# Network Security - Part I

Computer Networks, Winter 2011/2012

# Chapter 7: 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

# Chapter 7 roadmap

7.1 What is network security?

7.2 Principles of cryptography

7.3 Message integrity

7.4 End point authentication

7.5 Securing e-mail

7.6 Securing TCP connections: SSL

7.7 Network layer security: IPsec

7.8 Securing wireless LANs

7.9 Operational security: firewalls and IDS

# What is network security?

Confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

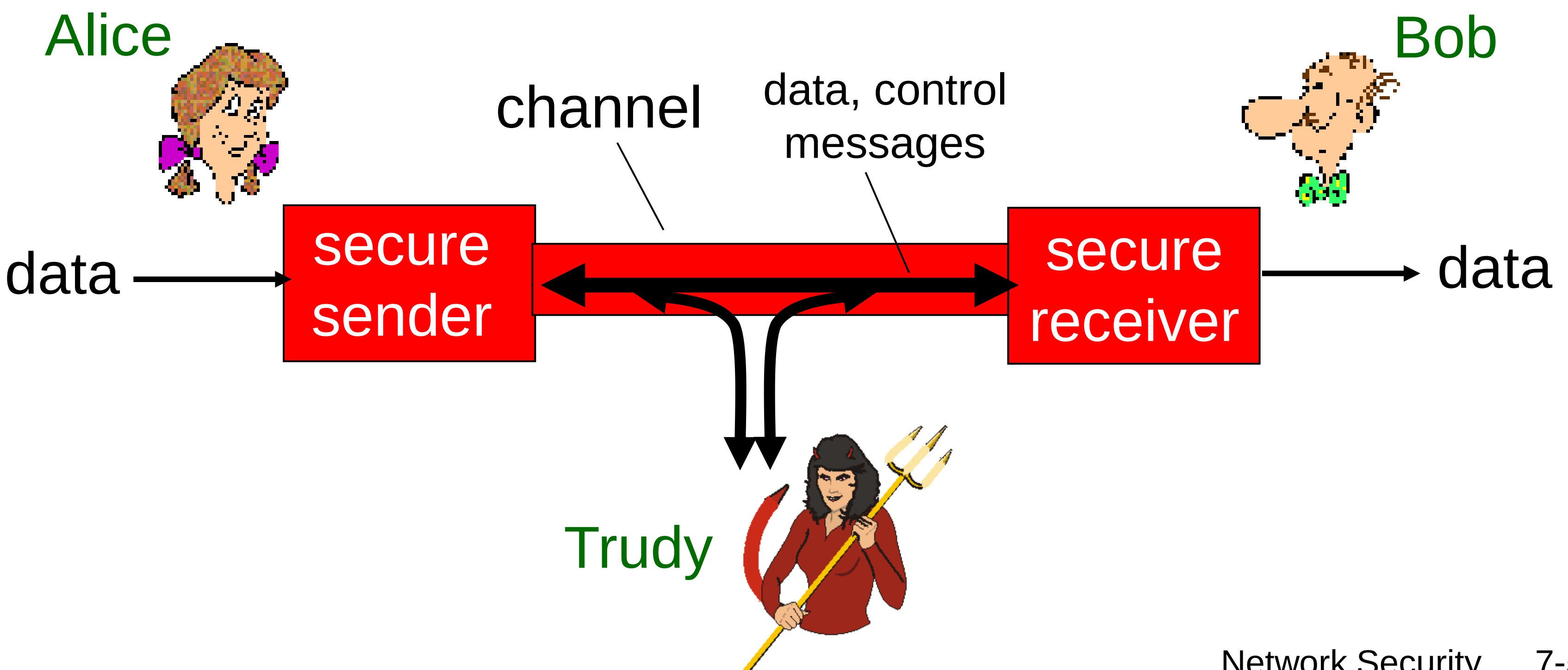
Authentication: sender, receiver want to confirm identity of each other

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?

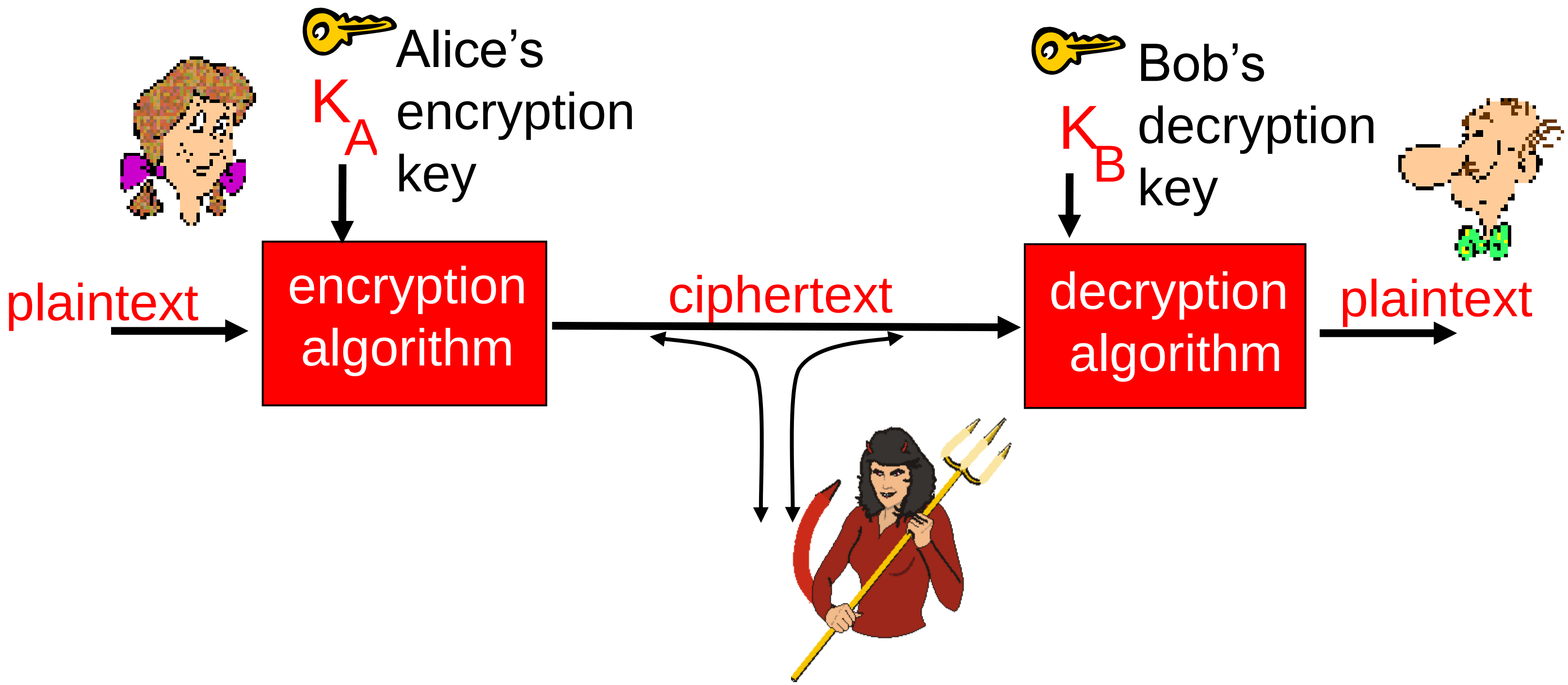
A: a lot!

- *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

# The language of cryptography




symmetric key crypto: sender, receiver keys *identical*

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

# Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
 plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:   mnbvcxzasdfghjklpoiuytrewq
```

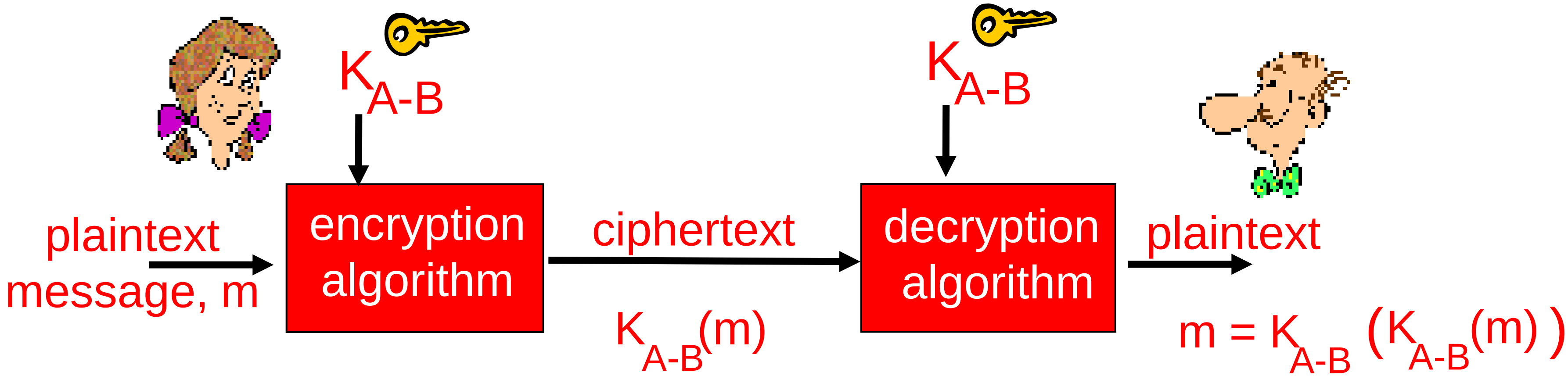
E.g.: **Plaintext: bob. i love you. alice**

**ciphertext: nkn. s gktc wky. mgsbc**

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?

# Symmetric key cryptography




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

- 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 [NIST 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

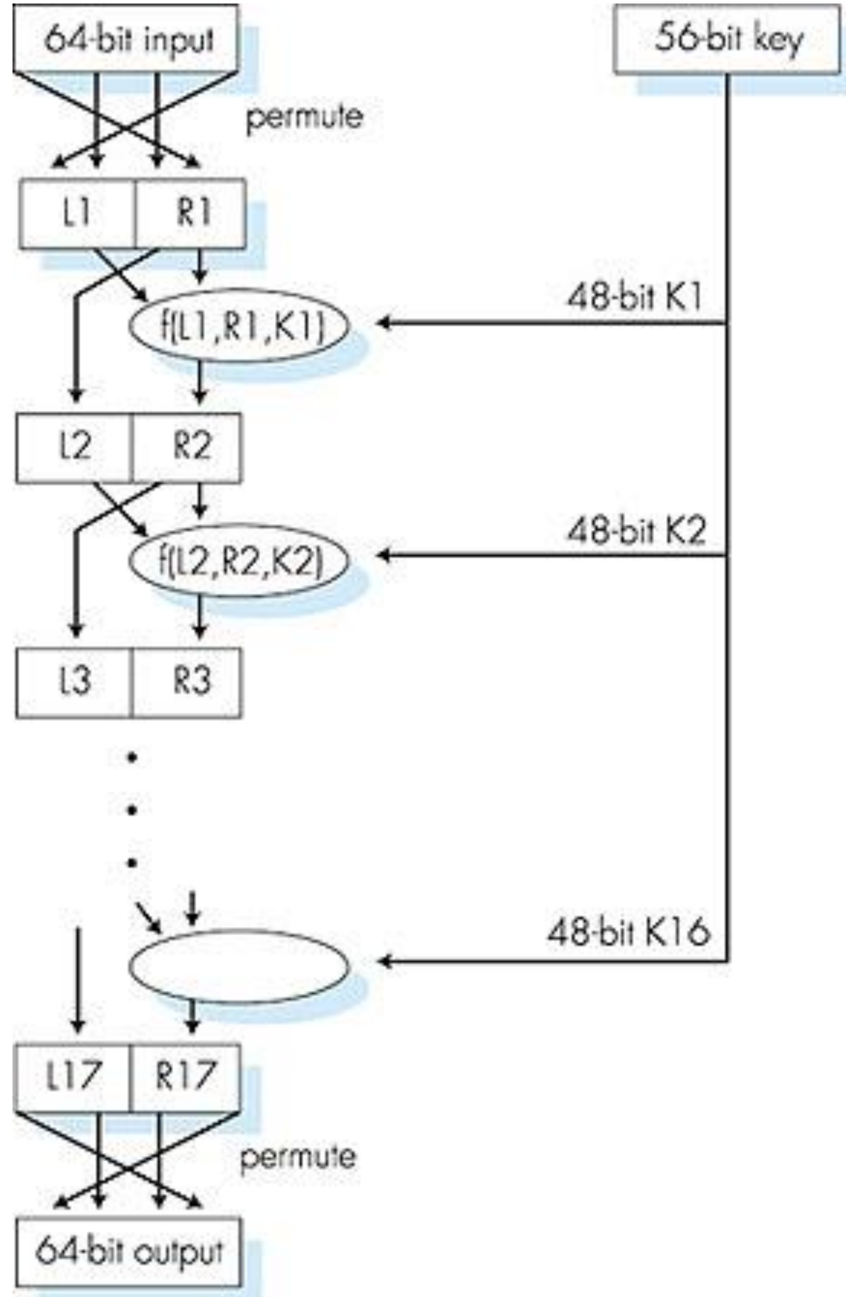
# Symmetric key crypto: DES

DES operation

initial permutation

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

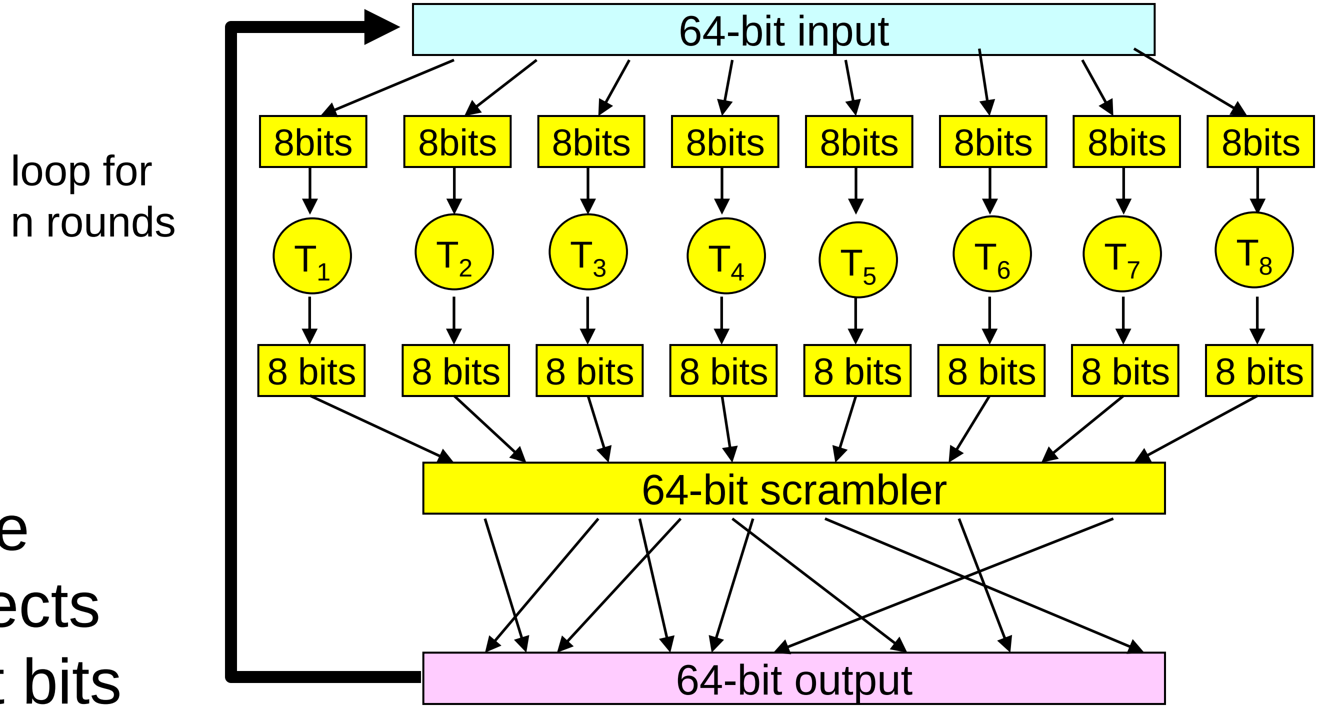
final permutation



# 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

# Block Cipher

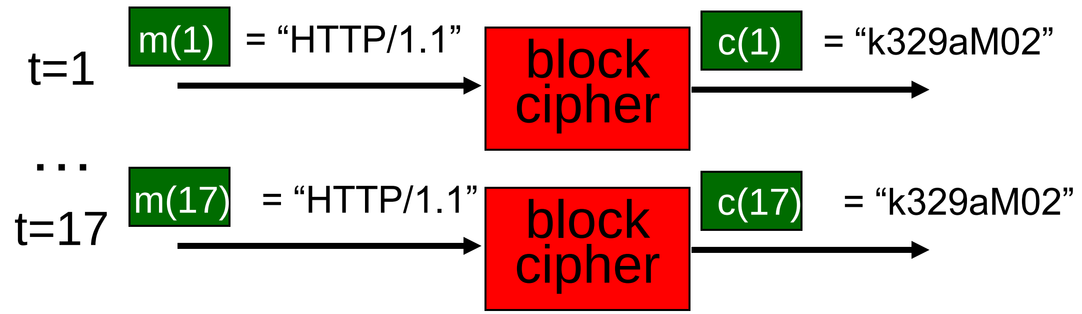



- one pass through: one input bit affects eight output bits

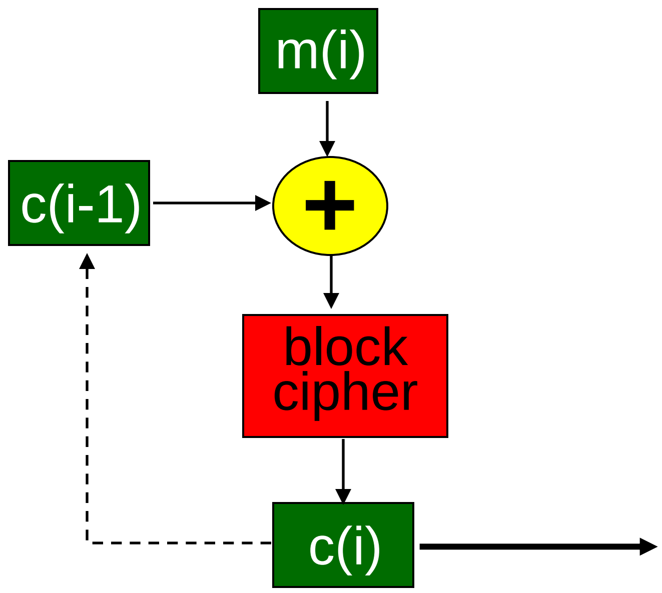
- multiple passes: each input bit afects all output bits
- block ciphers: DES, 3DES, AES

# Cipher Block Chaining

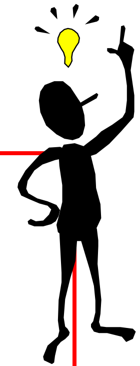
- cipher block: if input block repeated, will produce same cipher text:




- *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?



# Public key cryptography



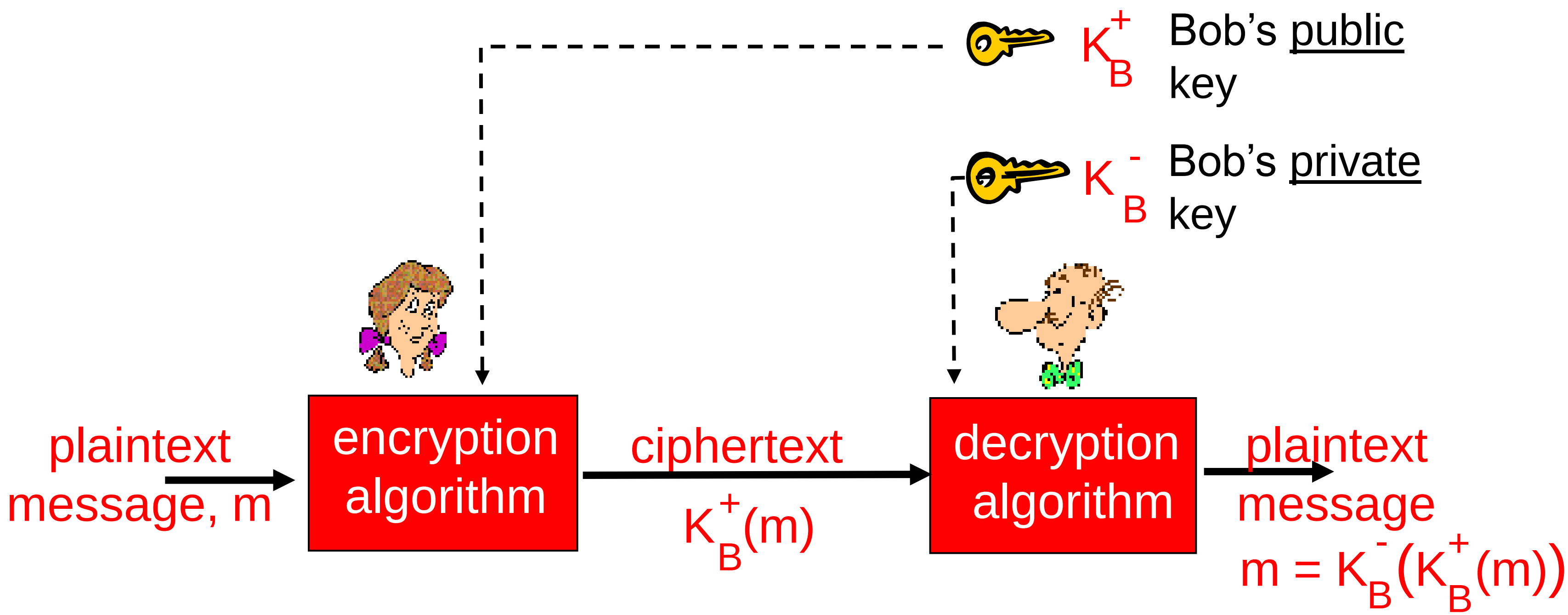
*symmetric* key crypto:

- 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, Adleman 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 $(n,d)$.

$\underbrace{(n,e)}_{K_B^+}$ $\underbrace{(n,d)}_{K_B^-}$

# 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.

| | letter | m | $m^e$ | $c = m^e \bmod n$ |
|---|---|---|---|---|
| encrypt: | l | 12 | 1524832 | 17 |

| | c | $c^d$ | $m = c^d \bmod n$ | letter |
|---|---|---|---|---|
| decrypt: | 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 later:

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

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 Message integrity

7.4 End point authentication

7.5 Securing e-mail

7.6 Securing TCP connections: SSL

7.7 Network layer security: IPsec

7.8 Securing wireless LANs

7.9 Operational security: firewalls and IDS

# Message Integrity

Bob receives msg from Alice, wants to ensure:

- message originally came from Alice
- message not changed since sent by Alice

## Cryptographic Hash:

- takes input m, produces fixed length value, H(m)
  - e.g., as in Internet checksum
- computationally infeasible to find two different messages, x, y such that H(x) = H(y)
  - equivalently: given m = H(x), (x unknown), can not determine x.
  - note: Internet checksum *fails* this requirement!

# 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

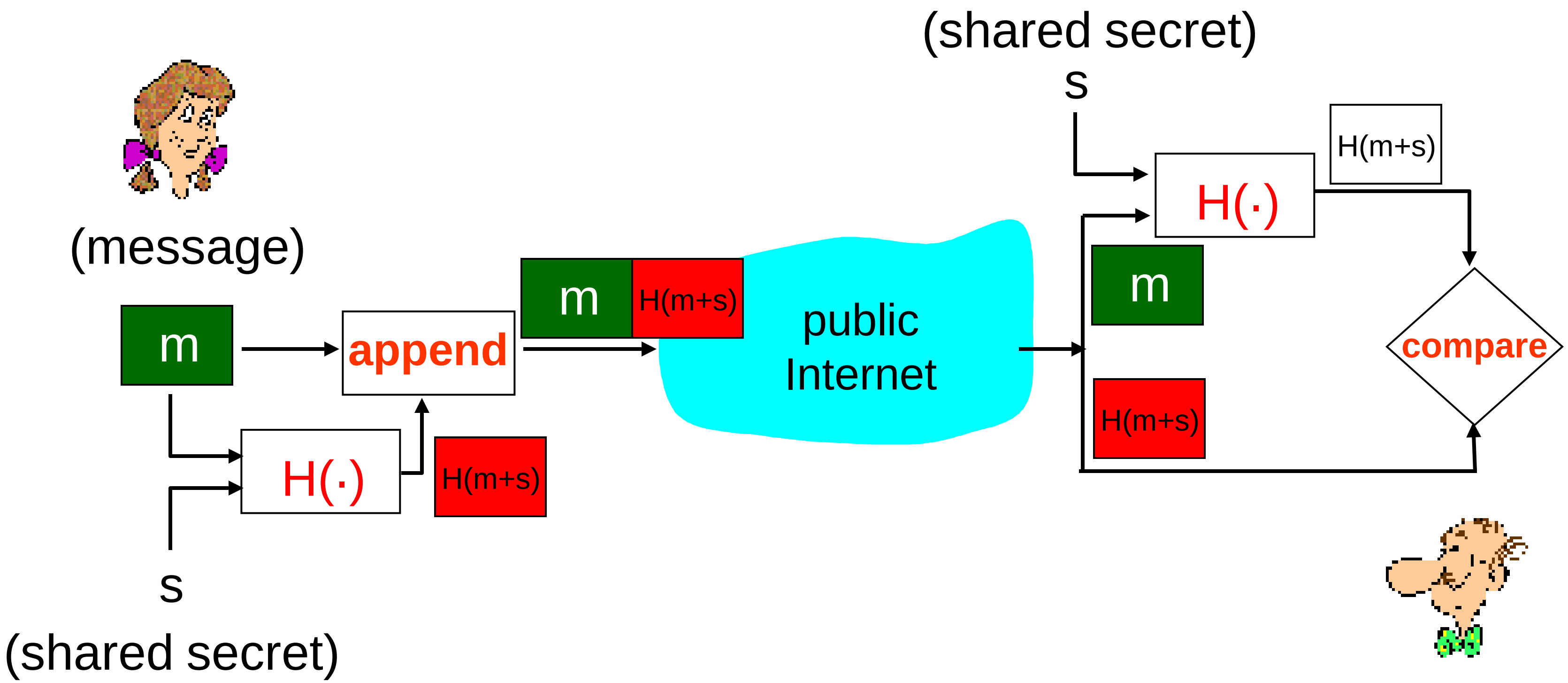
But given message with given hash value, it is *easy* to find another message with same hash value:

| message | ASCII format |
|---|---|
| I O U 1 | 49 4F 55 31 |
| 0 0 . 9 | 30 30 2E 39 |
| 9 B O B | 39 42 4F 42 |
| | B2 C1 D2 AC |

| message | ASCII format |
|---|---|
| I O U 9 | 49 4F 55 39 |
| 0 0 . 1 | 30 30 2E 31 |
| 9 B O B | 39 42 4F 42 |
| | B2 C1 D2 AC |

different messages but identical checksums!

# Message Authentication Code



# MACs in practice

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit MAC in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
    - recent (2005) attacks on MD5
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit MAC

# Digital Signatures

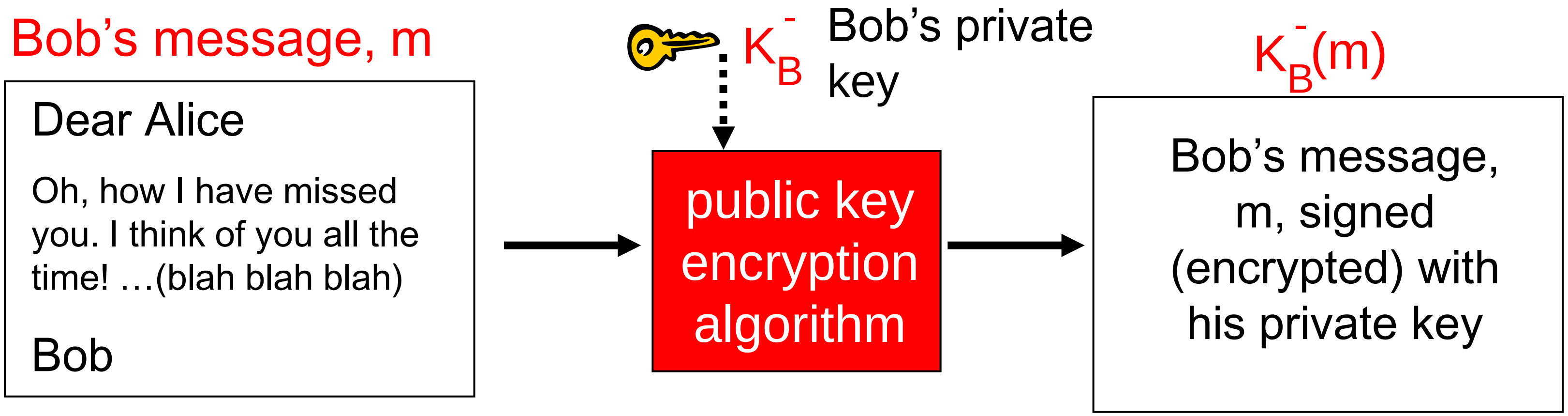
cryptographic technique analogous to hand-written signatures.

- 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, $K_B^-(m)$



# 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.

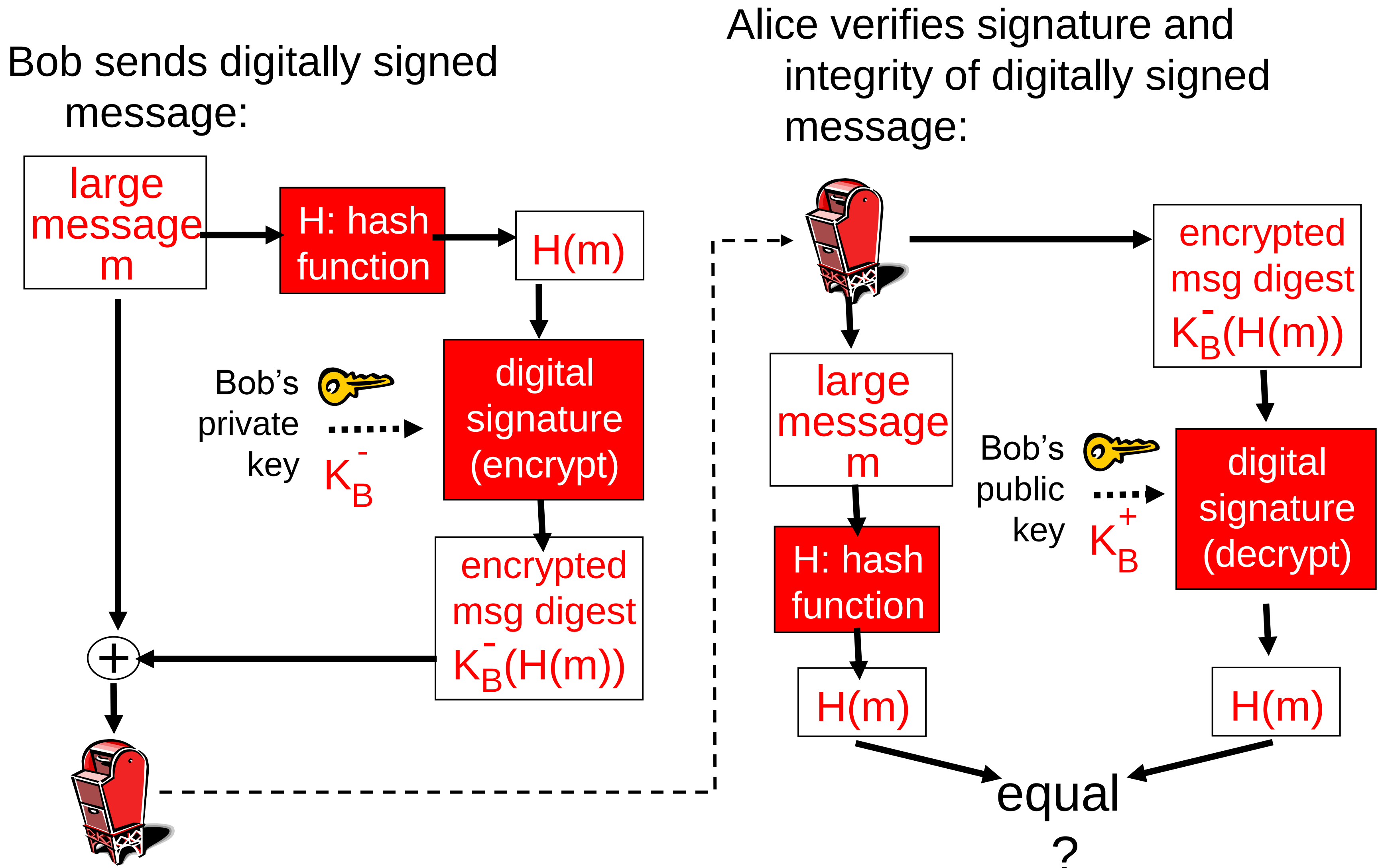
Alice thus verifies that:

- ✓ 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.

# Digital signature = signed MAC



# Public Key Certification

## public key problem:

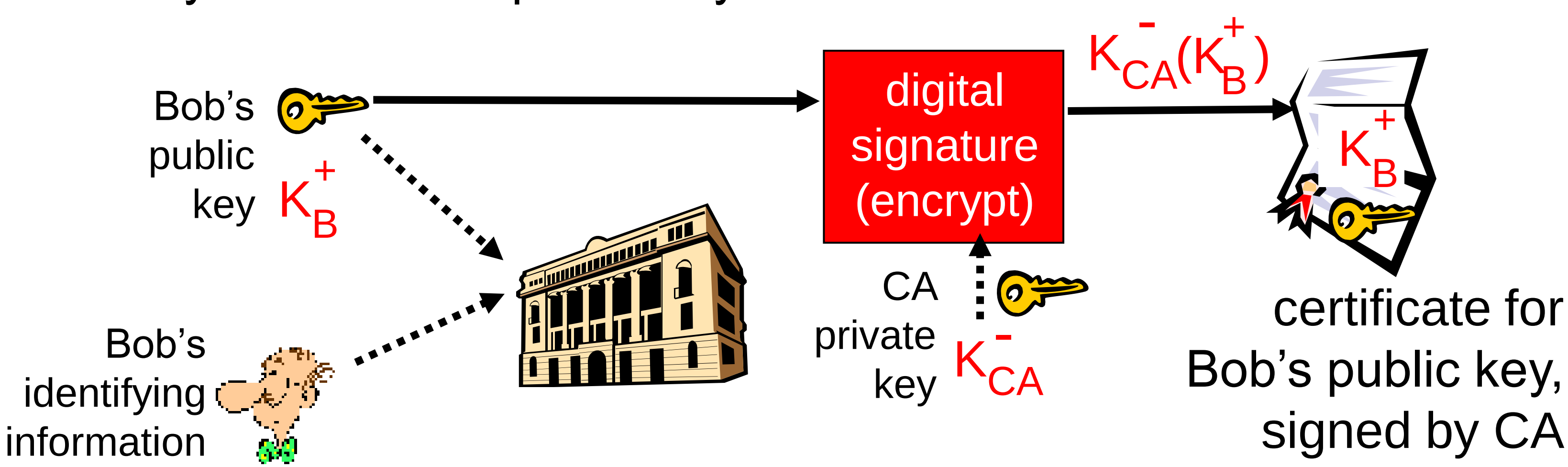
- When Alice obtains Bob's public key (from web site, e-mail, disk), how does she *know* it is Bob's public key, not Trudy's?

## solution:

- trusted certification authority (CA)

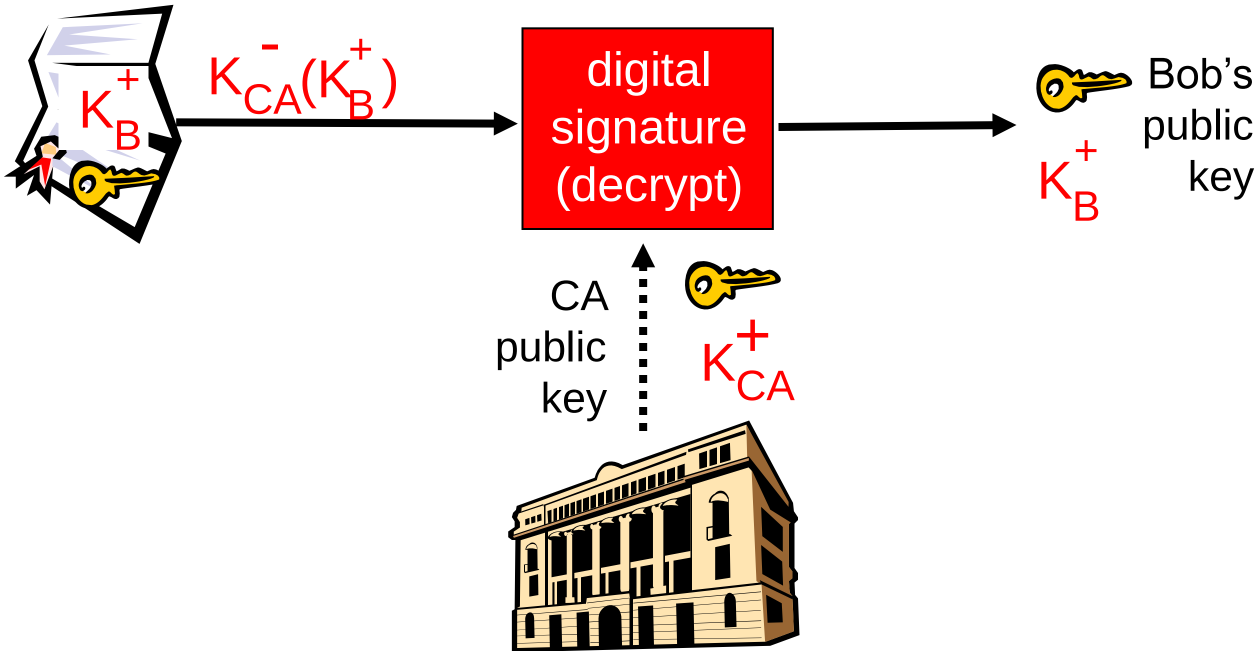
# Certification Authorities

- Certification Authority (CA): binds public key to particular entity, E.
- E 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."



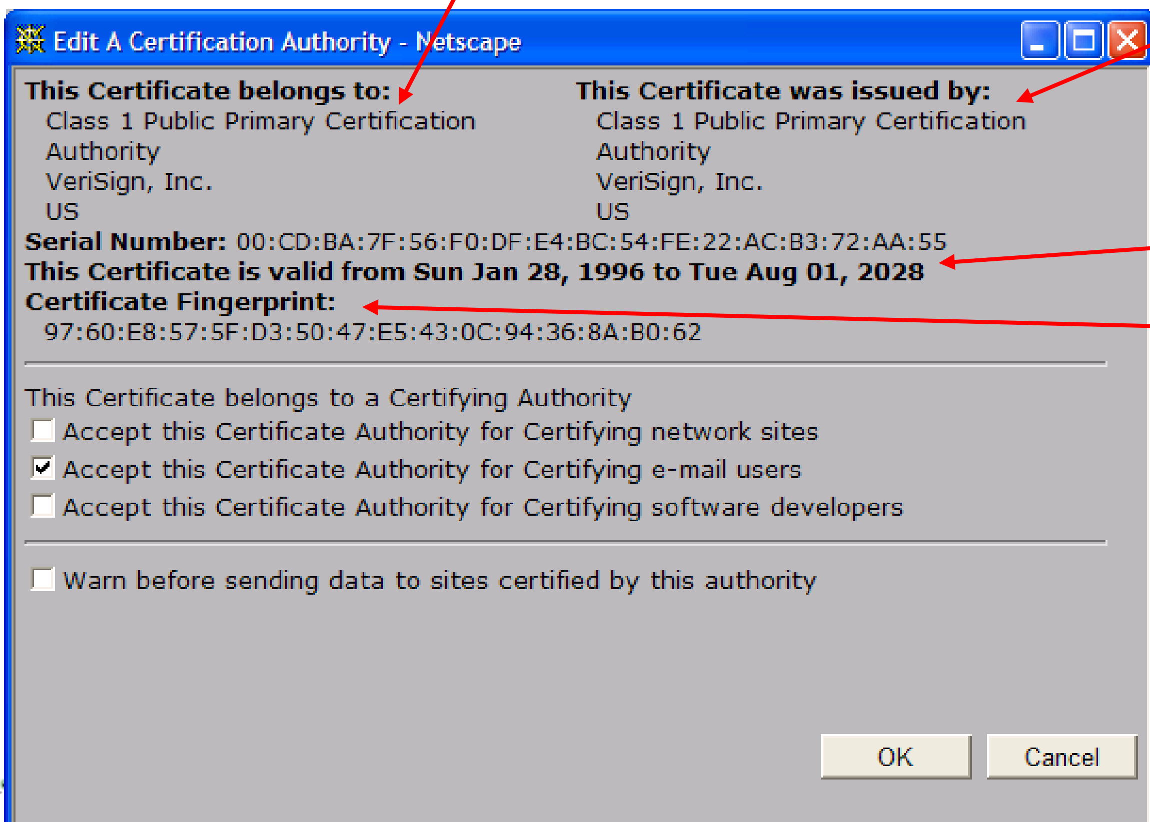
# Certification Authorities

- 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 Message integrity

7.4 End point authentication

7.5 Securing e-mail

7.6 Securing TCP connections: SSL

7.7 Network layer security: IPsec

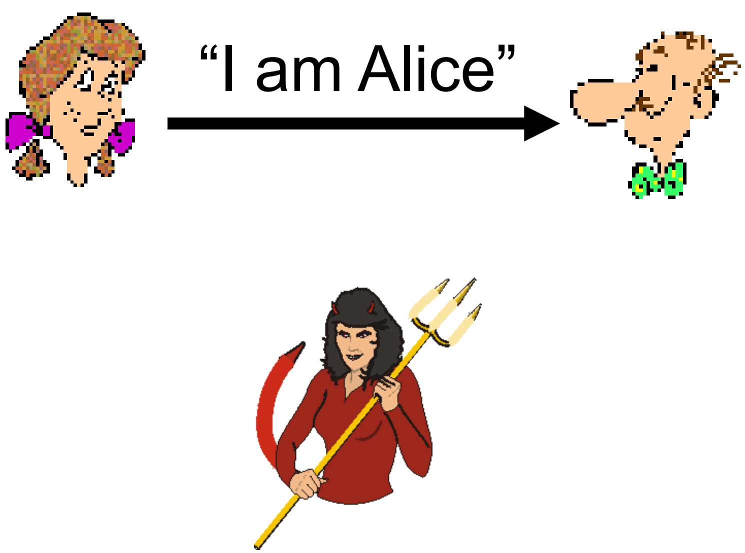
7.8 Securing wireless LANs

7.9 Operational security: firewalls and IDS

# Authentication

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

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

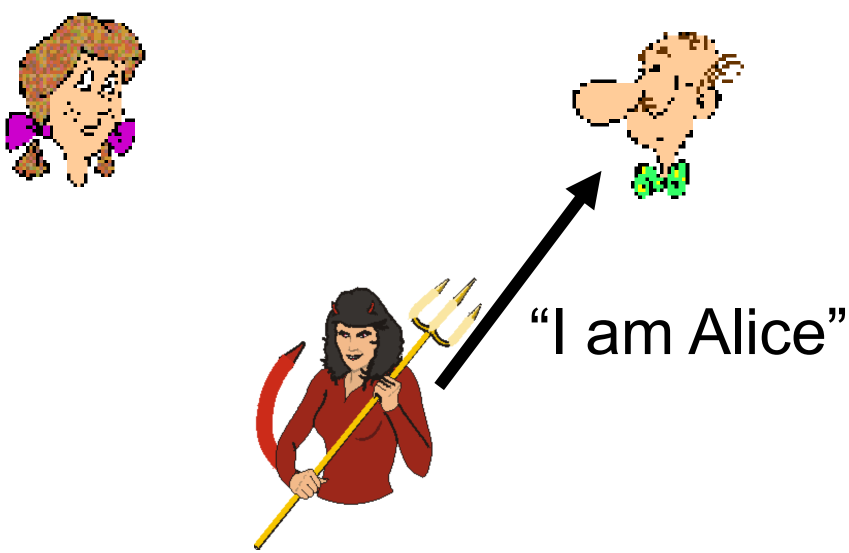



Failure scenario??

# Authentication

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

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




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

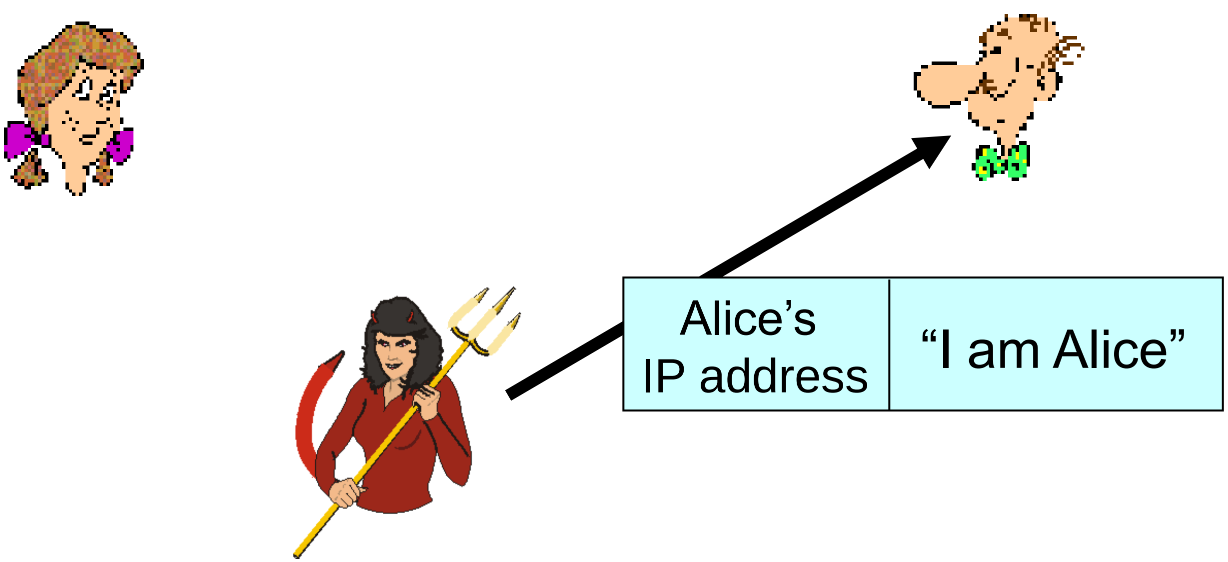
# Authentication: another try

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


Failure scenario??

# Authentication: another try

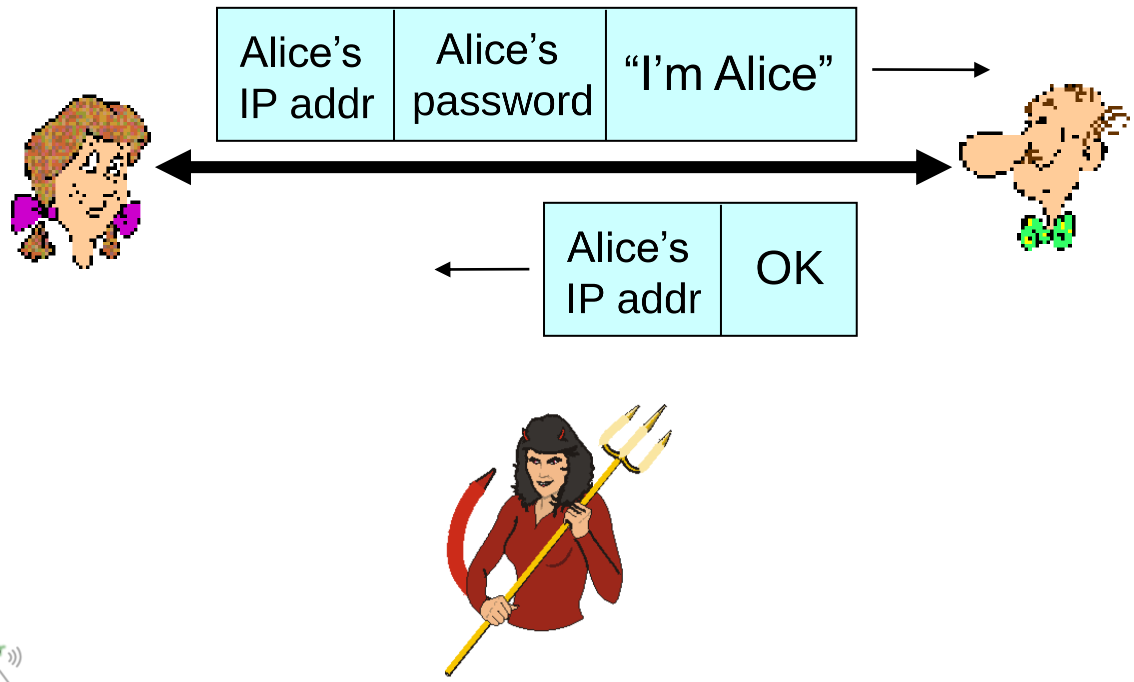
<u>Protocol ap2.0:</u> Alice says “I am Alice” in an IP packet containing her source IP address




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

# Authentication: another try

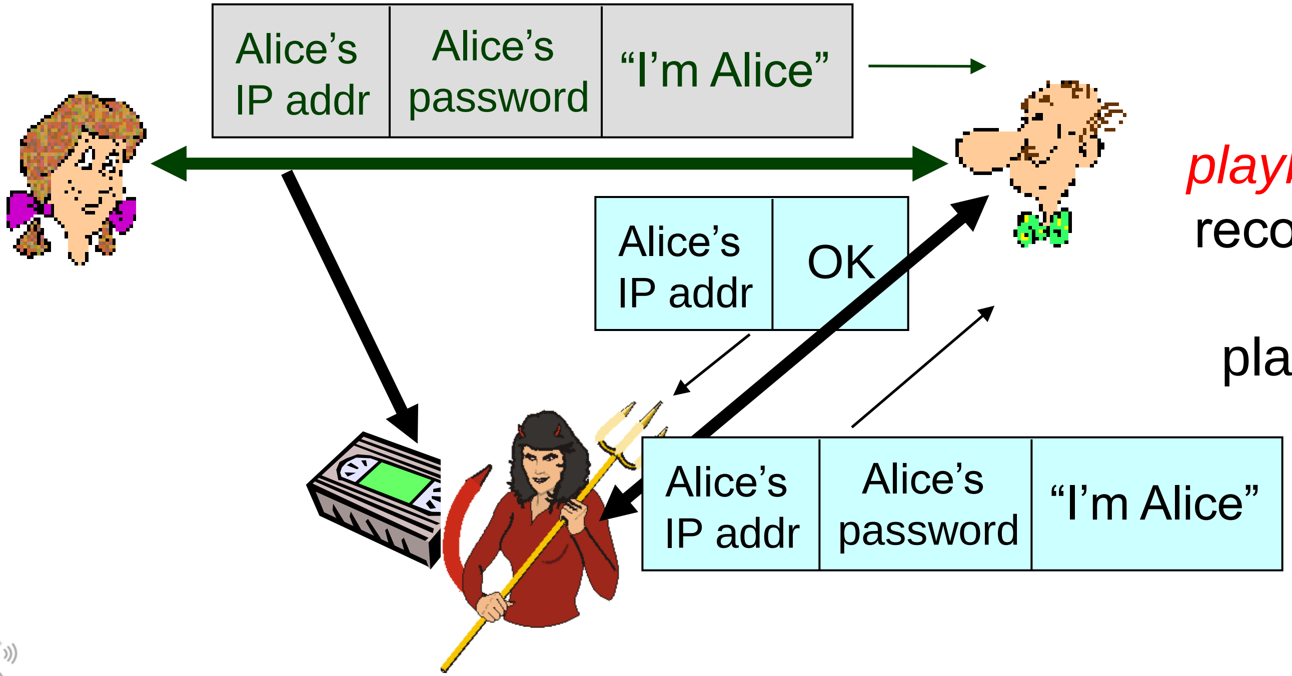
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.




Failure scenario??

# Authentication: another try

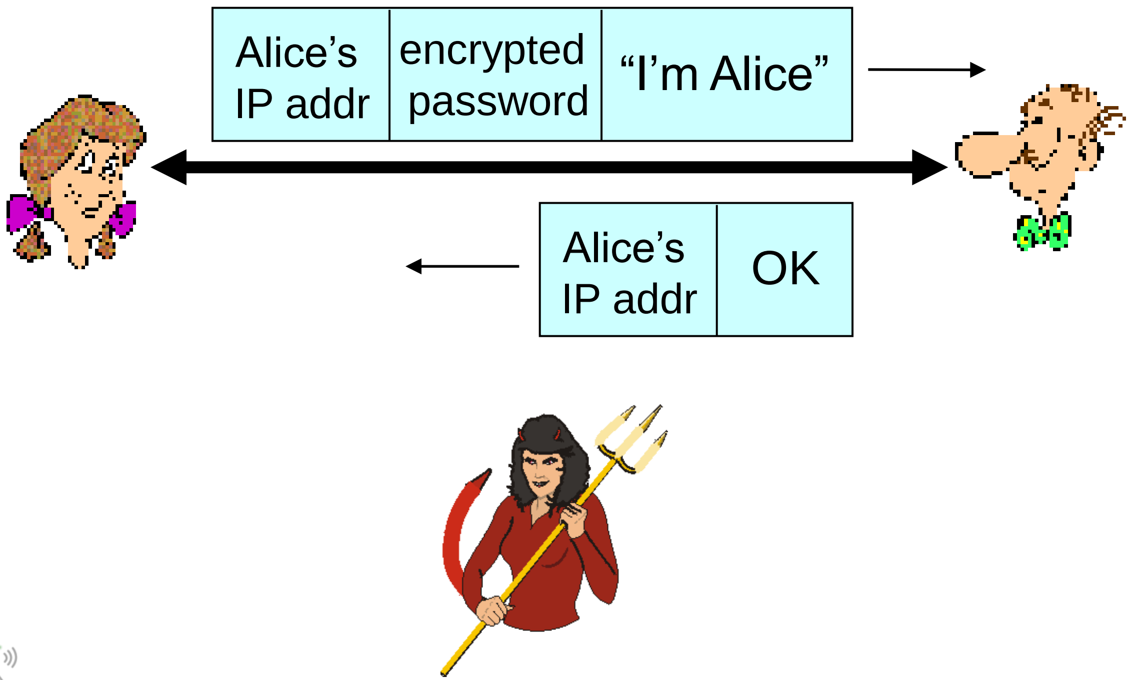
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.




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

# Authentication: yet another try

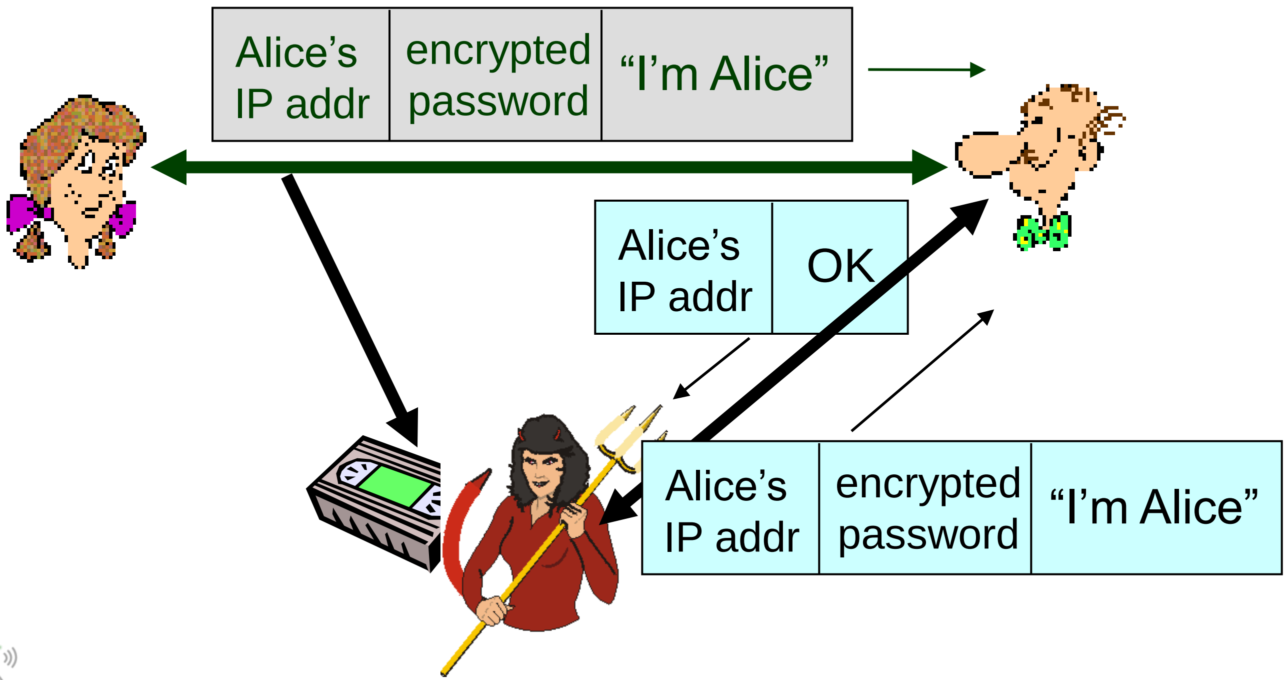
<u>Protocol ap3.1:</u> Alice says “I am Alice” and sends her *encrypted* secret password to “prove” it.




Failure scenario??

# Authentication: another try

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



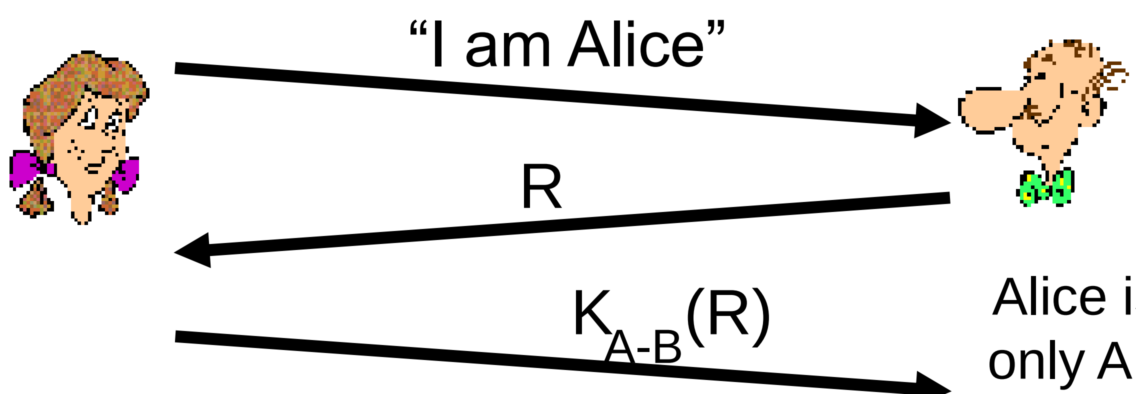

record and playback still works!

# 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 a 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!

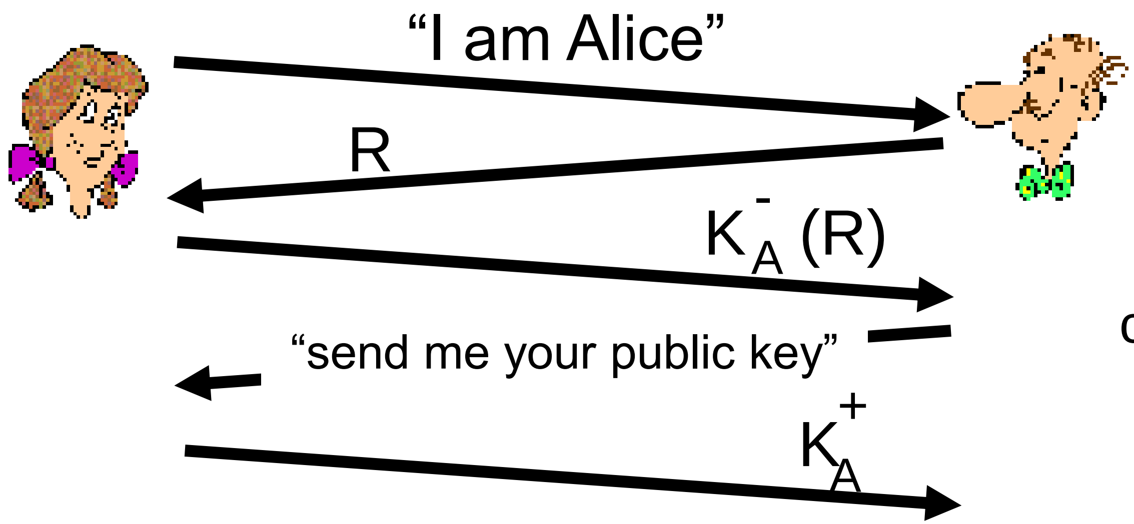
Failures, drawbacks?

# 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

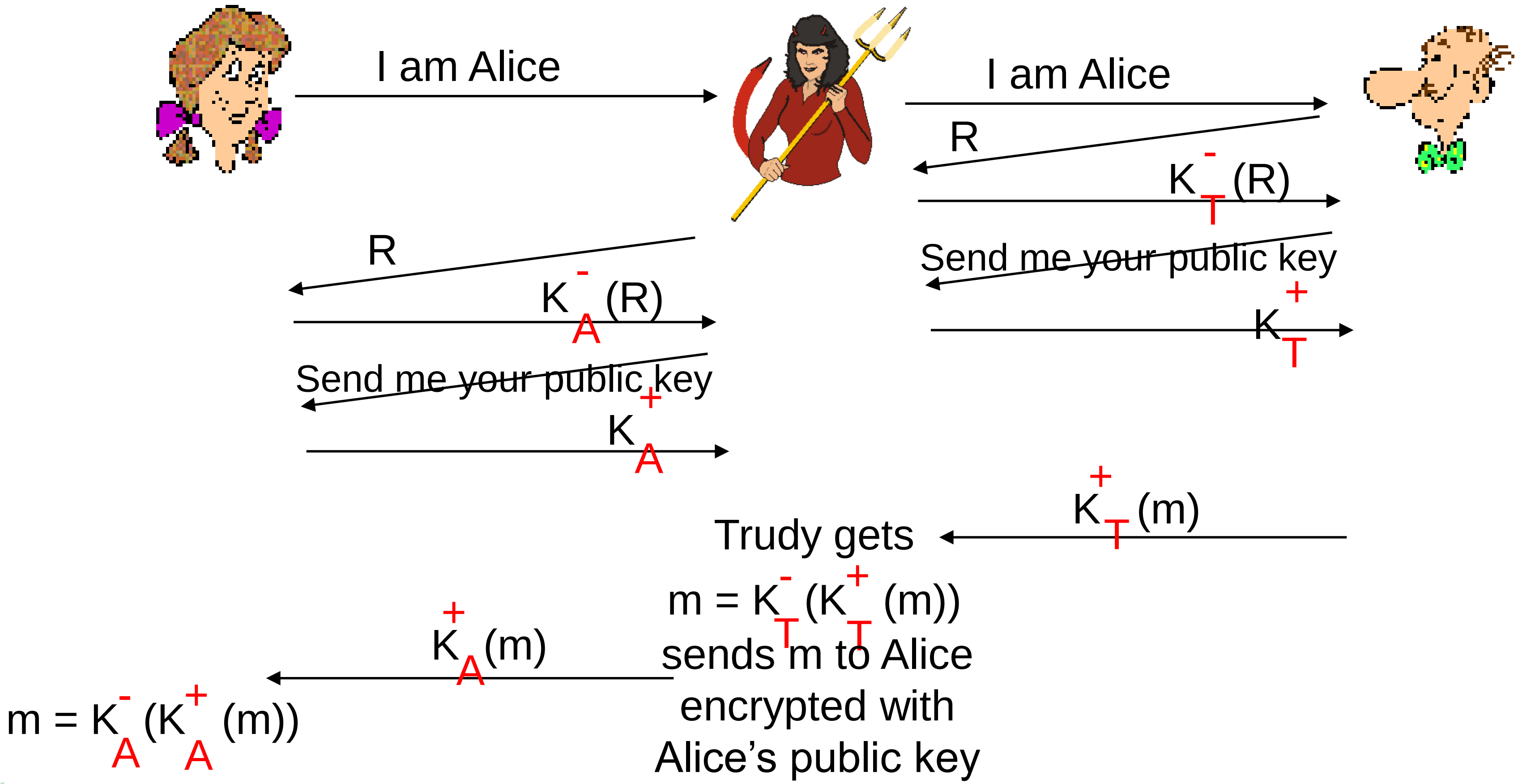
$$K_A^+(K_A^-(R)) = R$$

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

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

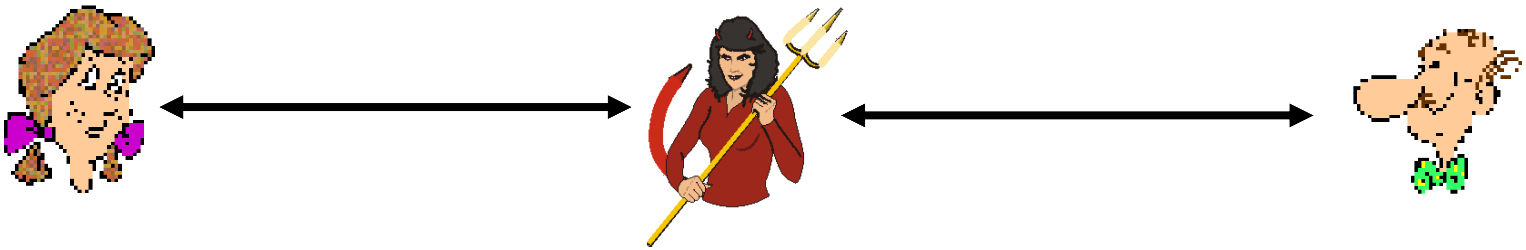
# ap5.0: security hole

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



# 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!