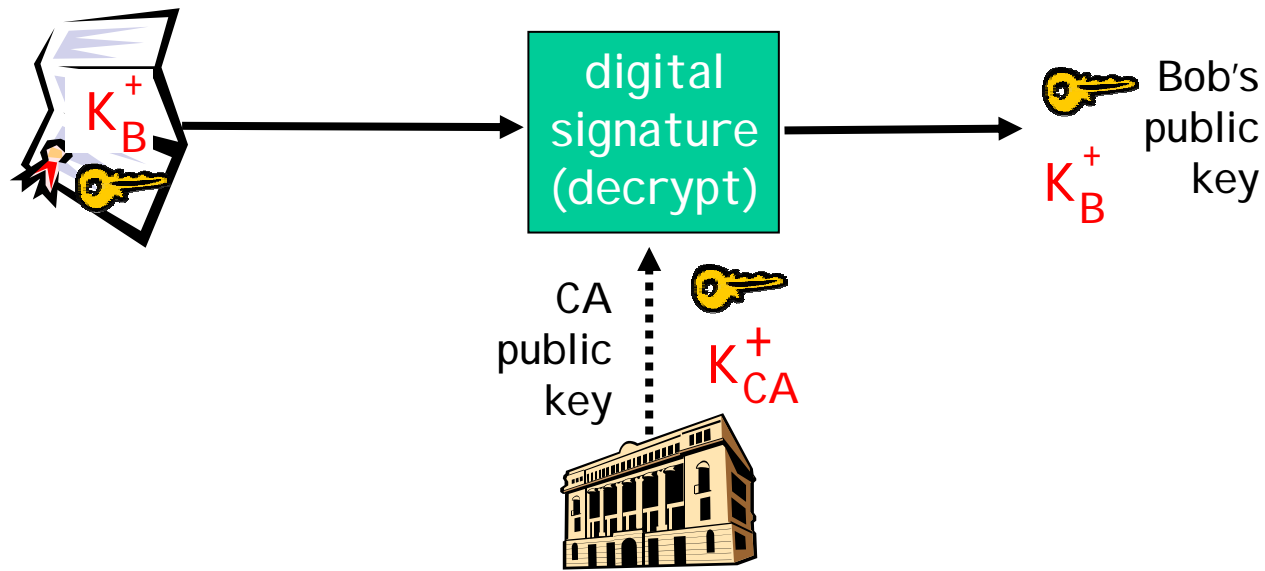
# Certification Authorities

- ❑ **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

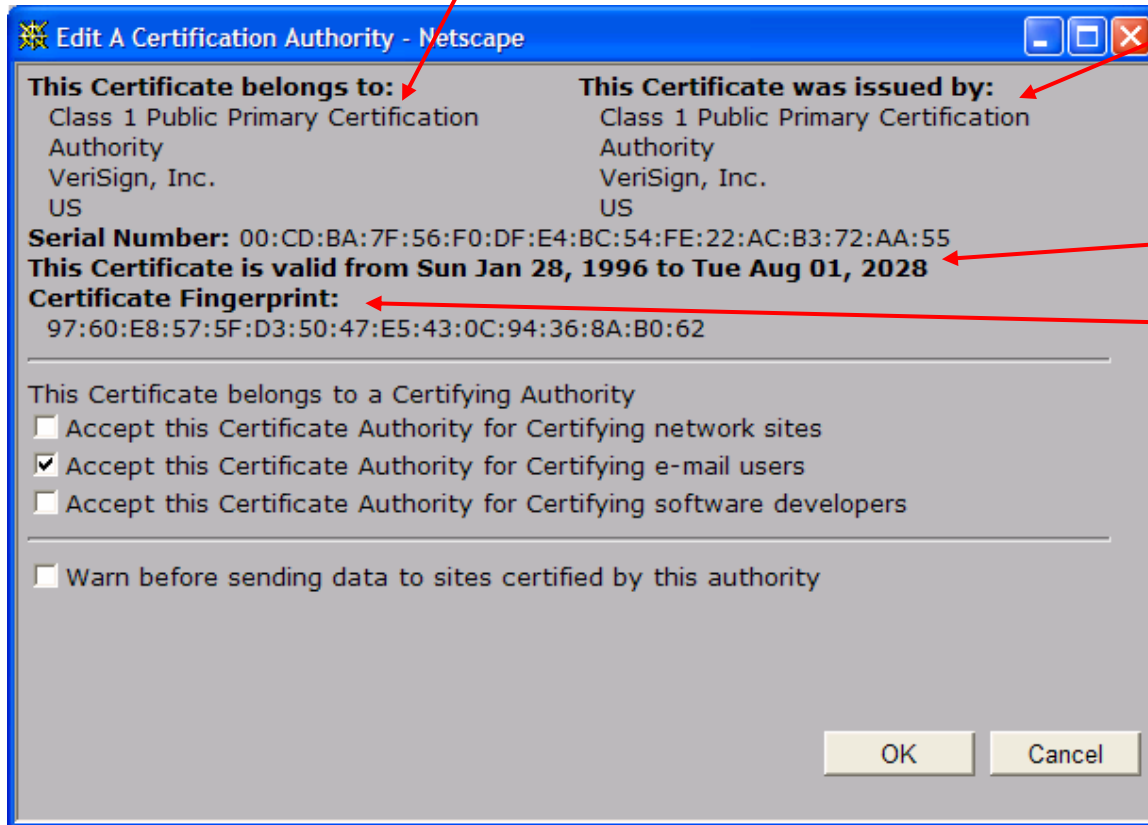
- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key





# 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

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

7.5 Key Distribution and certification

7.6 Access control: firewalls

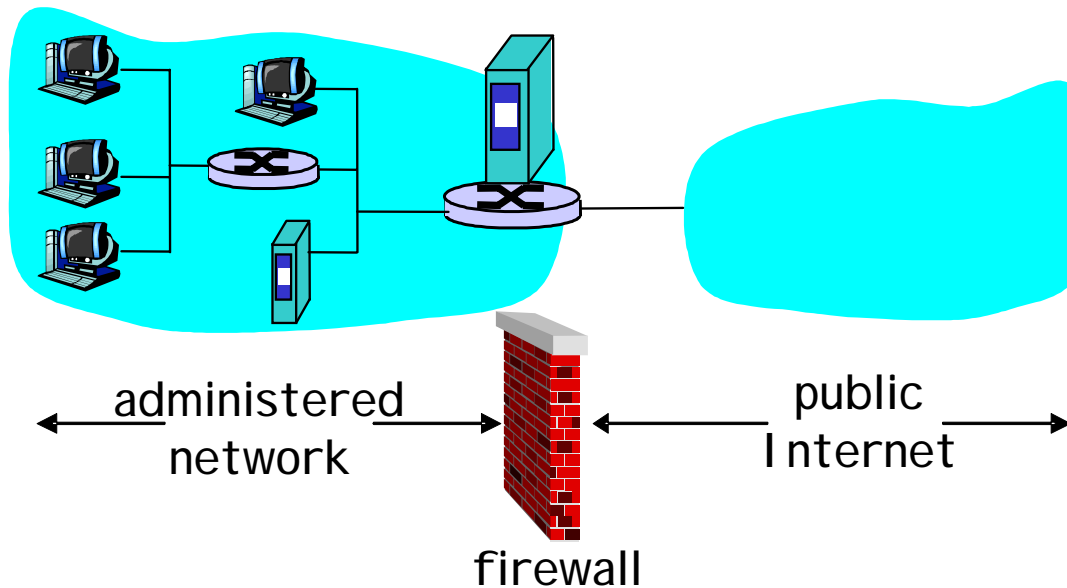
7.7 Attacks and counter measures

7.8 Security in many layers

# 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.

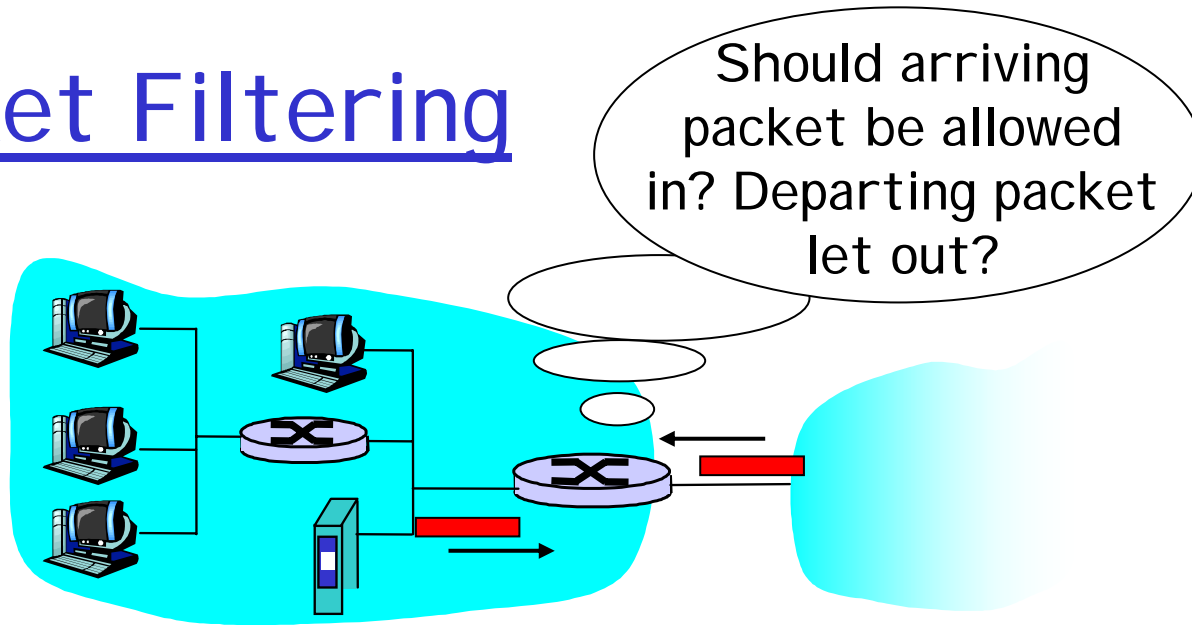
- e.g., attacker replaces CIA’s homepage with something else

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

two types of firewalls:

- application-level
- packet-filtering

# Packet Filtering



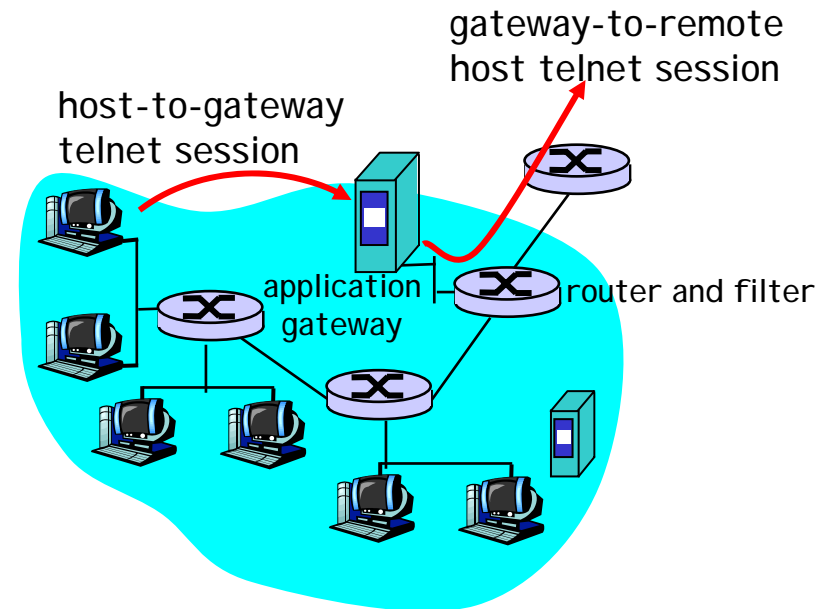
- ❑ internal network connected to Internet via **router firewall**
- ❑ router **filters packet-by-packet**, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

# Packet Filtering

- ❑ Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and 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.

# 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.



# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Authentication

7.4 Integrity

7.5 Key Distribution and certification

7.6 Access control: firewalls

7.7 Attacks and counter measures

7.8 Security in many layers

# Internet security threats

## Mapping:

- before attacking: “case the joint” – find out what services are implemented on network
- Use `ping` to determine what hosts have addresses on network
- Port-scanning: try to establish TCP connection to each port in sequence (see what happens)
- `nmap` (<http://www.insecure.org/nmap/>) mapper: “network exploration and security auditing”

## Countermeasures?

# Internet security threats

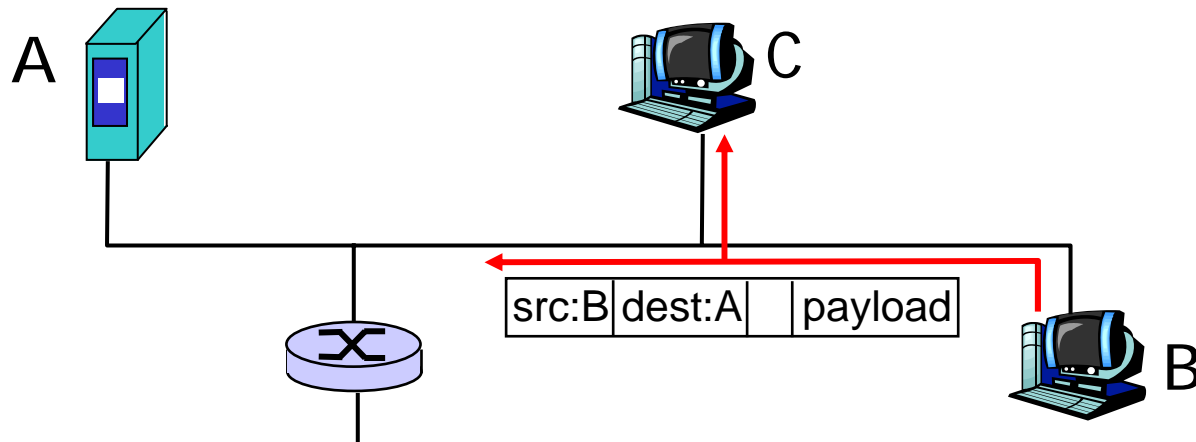
## Mapping: countermeasures

- record traffic entering network
- look for suspicious activity (IP addresses, ports being scanned sequentially)

# Internet security threats

## Packet sniffing:

- broadcast media
- promiscuous N I C reads all packets passing by
- can read all unencrypted data (e.g. passwords)
- e.g.: C sniffs B's packets

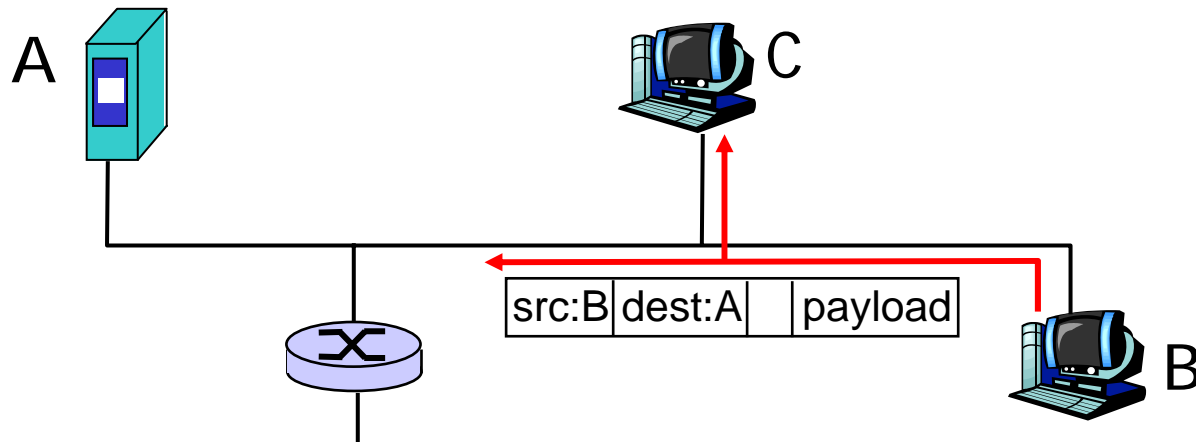


Countermeasures?

# Internet security threats

## Packet sniffing: countermeasures

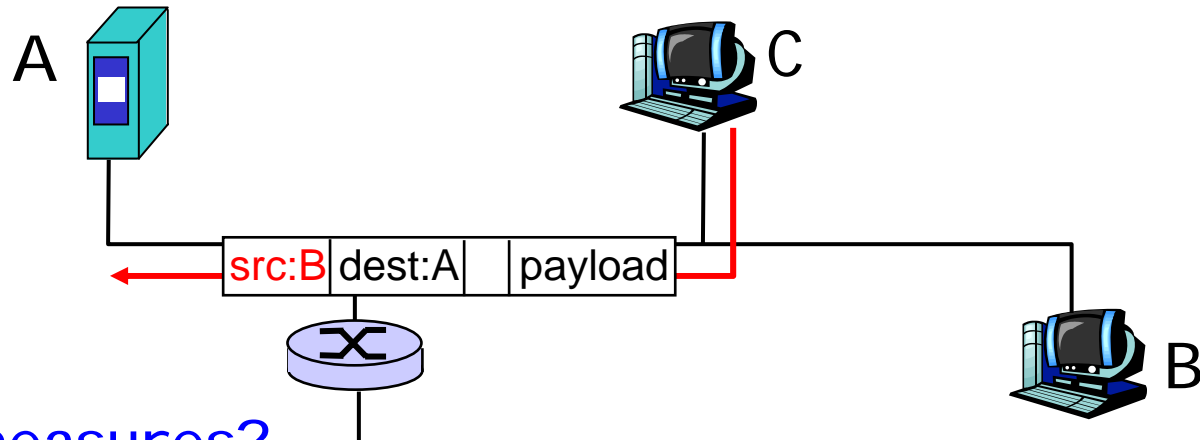
- all hosts in organization run software that checks periodically if host interface in promiscuous mode.
- one host per segment of broadcast media (switched Ethernet at hub)



# Internet security threats

## IP Spoofing:

- can generate “raw” IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is spoofed
- e.g.: C pretends to be B

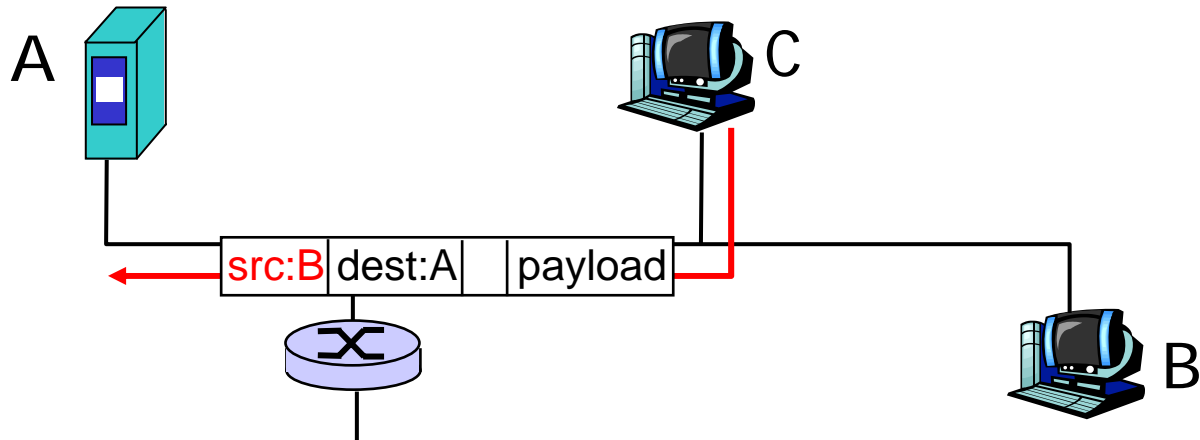


Countermeasures?

# Internet security threats

## IP Spoofing: ingress filtering

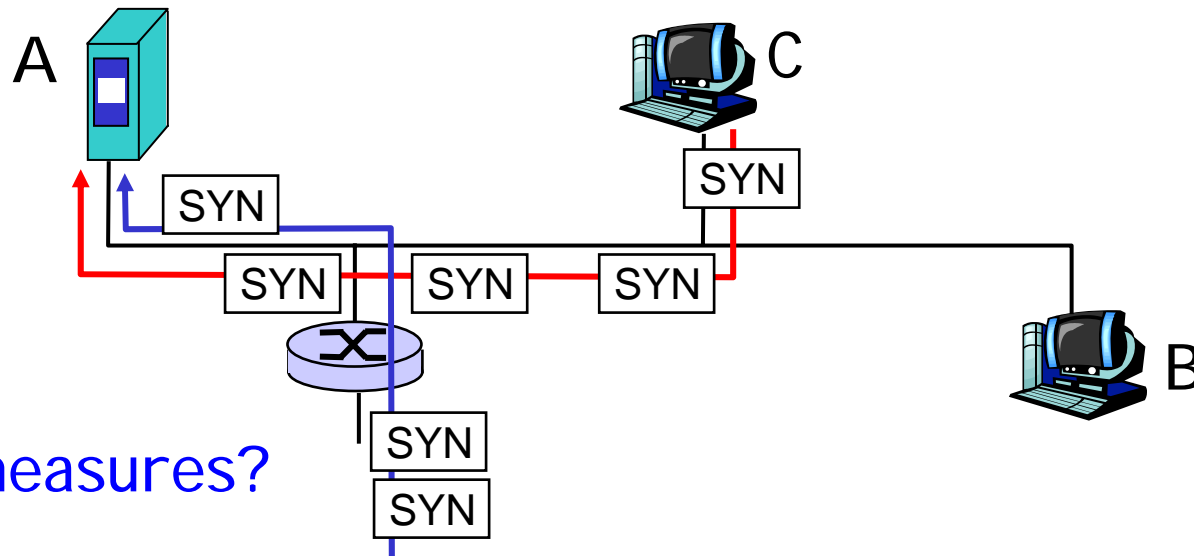
- routers should not forward outgoing packets with invalid source addresses (e.g., datagram source address not in router's network)
- great, but ingress filtering can not be mandated for all networks



# Internet security threats

## Denial of service (DOS):

- flood of maliciously generated packets “swamp” receiver
- Distributed DOS (DDOS): multiple coordinated sources swamp receiver
- e.g., C and remote host SYN-attack A



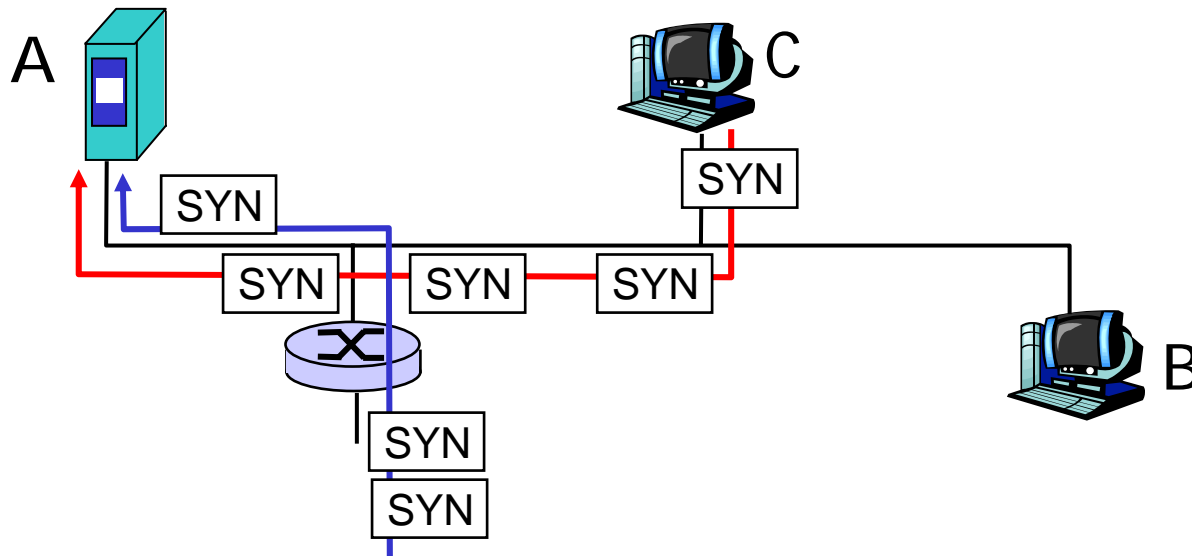
Countermeasures?



# Internet security threats

## Denial of service (DOS): countermeasures

- filter out flooded packets (e.g., SYN) before reaching host: throw out good with bad
- **traceback** to source of floods (most likely an innocent, compromised machine)

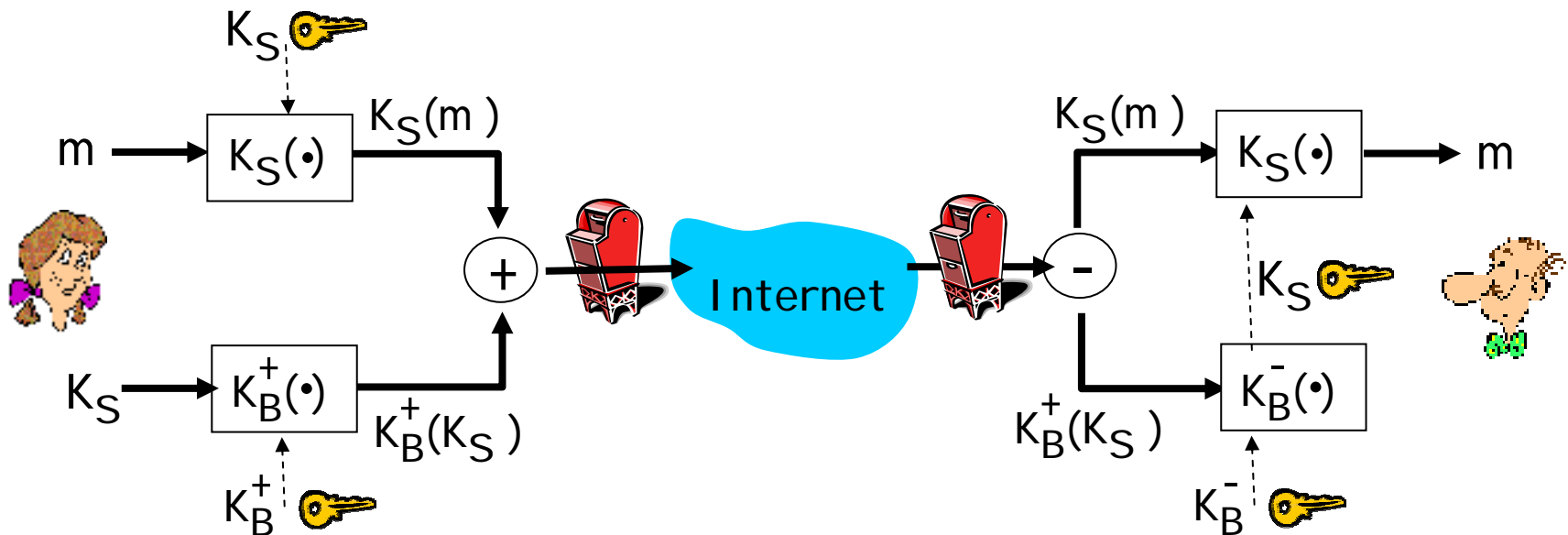


# Chapter 7 roadmap

- 7.1 What is network security?
- 7.2 Principles of cryptography
- 7.3 Authentication
- 7.4 Integrity
- 7.5 Key Distribution and certification
- 7.6 Access control: firewalls
- 7.7 Attacks and counter measures
- 7.8 Security in many layers**
  - 7.8.1. Secure email**
  - 7.8.2. Secure sockets**
  - 7.8.3. IPsec**
  - 8.8.4. 802.11 WEP**

# Secure e-mail

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

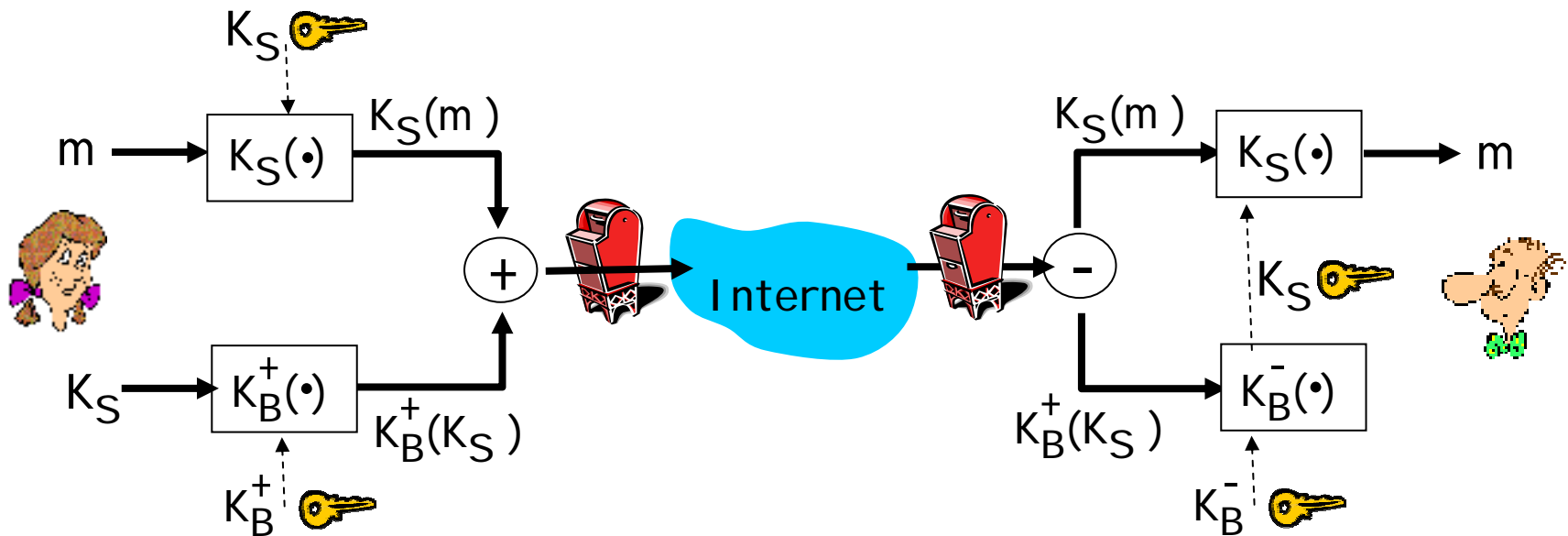


## Alice:

- 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.

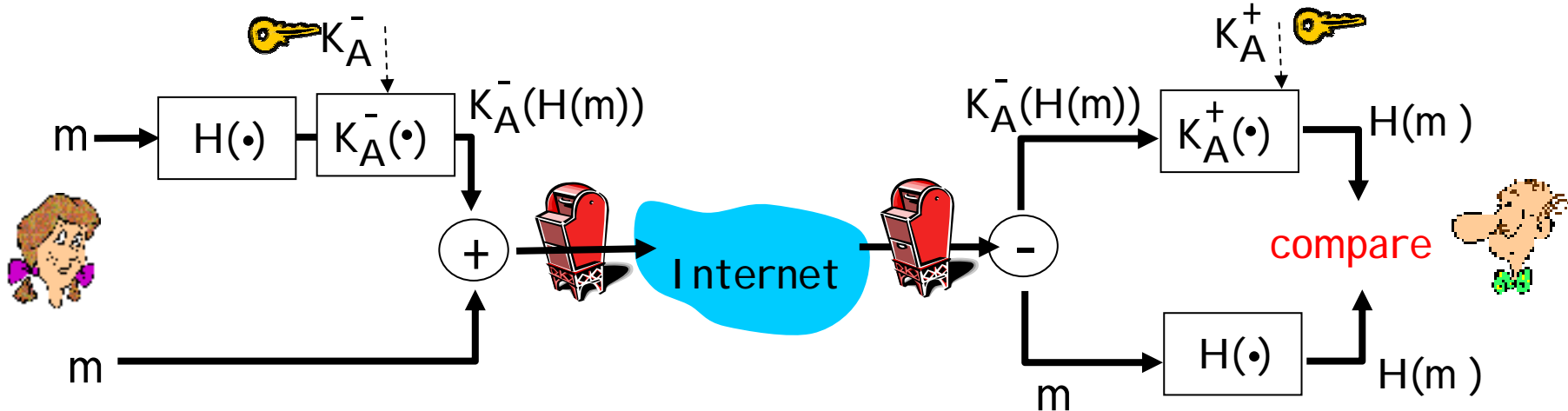


## Bob:

- 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)

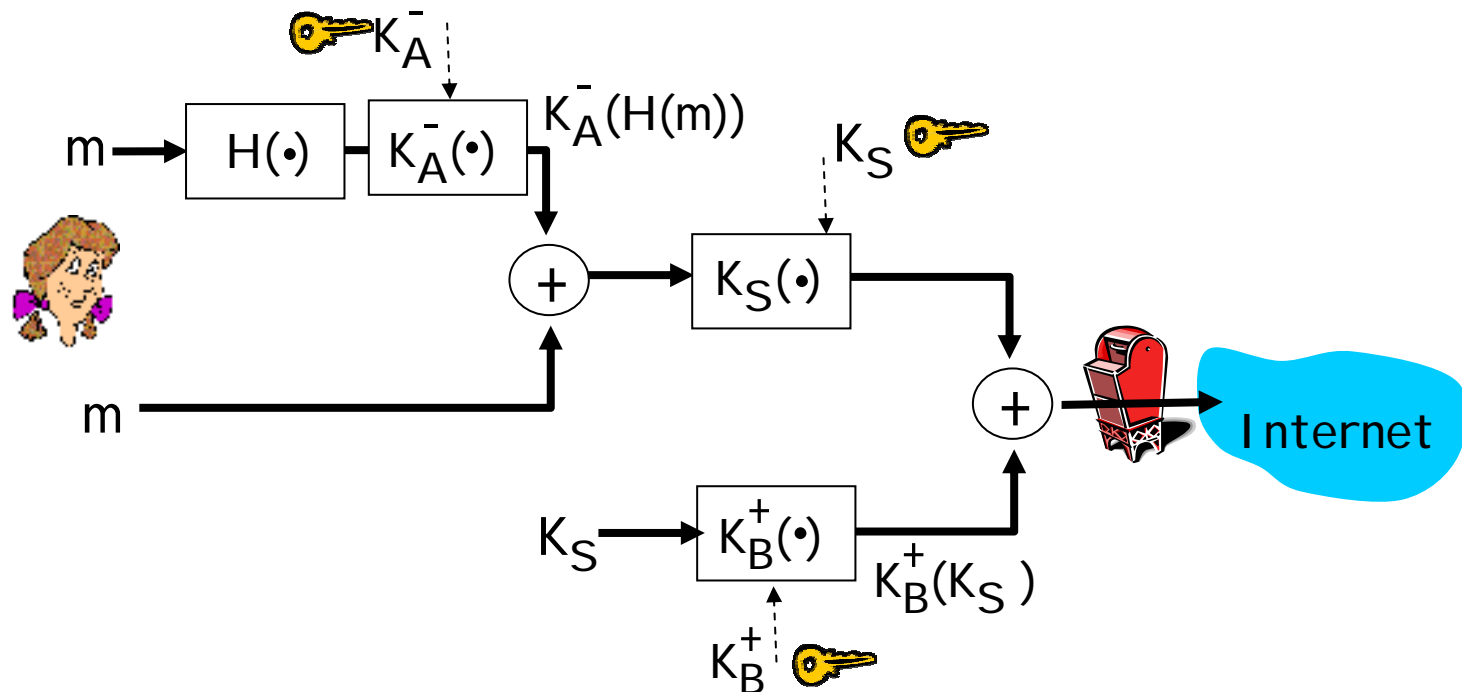
- Alice wants to provide sender authentication message integrity.



- 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

# Pretty good privacy (PGP)

- ❑ Internet e-mail encryption scheme, de-facto standard.
- ❑ uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
- ❑ provides secrecy, sender authentication, integrity.
- ❑ inventor, Phil Zimmerman, was target of 3-year federal investigation.

## A PGP signed message:

```
---BEGIN PGP SIGNED MESSAGE---  
Hash: SHA1  
  
Bob:My husband is out of town  
    tonight.Passionately yours,  
    Alice  
  
---BEGIN PGP SIGNATURE---  
Version: PGP 5.0  
Charset: noconv  
yhHJRHhGJGhgg/12EpJ+1o8gE4vB3mqJ  
    hFEvZP9t6n7G6m5Gw2  
---END PGP SIGNATURE---
```

# Secure sockets layer (SSL)

- ❑ transport layer security to any TCP-based app using SSL services.
- ❑ used between Web browsers, servers for e-commerce (shttp).
- ❑ security services:
  - server authentication
  - data encryption
  - client authentication (optional)
- ❑ server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.
- ❑ check your browser's security menu to see its trusted CAs.



# SSL (continued)

## Encrypted SSL session:

- ❑ Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
- ❑ Using private key, server decrypts session key.
- ❑ Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.
- ❑ SSL: basis of IETF Transport Layer Security (TLS).
- ❑ SSL can be used for non-Web applications, e.g., IMAP.
- ❑ Client authentication can be done with client certificates.

# IPsec: Network Layer Security

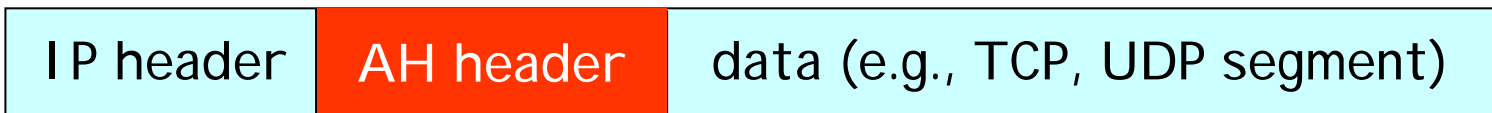
- ❑ **Network-layer secrecy:**
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages.
- ❑ **Network-layer authentication**
  - destination host can authenticate source IP address
- ❑ **Two principle protocols:**
  - authentication header (AH) protocol
  - 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

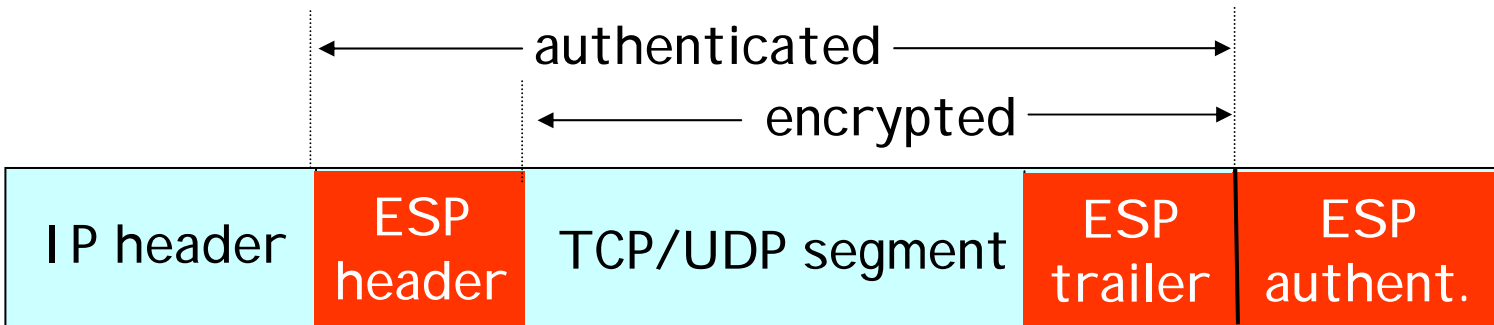
## AH header includes:

- ❑ 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.



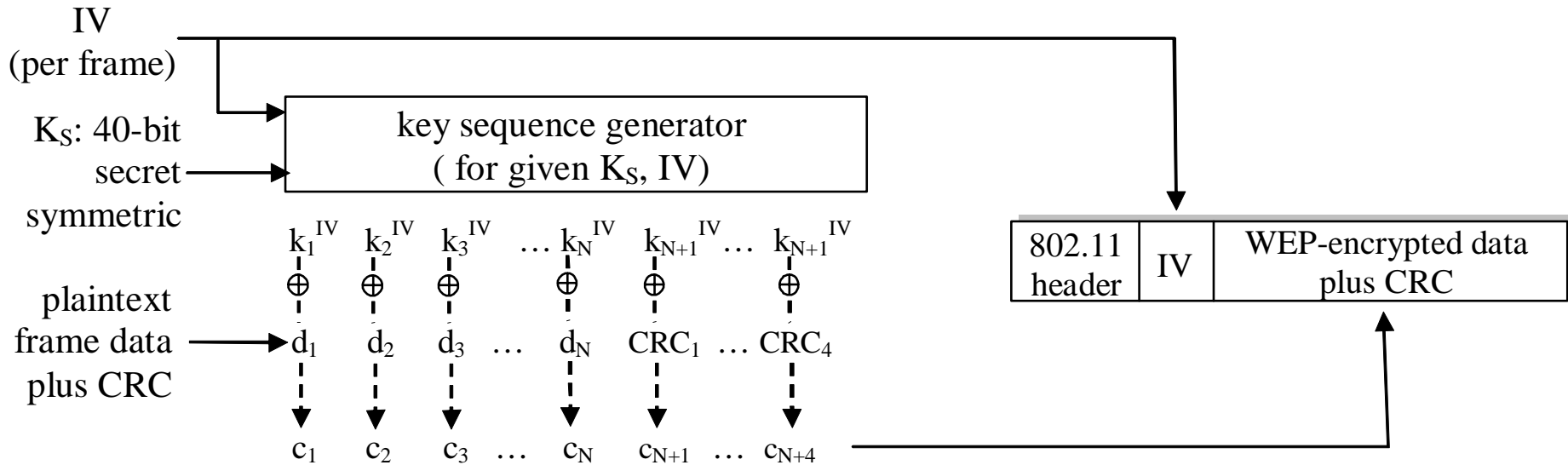
# IEEE 802.11 security

- *War-driving*: drive around Bay area, see what 802.11 networks available?
  - More than 9000 accessible from public roadways
  - 85% use no encryption/authentication
  - packet-sniffing and various attacks easy!
- *Wired Equivalent Privacy (WEP)*: authentication as in protocol *ap4.0*
  - host requests authentication from access point
  - access point sends 128 bit nonce
  - host encrypts nonce using shared symmetric key
  - access point decrypts nonce, authenticates host

# IEEE 802.11 security

- *Wired Equivalent Privacy (WEP): data encryption*
  - Host/AP share 40 bit symmetric key (semi-permanent)
  - Host appends 24-bit initialization vector (IV) to create 64-bit key
  - 64 bit key used to generate stream of keys,  $k_i^{IV}$
  - $k_i^{IV}$  used to encrypt  $i$ th byte,  $d_i$ , in frame:
$$c_i = d_i \text{ XOR } k_i^{IV}$$
  - IV and encrypted bytes,  $c_i$  sent in frame

# 802.11 WEP encryption



**Sender-side WEP encryption**

# Breaking 802.11 WEP encryption

## Security hole:

- ❑ 24-bit IV, one IV per frame, -> IV's eventually reused
- ❑ IV transmitted in plaintext -> IV reuse detected
- ❑ **Attack:**
  - Trudy causes Alice to encrypt known plaintext  $d_1 d_2 d_3 d_4 \dots$
  - Trudy sees:  $c_i = d_i \text{ XOR } k_i^{\text{IV}}$
  - Trudy knows  $c_i d_i$ , so can compute  $k_i^{\text{IV}}$
  - Trudy knows encrypting key sequence  $k_1^{\text{IV}} k_2^{\text{IV}} k_3^{\text{IV}} \dots$
  - Next time IV is used, Trudy can decrypt!



# Network Security (summary)

## Basic techniques.....

- cryptography (symmetric and public)
- authentication
- message integrity
- key distribution

## .... used in many different security scenarios

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