# Chapter 8
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 see the animations; and 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) that you mention their source
  (after all, we'd like people to use our book!)
❖ If you post any slides 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
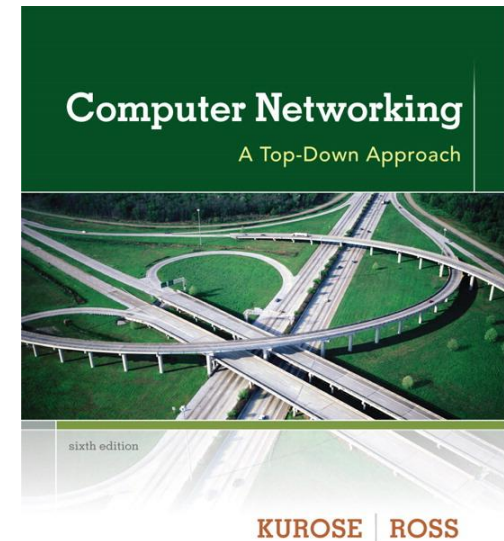
**Computer
Networking: A Top
Down Approach**
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

What is network security?

confidentiality: only sender, intended receiver should "understand" message contents
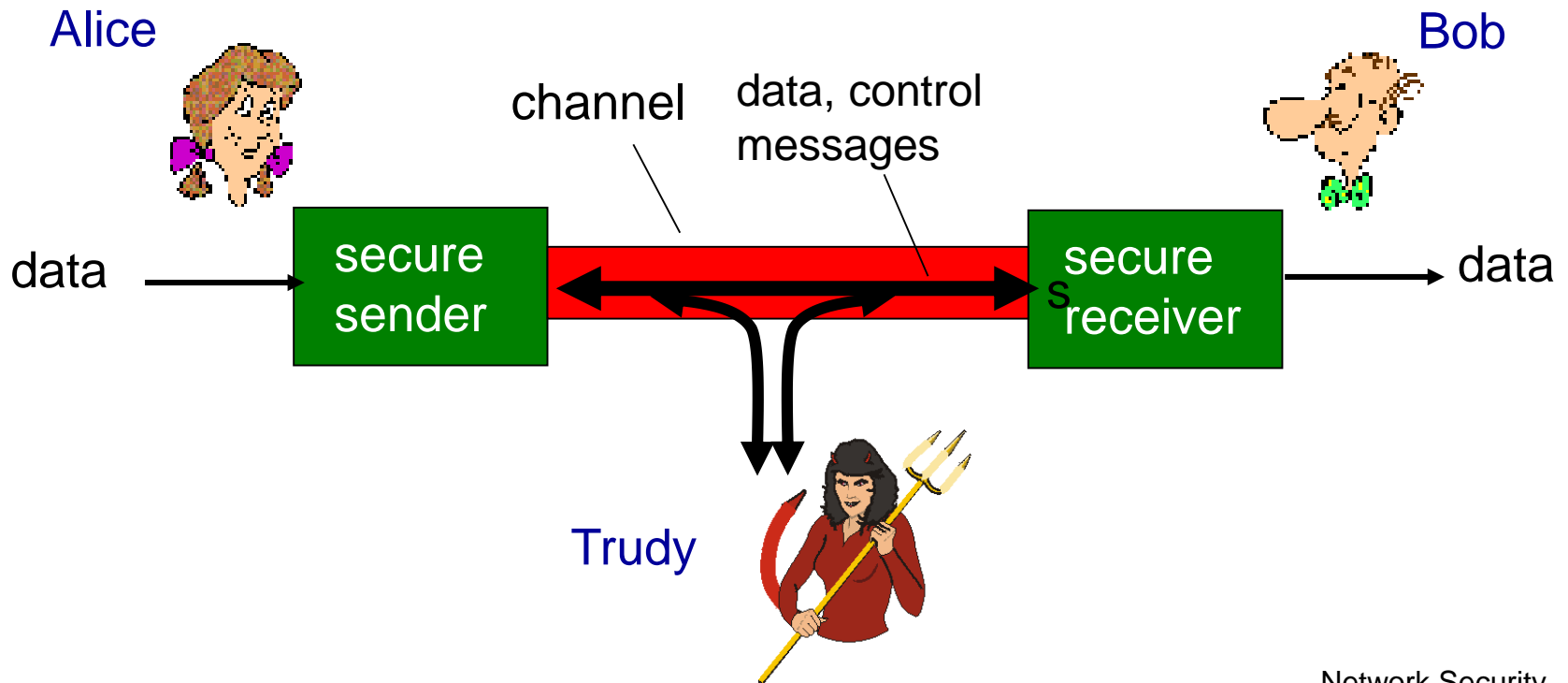- sender encrypts message
- receiver decrypts message

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

Alice

Bob

channel   data, control
                 messages

data → | secure
         sender | ←→ | secure
                      receiver | → data

Trudy

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)

Breaking an encryption scheme

❖ **cipher-text only attack:** Trudy has ciphertext she can analyze

❖ **two approaches:**
  - brute force: search through all keys
  - statistical analysis

❖ **known-plaintext attack:** Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,

❖ **chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext
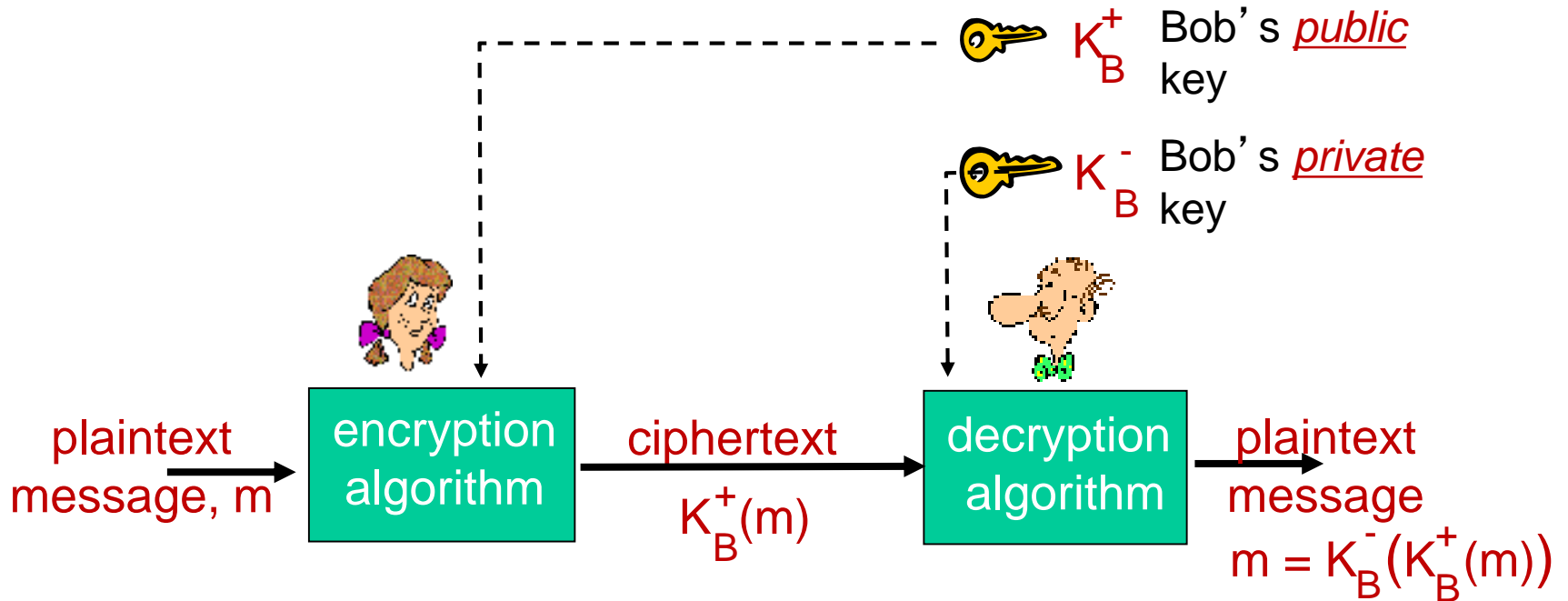
Public Key Cryptography

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 crypto

- ❖ 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

K_B^+ Bob's *public* key

K_B^- Bob's *private* key

plaintext message, m → encryption algorithm → ciphertext $K_B^+(m)$ → decryption algorithm → plaintext message $m = K_B^-(K_B^+(m))$

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 mod n = remainder of x when divide by n

❖ facts:

[(a mod n) + (b mod n)] mod n = (a+b) mod n

[(a mod n) - (b mod n)] mod n = (a-b) mod n

[(a mod n) * (b mod n)] mod n = (a*b) mod 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

❖ message: just a bit pattern

❖ 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 ciphertext).

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).

   $$K_B^+ \qquad\qquad\qquad K_B^-$$

RSA: encryption, decryption

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

1. to encrypt message m ($<n$), compute

   $c = m^e \mod n$

2. to decrypt received bit pattern, c, compute

   $m = c^d \mod n$

magic happens! $\quad m = (\underbrace{m^e \mod n}_{c})^d \mod n$
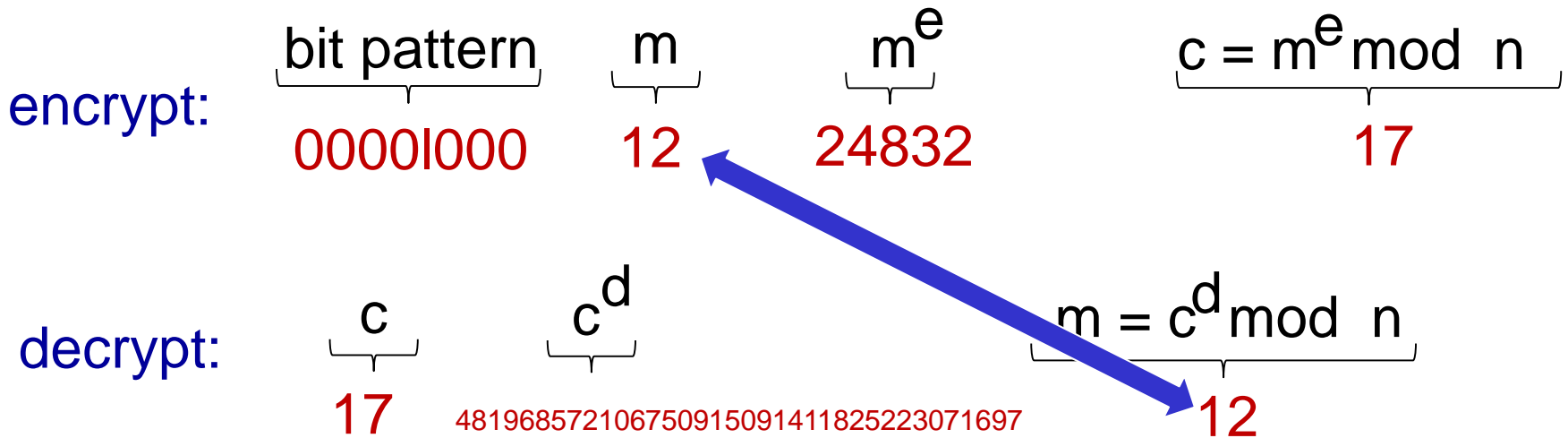
RSA example:

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:

bit pattern	m	m^e	$c = m^e \bmod n$
0000l000	12	24832	17

decrypt:

c	c^d	$m = c^d \bmod n$
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,

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

$\qquad = m^{ed} \bmod n$

$\qquad = m^{(ed \bmod z)} \bmod n$

$\qquad = m^1 \bmod n$

$\qquad = m$

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!

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

follows directly from modular arithmetic:

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

$$= m^{de} \bmod n$$

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

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

Chapter 8 roadmap

Authentication

Goal: Bob wants Alice to "prove" her identity to him
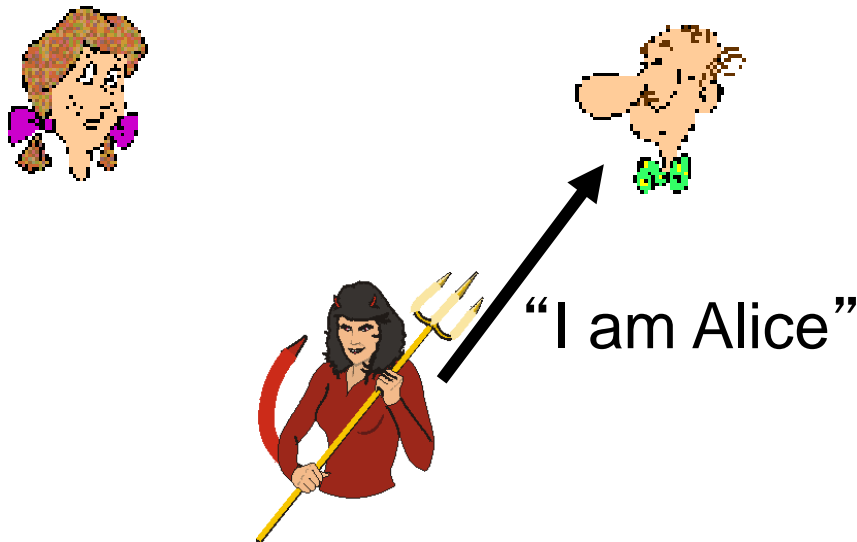
Protocol ap1.0: Alice says "I am Alice"

"I am Alice"

Failure scenario??

Authentication

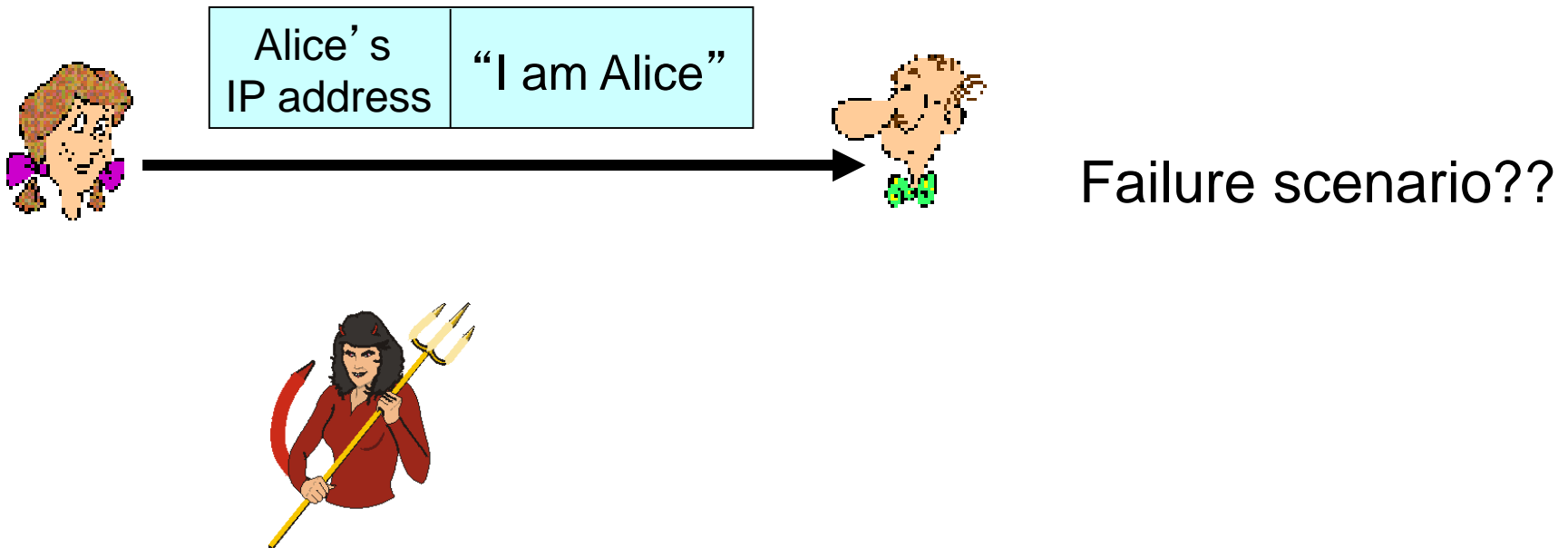*Goal:* Bob wants Alice to "prove" her identity to him

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
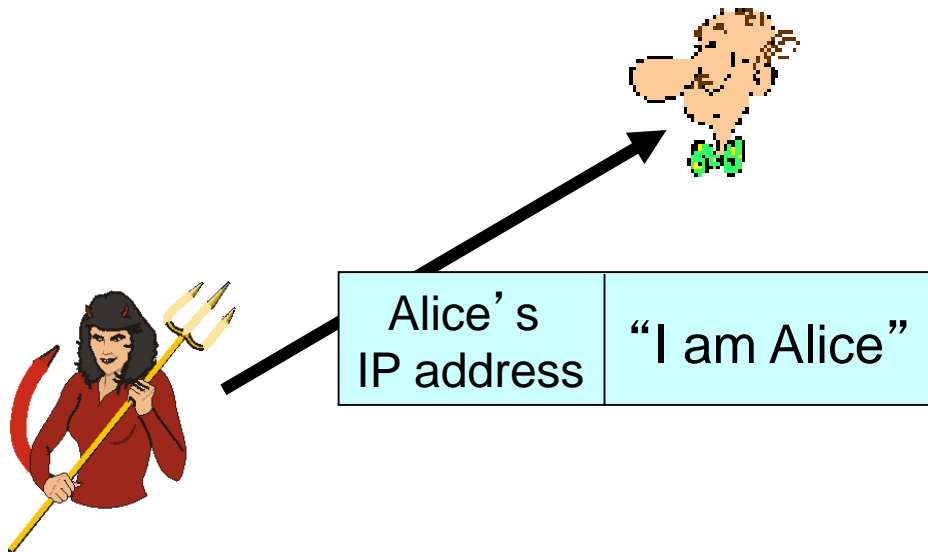
Authentication: another try

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

Alice's IP address | "I am Alice"

Failure scenario??

Authentication: another try

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

Alice's IP address | "I am Alice"

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.

| Alice's IP addr | Alice's password | "I'm Alice" | → |

| ← | Alice's IP addr | OK |

Failure scenario??

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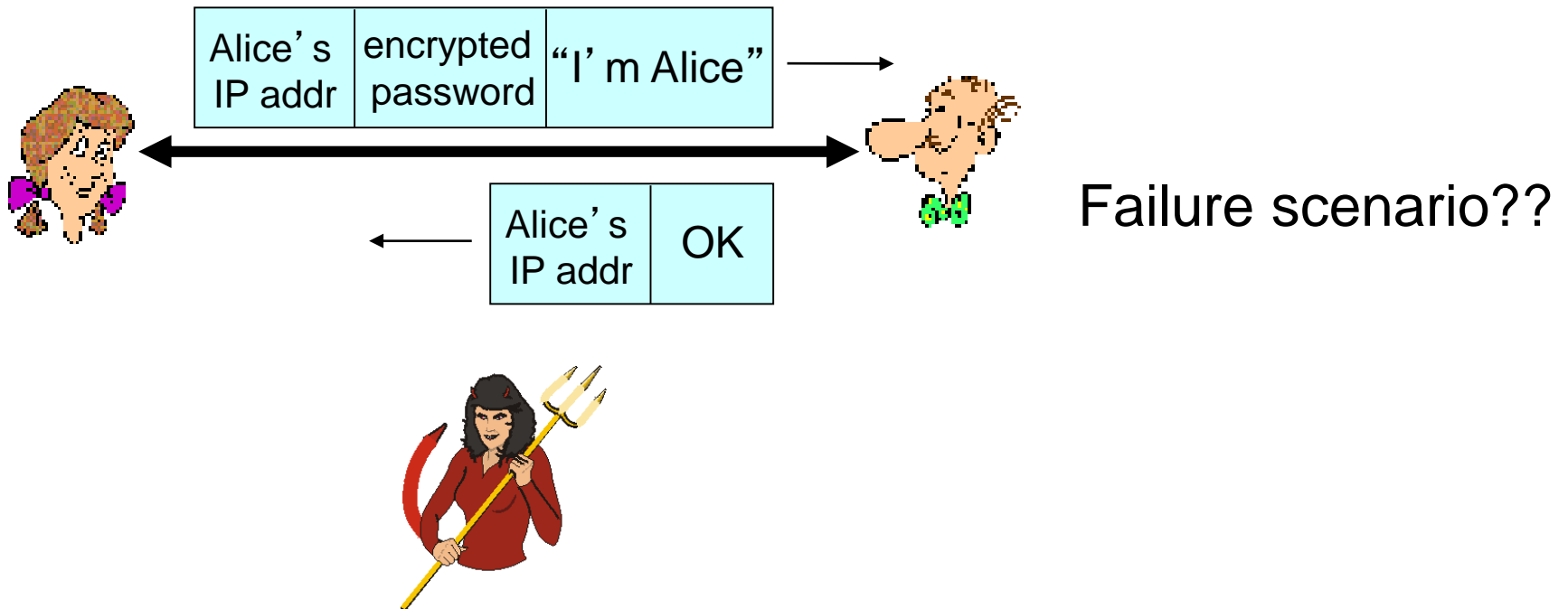
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

Alice's IP addr	Alice's password	"I'm Alice"

Alice's IP addr	OK

Alice's IP addr	Alice's password	"I'm Alice"

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

Authentication: yet another try

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

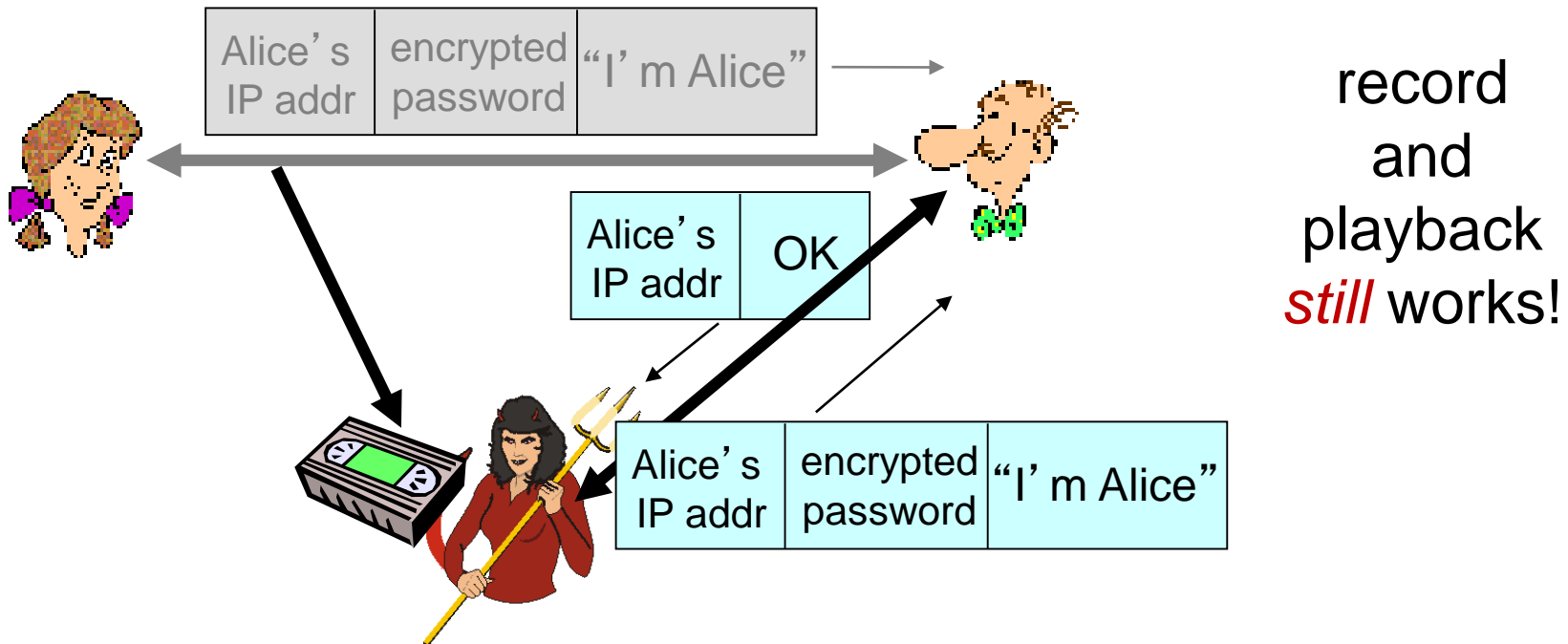| Alice's IP addr	encrypted password	"I'm Alice"

Alice's IP addr	OK

Failure scenario??

Authentication: yet another try

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

Alice's IP addr	encrypted password	"I'm Alice"

Alice's IP addr	OK

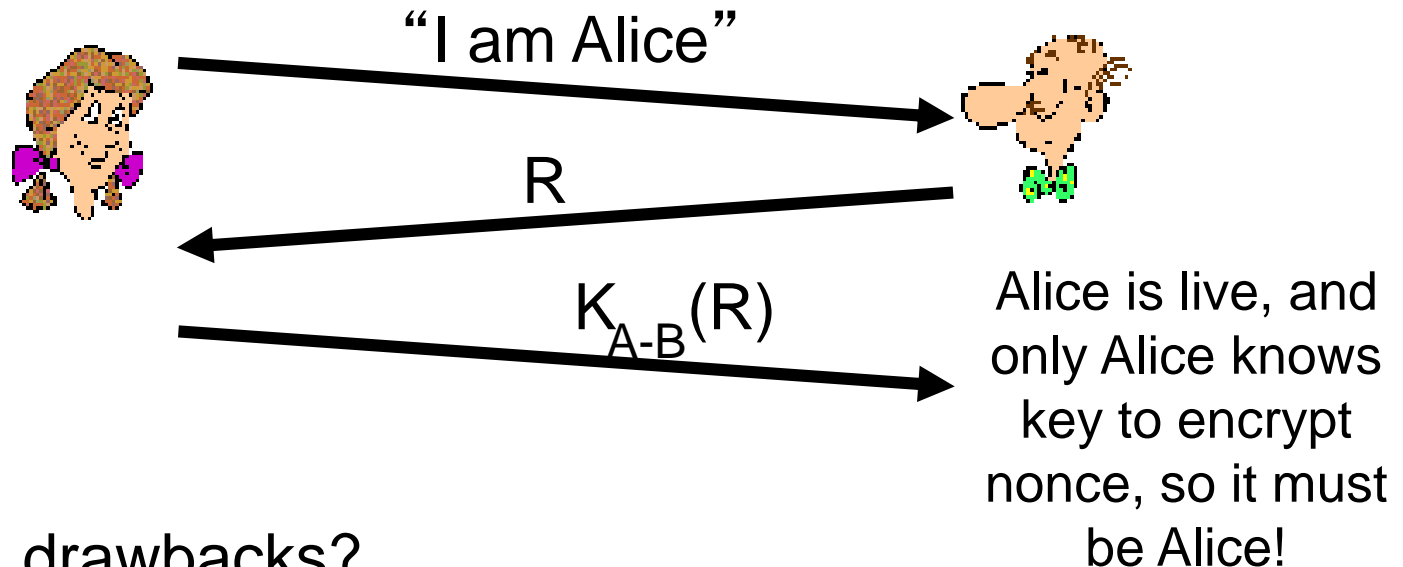| Alice's IP addr	encrypted password	"I'm Alice"

record
and
playback
still works!
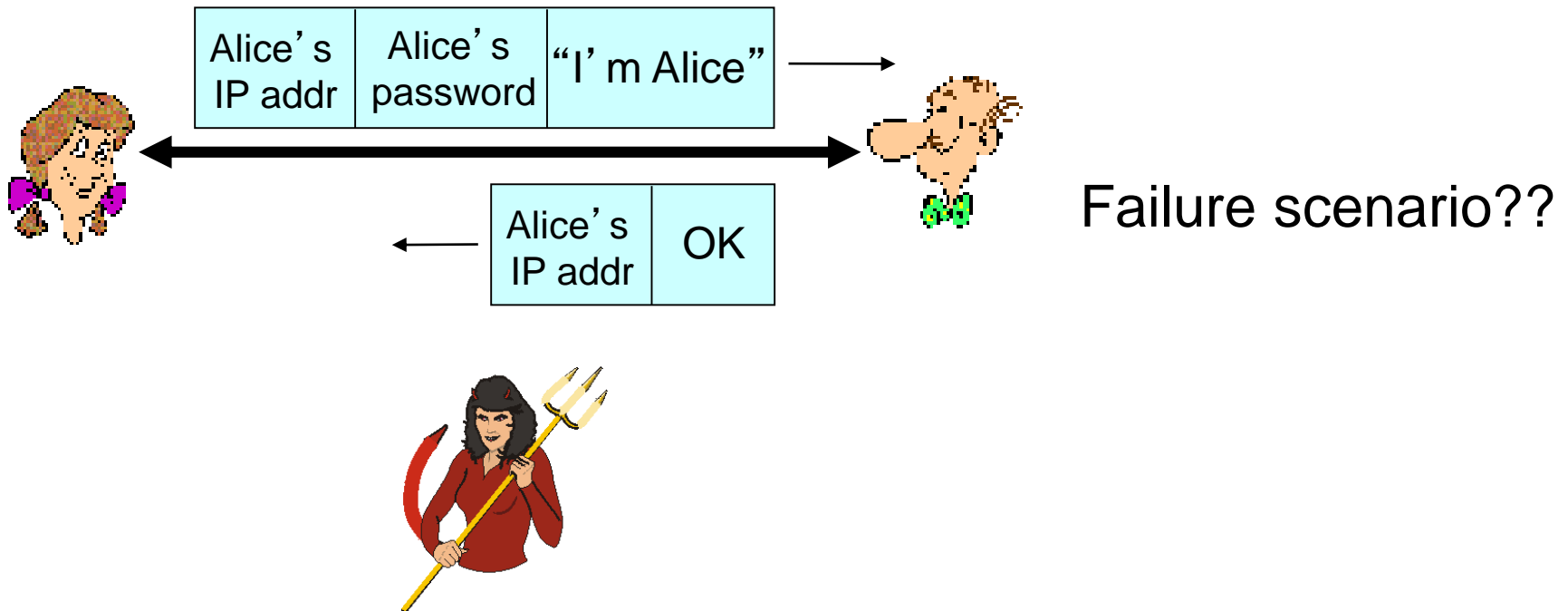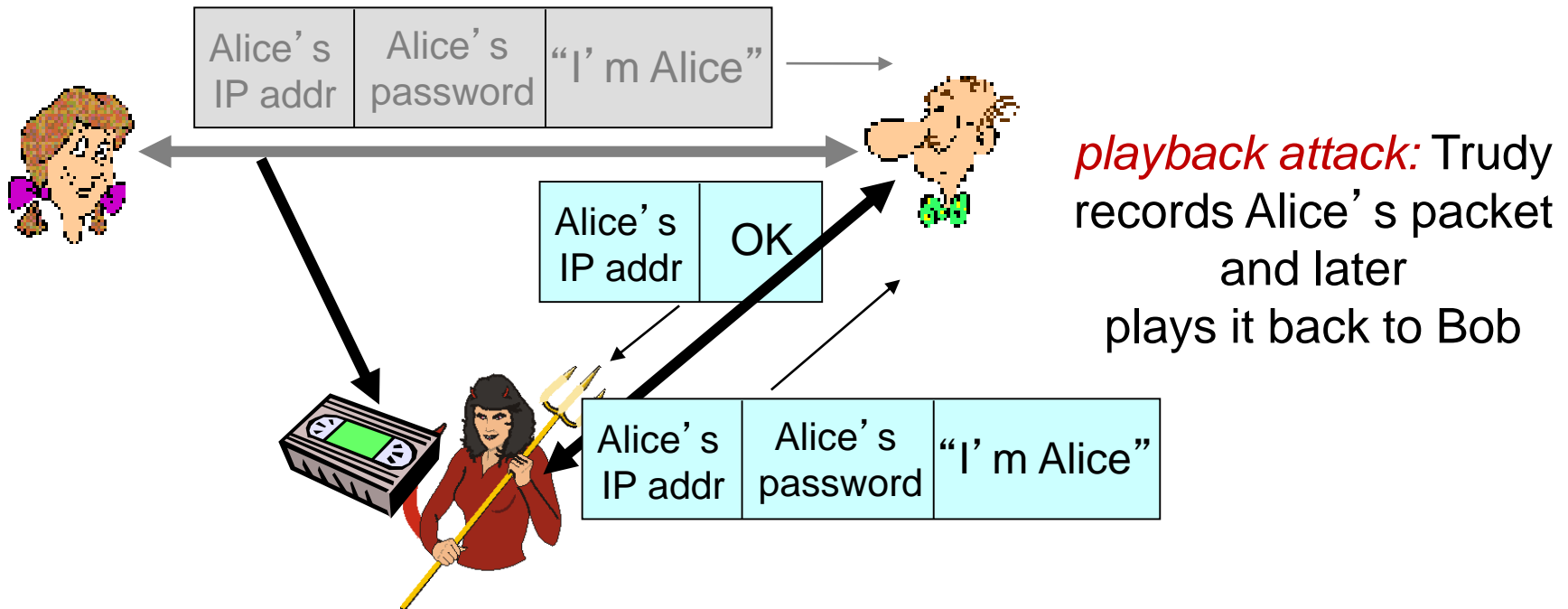
Authentication: yet another try

Goal: avoid playback attack

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

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

"I am Alice"

R

$K_{A-B}(R)$

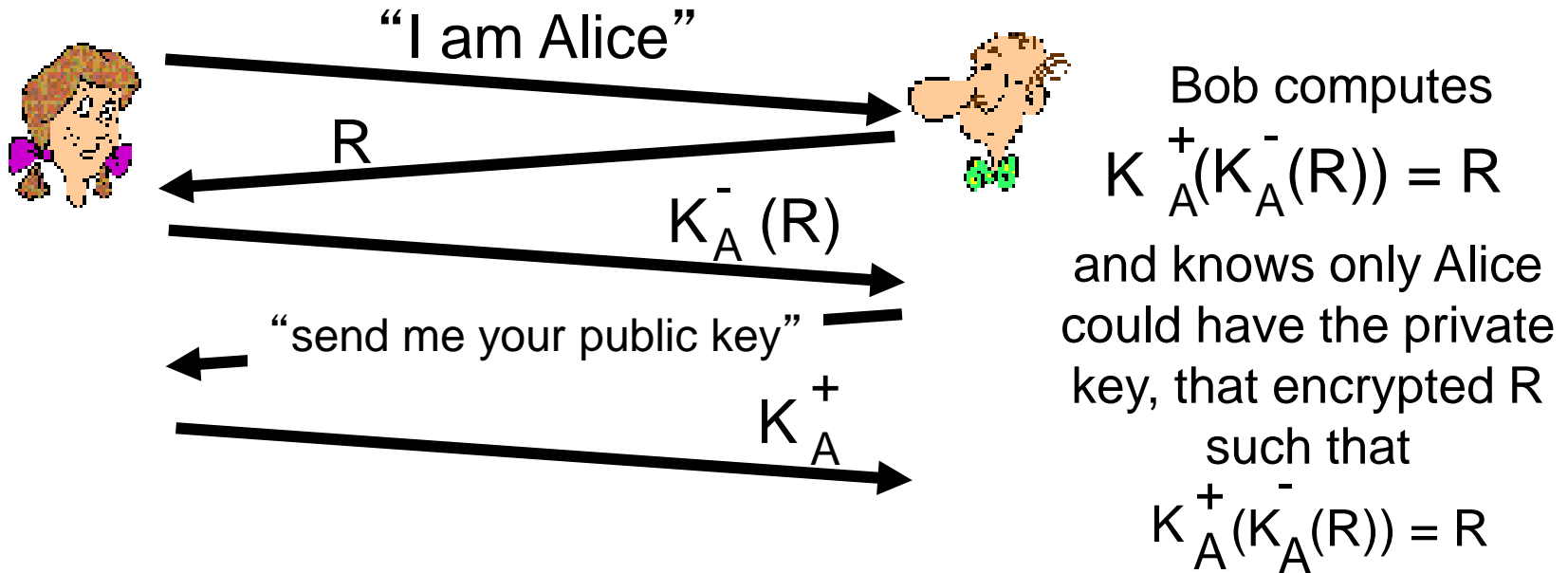Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

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

"I am Alice"

R

$K_A^-(R)$

"send me your public key"

K_A^+

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

I am Alice

I am Alice

R

$K_T^-(R)$

R

Send me your public key

$K_A^-(R)$

K_T^+

Send me your public key

K_A^+

$K_T^+(m)$

Trudy gets

$m = K_T^-(K_T^+(m))$
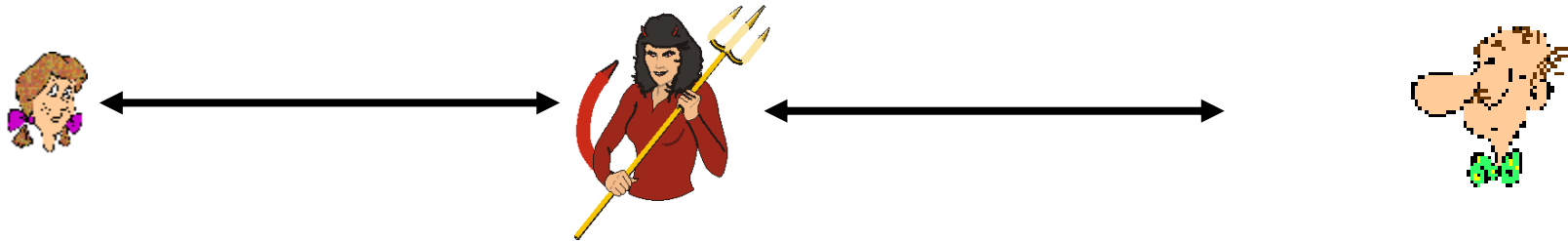
sends m to Alice
encrypted with
Alice's public key

$K_A^+(m)$

$m = K_A^-(K_A^+(m))$
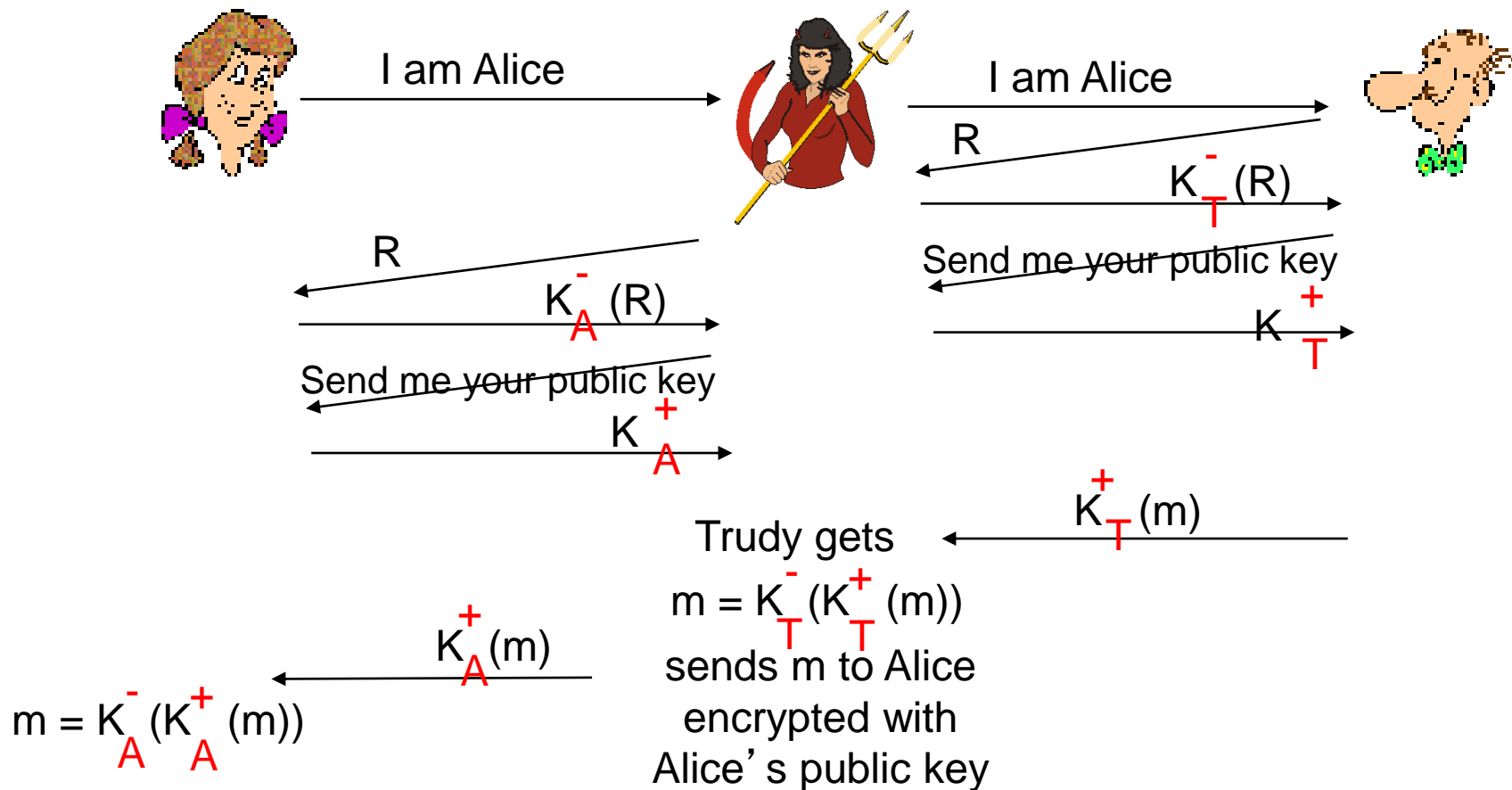
ap5.0: security hole

man (or 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!
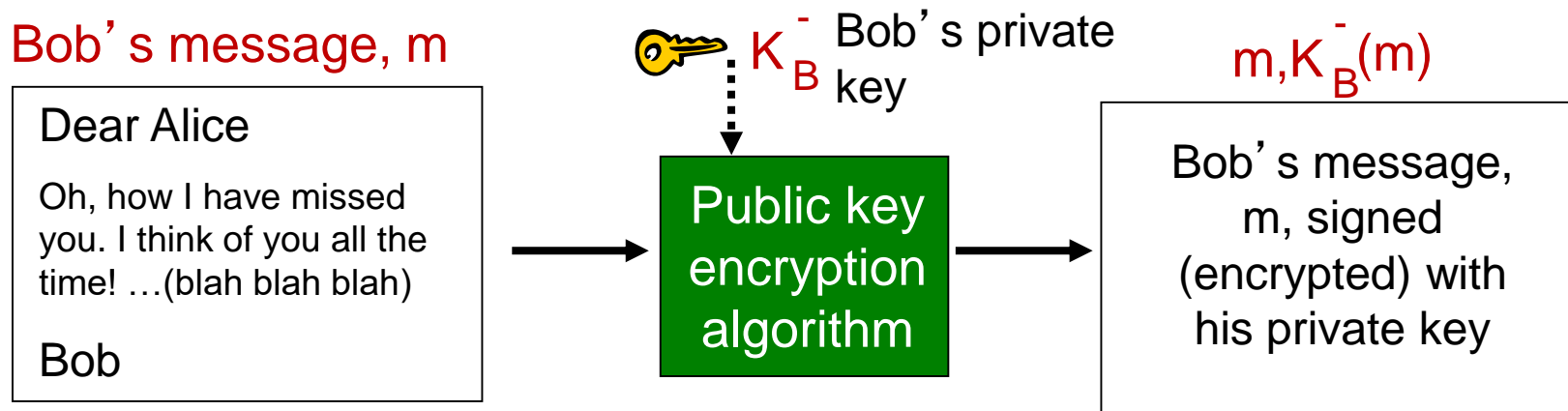
Chapter 8 roadmap

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, $K_B^-(m)$

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

$m, K_B^-(m)$

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

Digital signatures

❖ suppose Alice receives msg m, with signature: m, $K_B^-(m)$

❖ 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


```
large message m  →  H: Hash Function  →  H(m)
```

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)*.
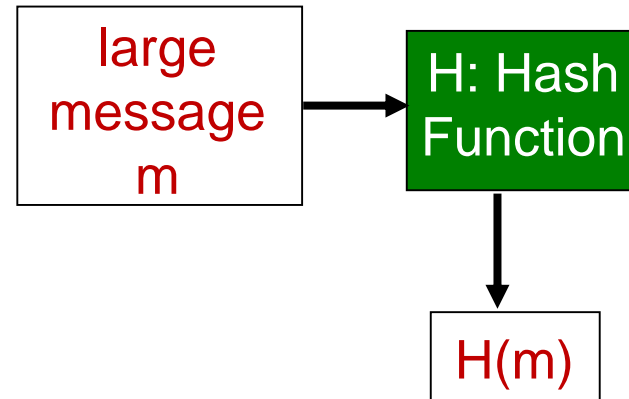
Hash function properties:

❖ 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
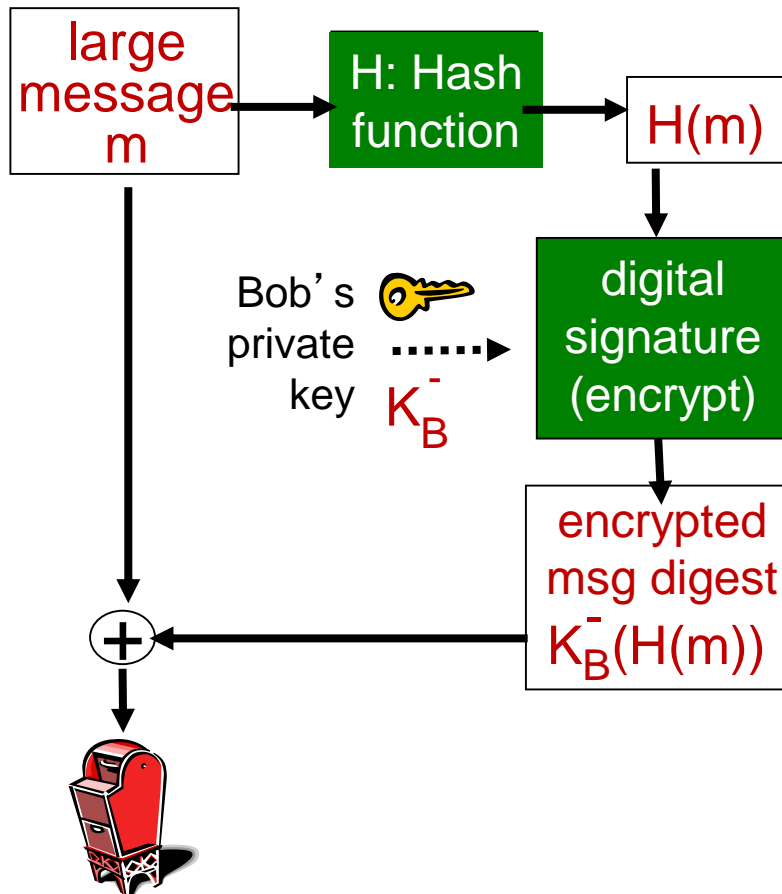
Internet checksum has some properties of hash function:

✓ 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:

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

different messages but identical checksums!

Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:

| large message m | → | H: Hash function | → | H(m) |

Bob's private key K_B^- ......→ digital signature (encrypt)

→ encrypted msg digest $K_B^-(H(m))$

large message m + encrypted msg digest $K_B^-(H(m))$

Alice side:

encrypted msg digest $K_B^-(H(m))$

Bob's public key K_B^+ ....→ digital signature (decrypt) → H(m)

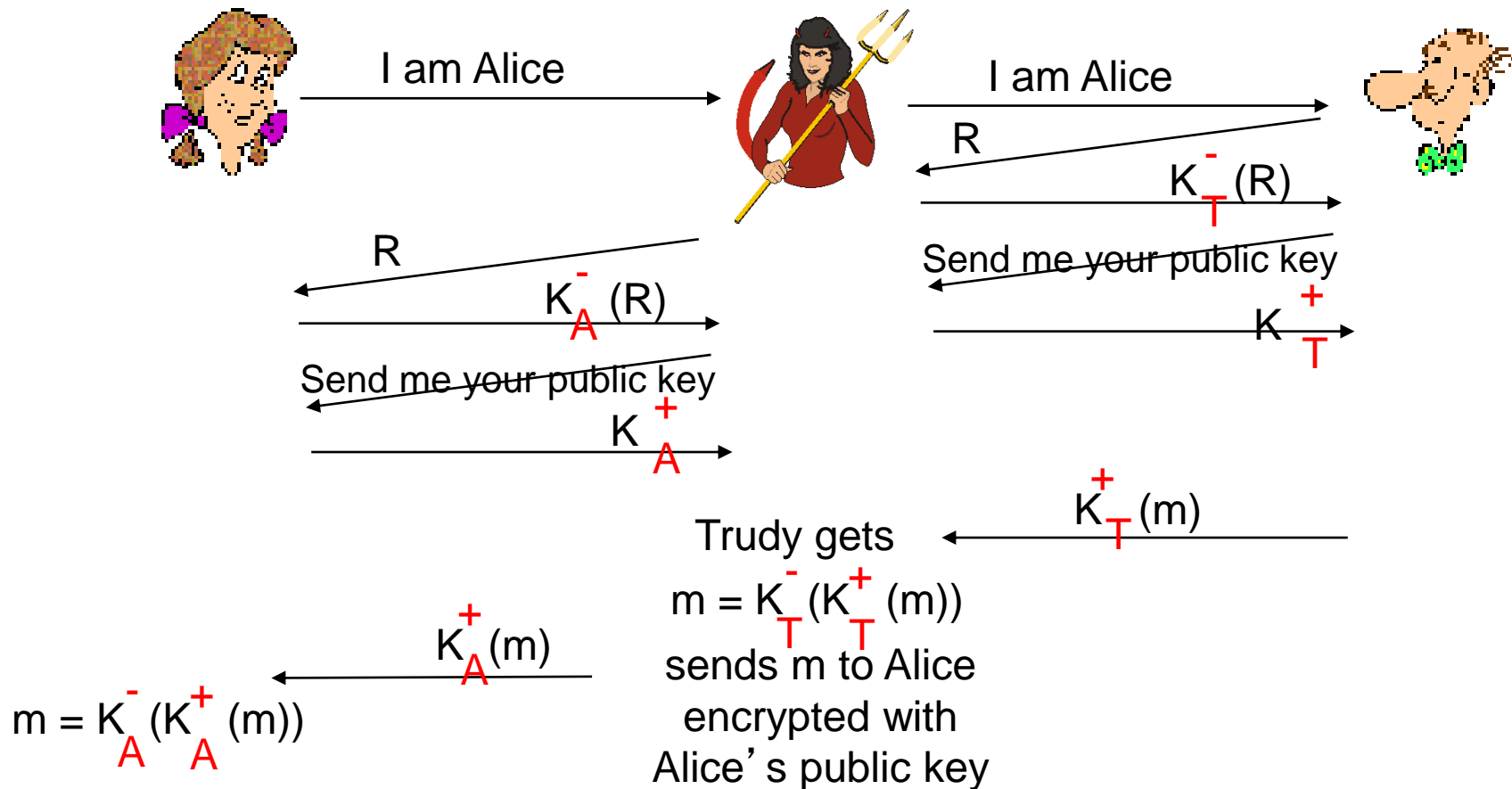large message m → H: Hash function → H(m)

H(m) ... equal ? ... H(m)
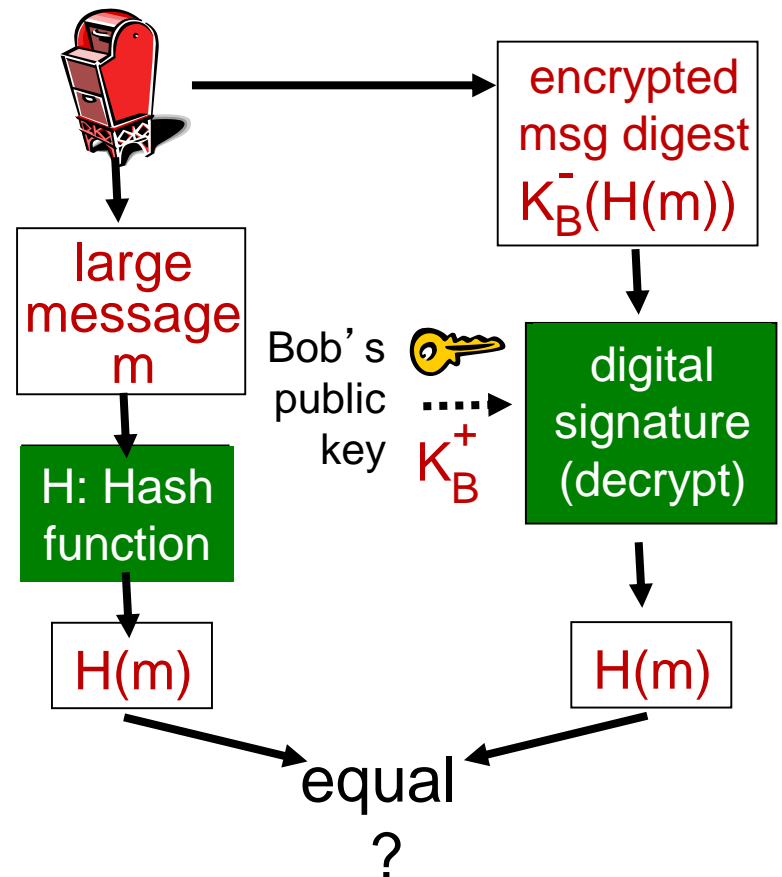
Hash function algorithms

❖ 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 [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Recall: ap5.0 security hole

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

I am Alice

I am Alice

R

$K_T^-(R)$

R

$K_A^-(R)$

Send me your public key

Send me your public key

K_T^+

K_A^+

$K_T^+(m)$

Trudy gets

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

sends m to Alice encrypted with Alice's public key
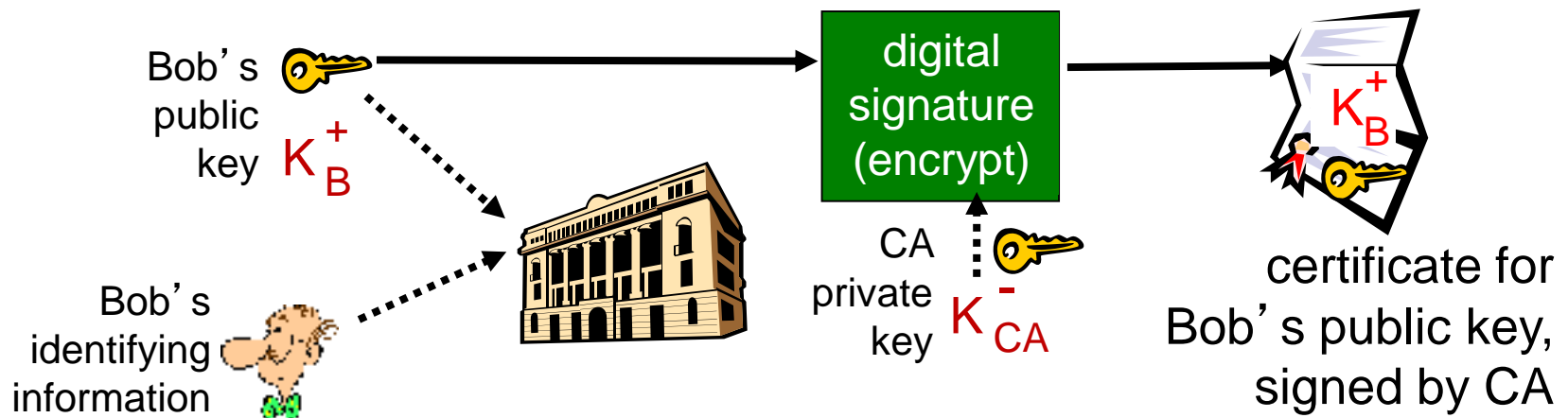
$K_A^+(m)$

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

Public-key certification

❖ motivation: Trudy plays pizza prank on Bob

- Trudy creates e-mail order:
  Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
- Trudy signs order with her private key
- Trudy sends order to Pizza Store
- Trudy sends to Pizza Store her public key, but says it's Bob's public key
- Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
- Bob doesn't even like pepperoni

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"

Bob's public key K_B^+

Bob's identifying information

digital signature (encrypt)

CA private key K_{CA}^-

K_B^+

certificate for Bob's public key, signed by CA

X509 Certificate format

Figure 4.4 X.509 Formats