456: Network-Centric Programming

Yingying (Jennifer) Chen

Web: https://www.winlab.rutgers.edu/~yychen/
Email: yingche@scarletmail.rutgers.edu

Department of Electrical and Computer Engineering
Rutgers University

What is network security?

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

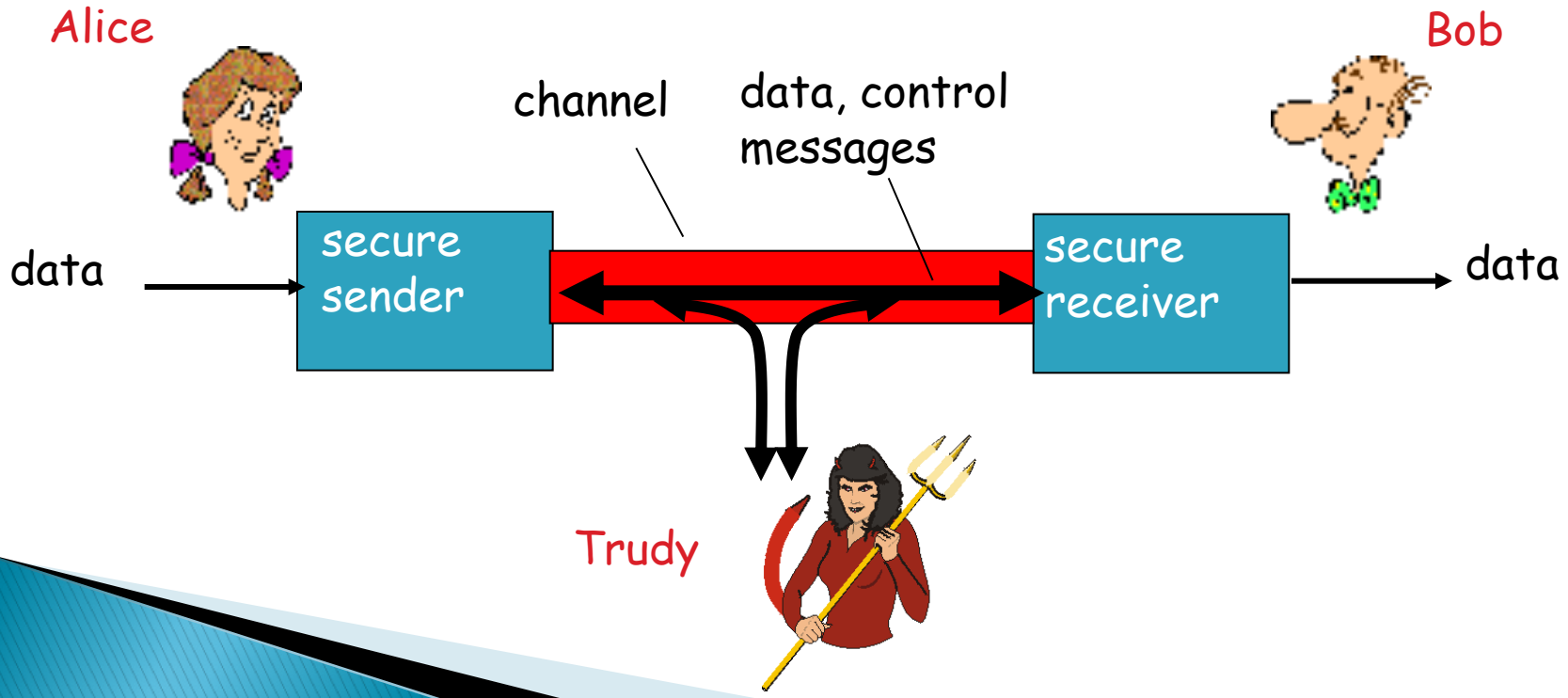
Authentication: sender, receiver want to confirm identity of each other

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 ……

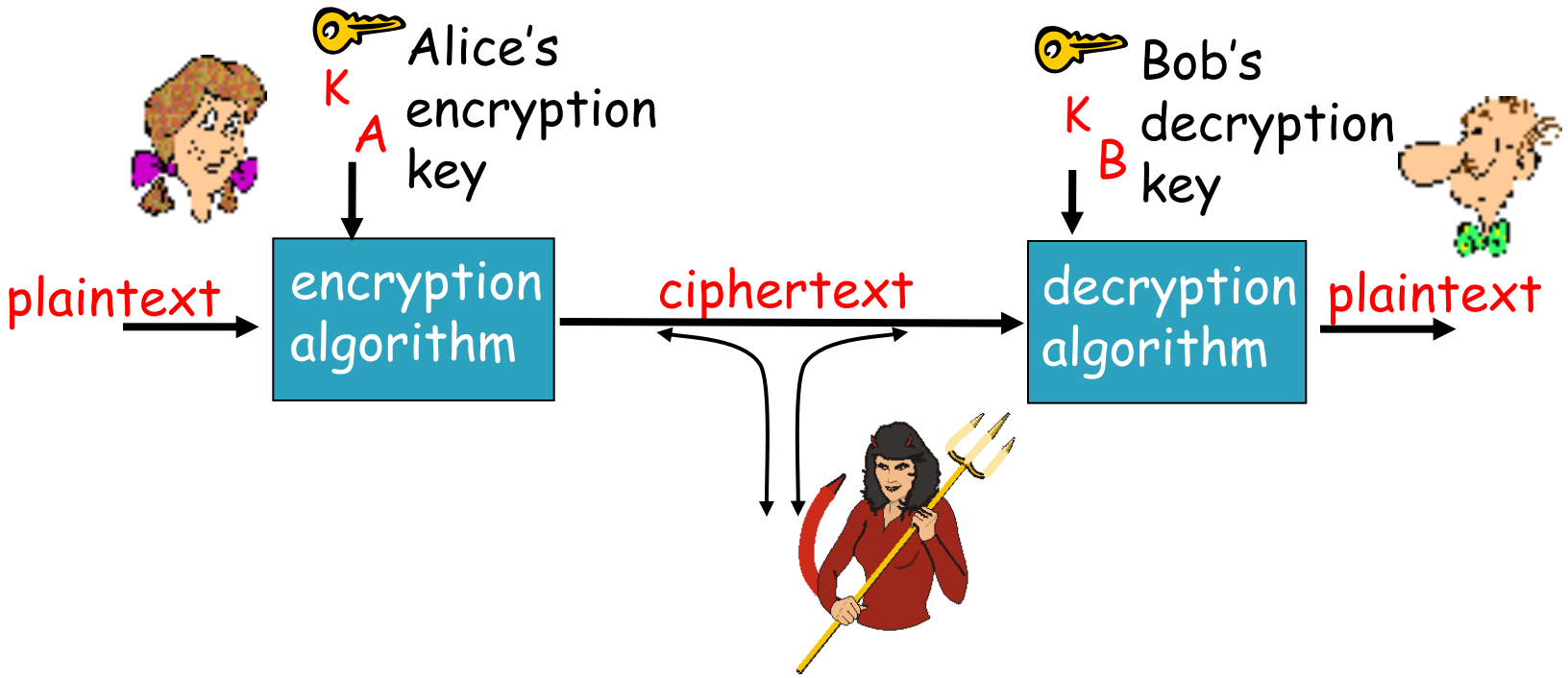
Available Security Tools

- Cryptography/Encryption:
  - Encode a message in a way that only the communicating parties can interpret it
  - Used for secrecy and authentication
- Signatures
  - Allow for the authentication of a message's sender and the message's integrity
  - Used for authentication, non-repudiation and integrity control

Rules for Encryption

- Encryption requires both an encryption algorithm and an encryption "key"
  - Key is a string which controls how the algorithm encrypts
- Algorithm:
  - Should be public and known to all
    - Inspires trust that the algorithm works
- Keys:
  - Should be long enough to prevent easy breaking of the encryption
  - Should be short enough to keep algorithm efficient
  - Typical key lengths: 56-bit, 128-bit, 256-bit, 512-bit

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

- Caesar cipher: fixed letter substitution use a regular pattern
- monoalphabetic cipher: substitute any letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz
                ↓                   ↓
ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

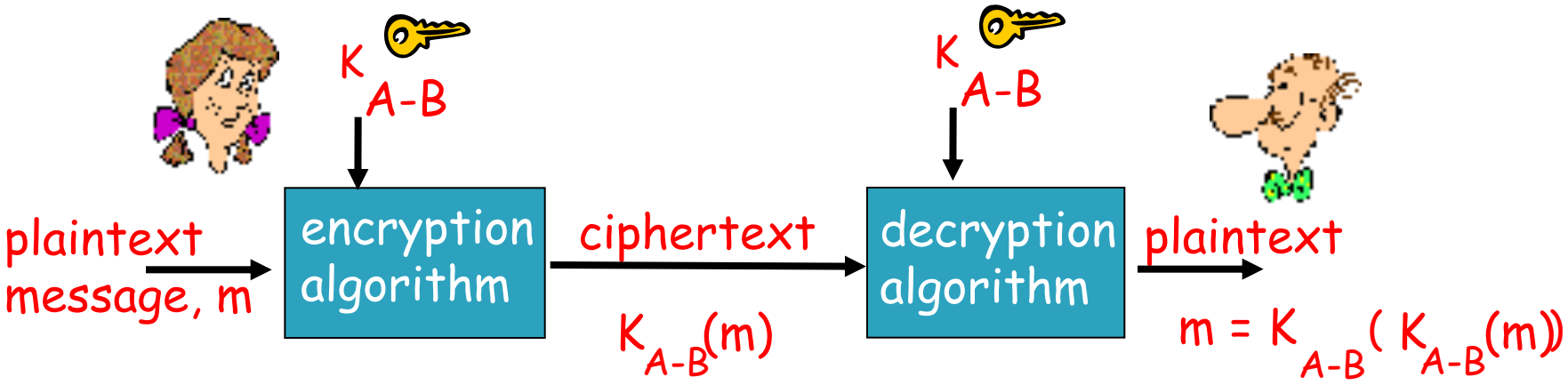
```
Plaintext: bob. i love you. alice

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-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

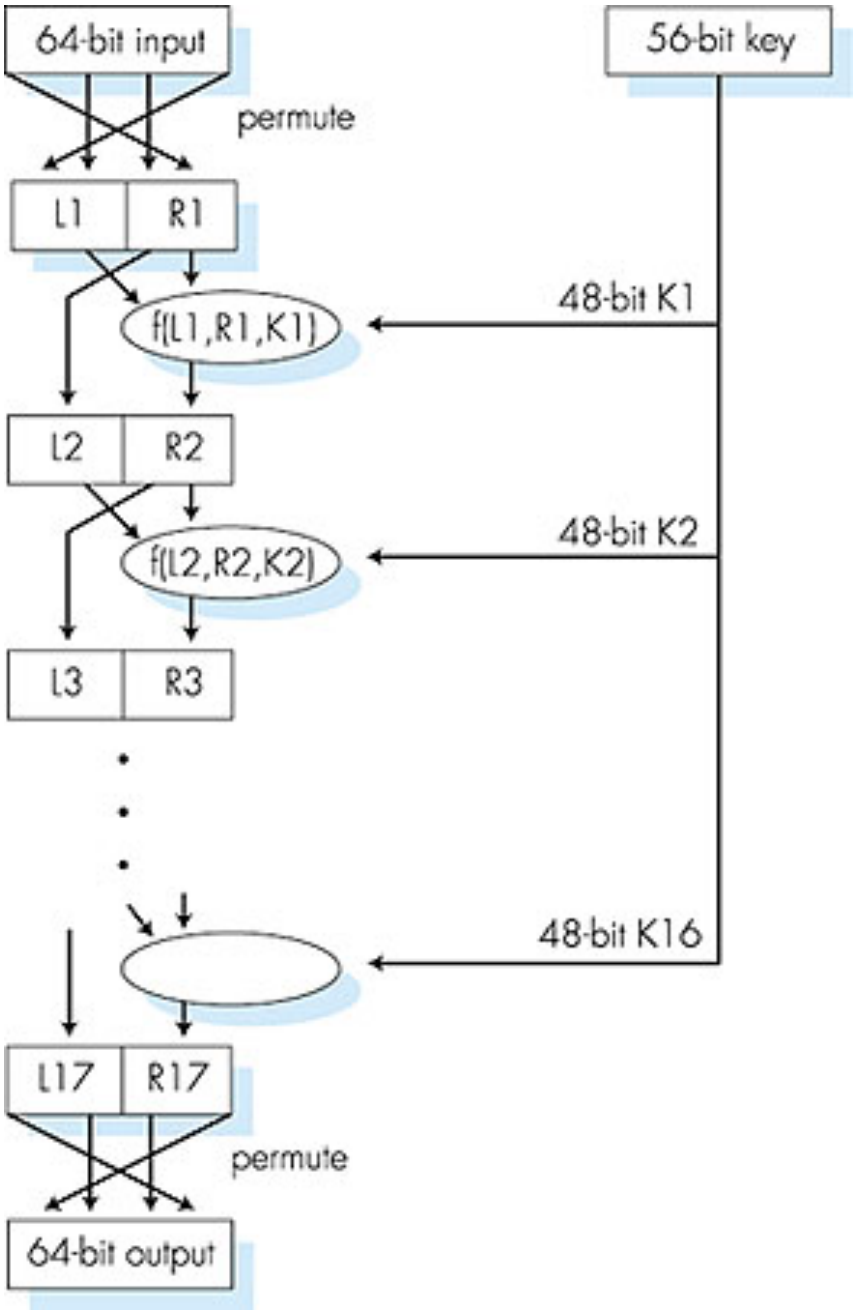
Symmetric key crypto: DES

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

Concerns

- Problem with all the cryptography algorithms used so far:
  - How to communicate the shared key safely
- Is there a way to do cryptography without worrying about this concern?

Yes, public key cryptography

Public Key Cryptography

- Uses two different keys: an encryption key and a decryption key
- Each user holds:
  - a private decryption key to decrypt messages sent to her
  - a public encryption key that everyone else should use to encrypt messages that are sent to her
- Important property of public key cryptography algorithms:
  - Private decryption key cannot be easily determined by knowing the public encryption key

Public Key Cryptography

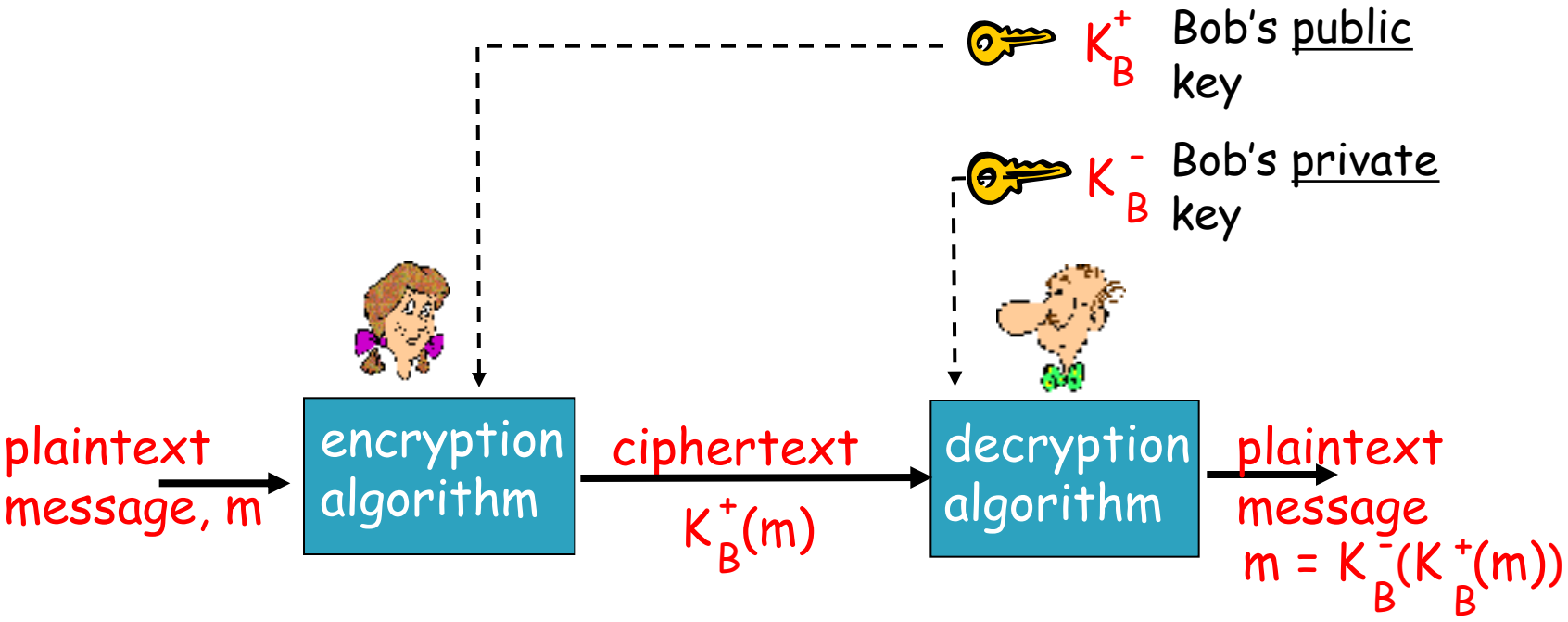
symmetric key crypto

- 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 $K_B^+(\ )$ and $K_B^-(\ )$ 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 *e*d-1* is exactly divisible by *z*. (in other words: *ed* mod *z* = *1*).

5. *Public* key is *(n,e)*. *Private* key is *(n,d)*.

K_B^+ 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.

encrypt:

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

decrypt:

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

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

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

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

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

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

(using number theory result above)

$$= m^1 \bmod n$$

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

$$= m$$

RSA: another important property

The following property will be *very* useful later:

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

use public key first, followed by private key

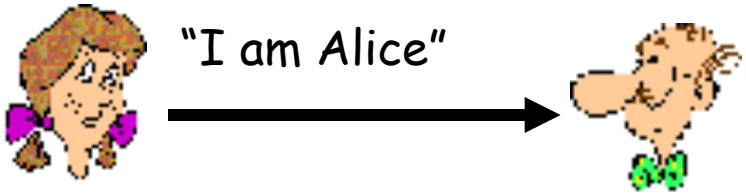
use private key first, followed by public key

Result is the same!

Authentication

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

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



Failure scenario??

Authentication

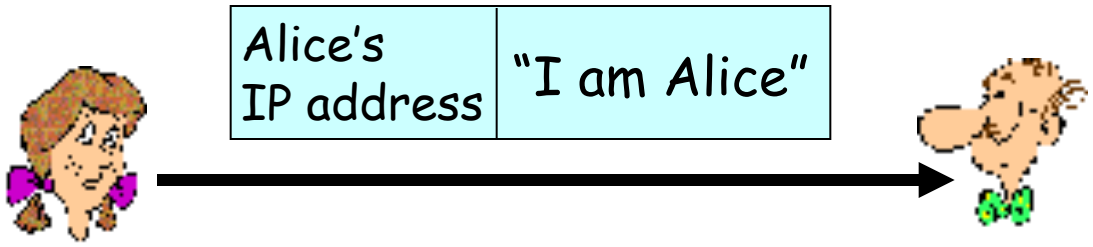
Goal: Bob wants Alice to "prove" her identity to him

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

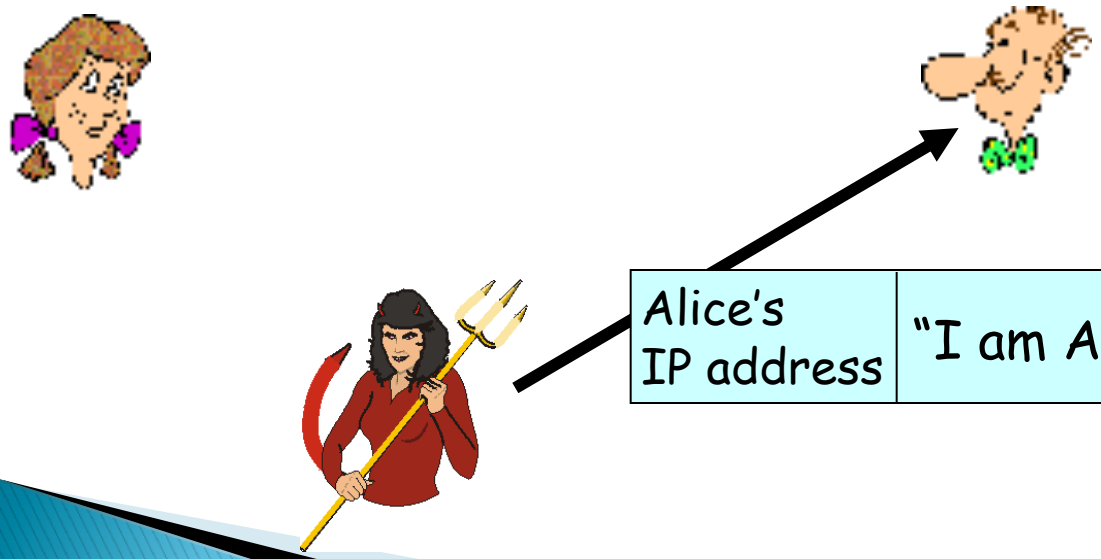
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Failure scenario??

Authentication: another try

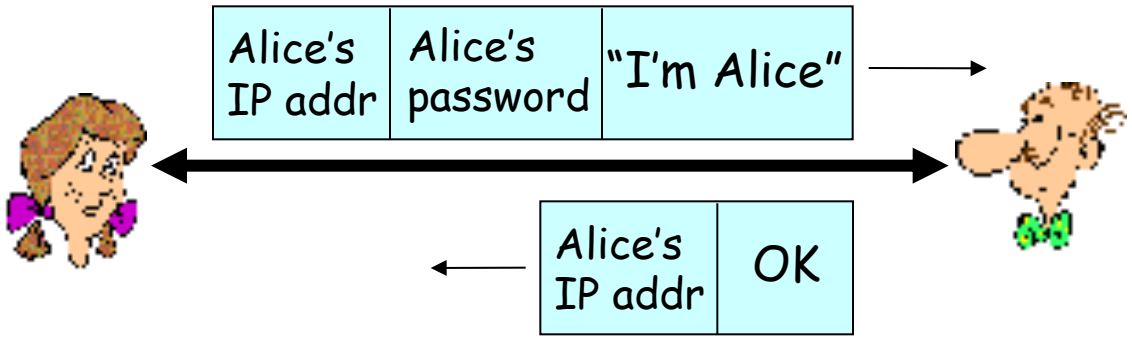
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



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

Authentication: another try

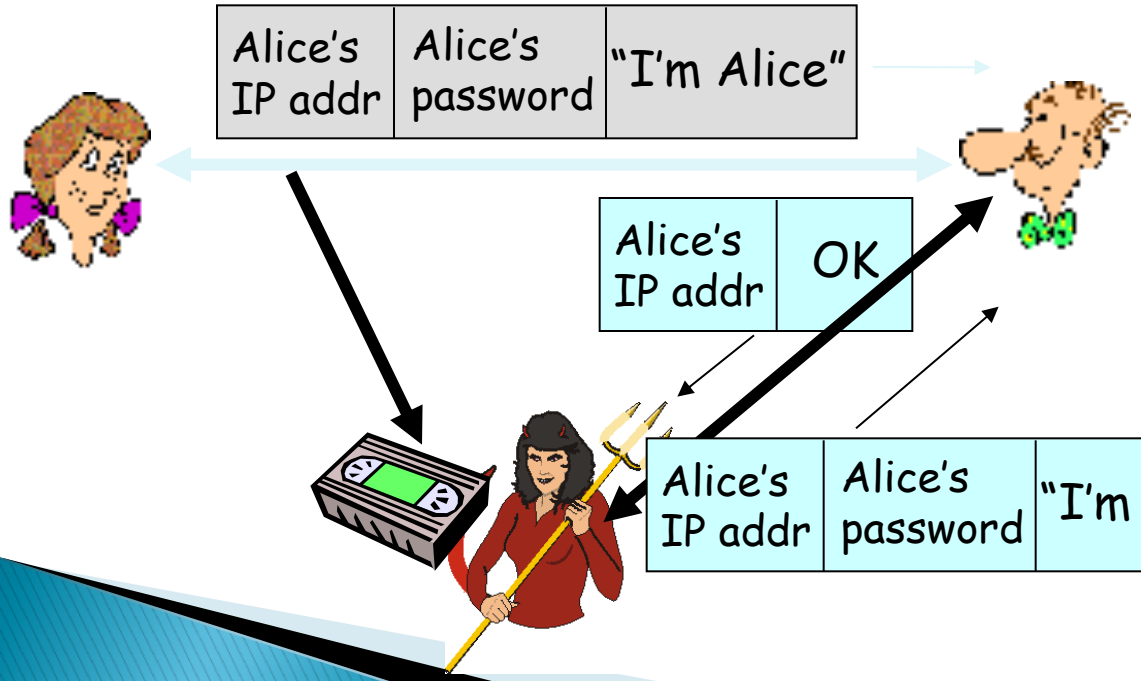
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

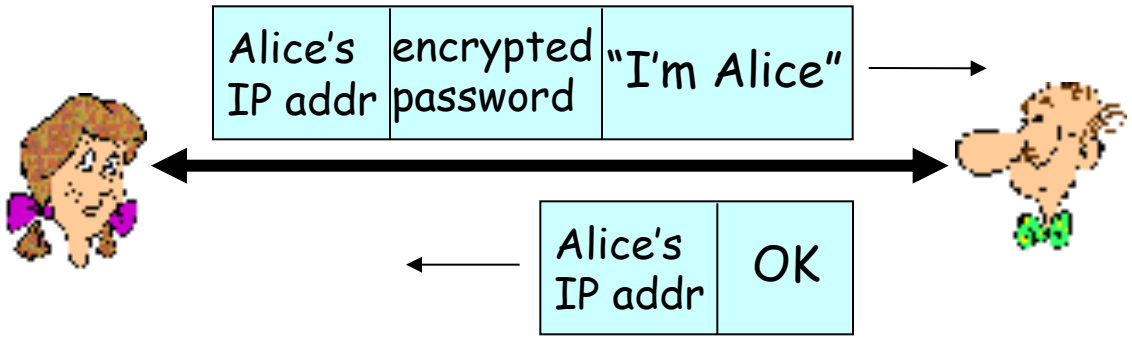
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



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

Authentication: yet another try

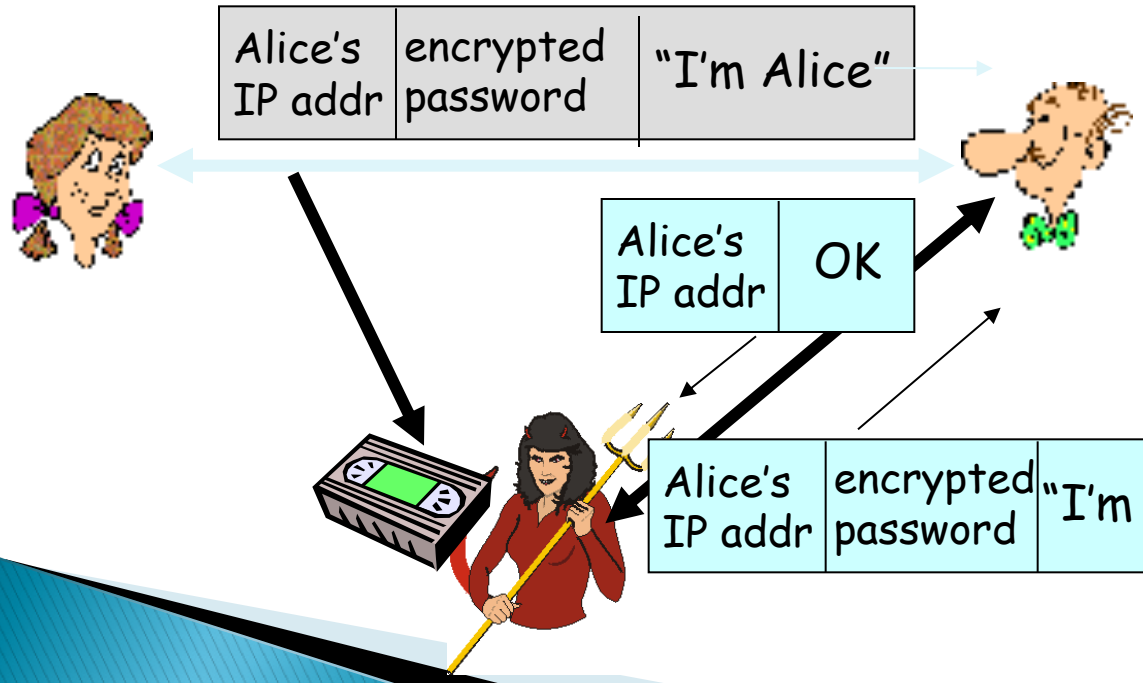
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: another try

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



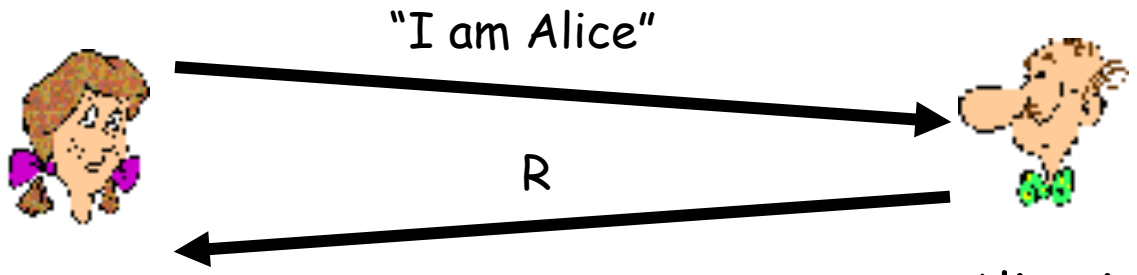
record and playback still works!

Authentication: yet another try

Goal: avoid playback attack

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

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



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

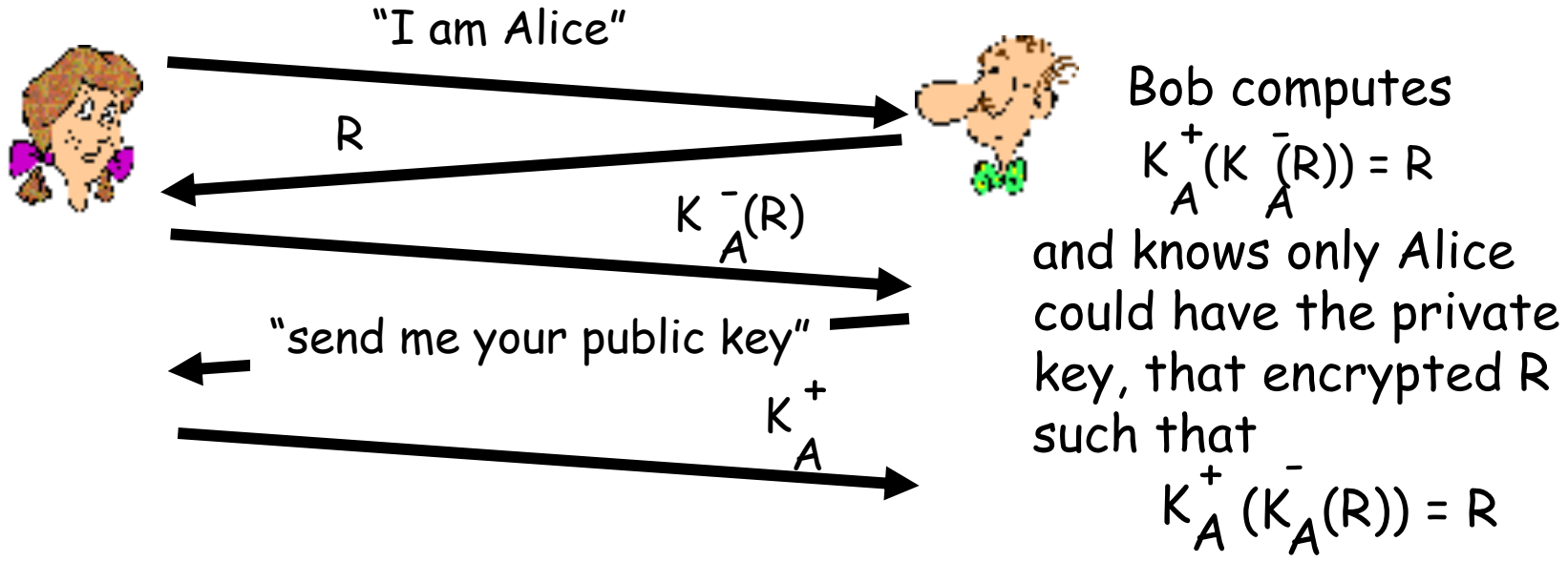
Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

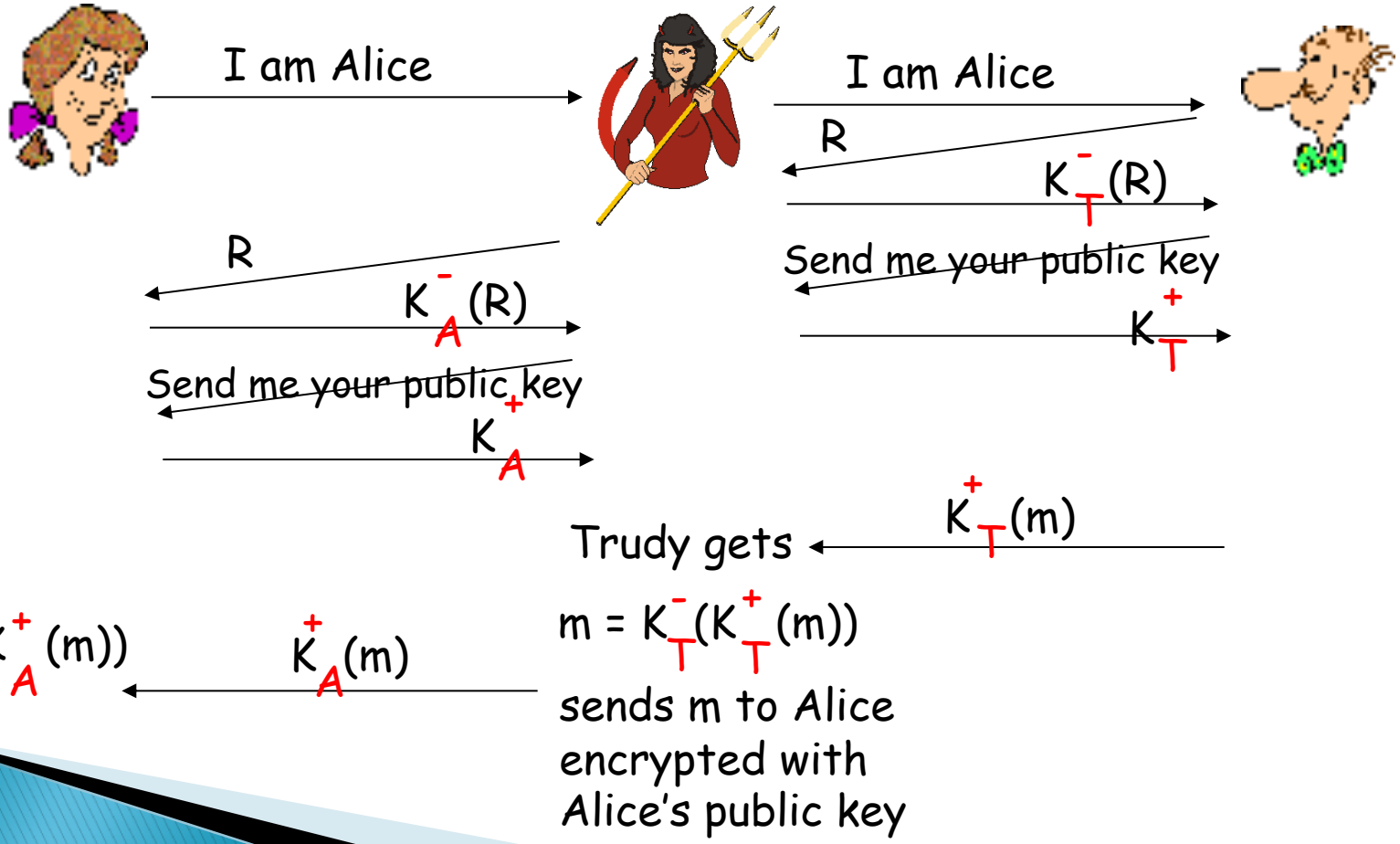
- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



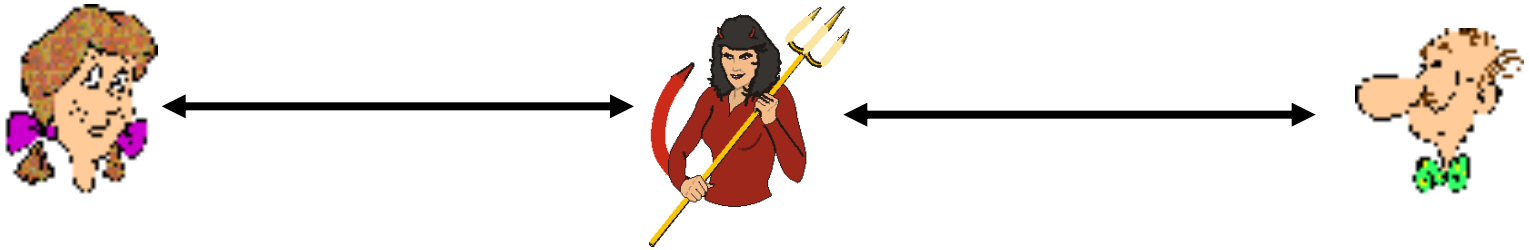
ap5.0: security hole

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 and monitor what they are doing.

Need to solve problem in public key distribution

Digital Signatures

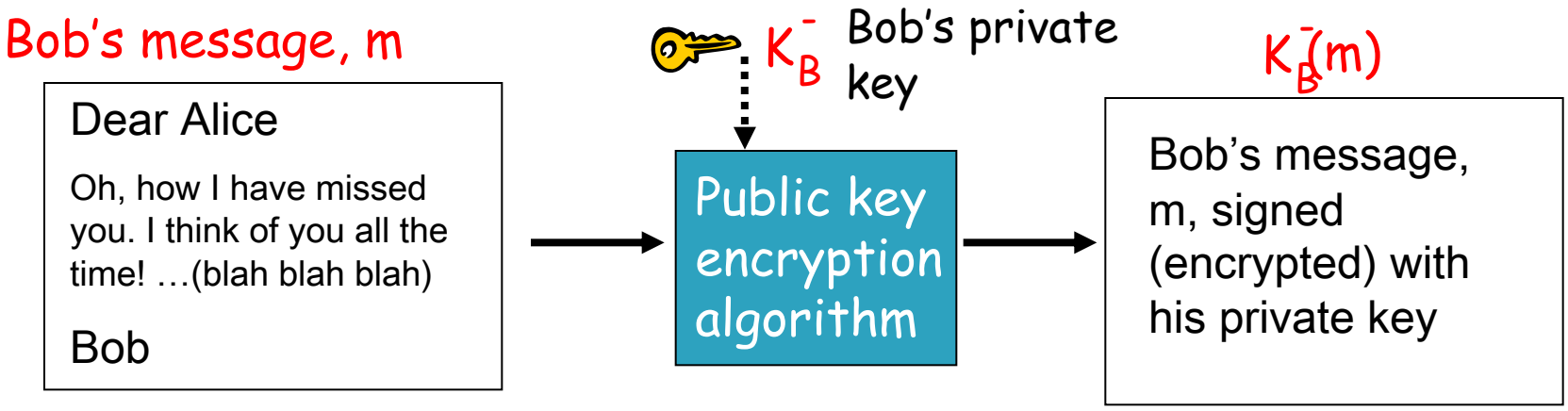
Cryptographic technique analogous to hand-written signatures.

- 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.

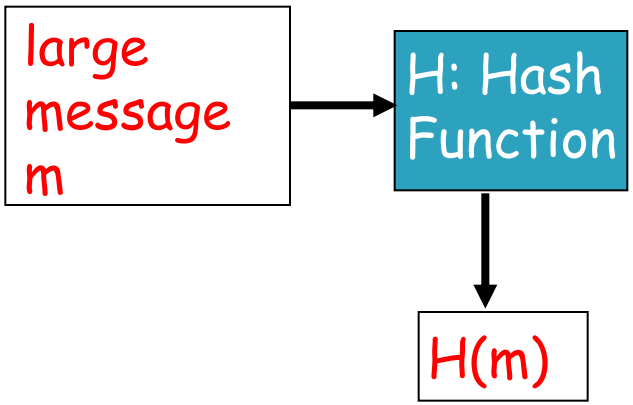
Alice thus verifies that:

- ✓ 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.

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:

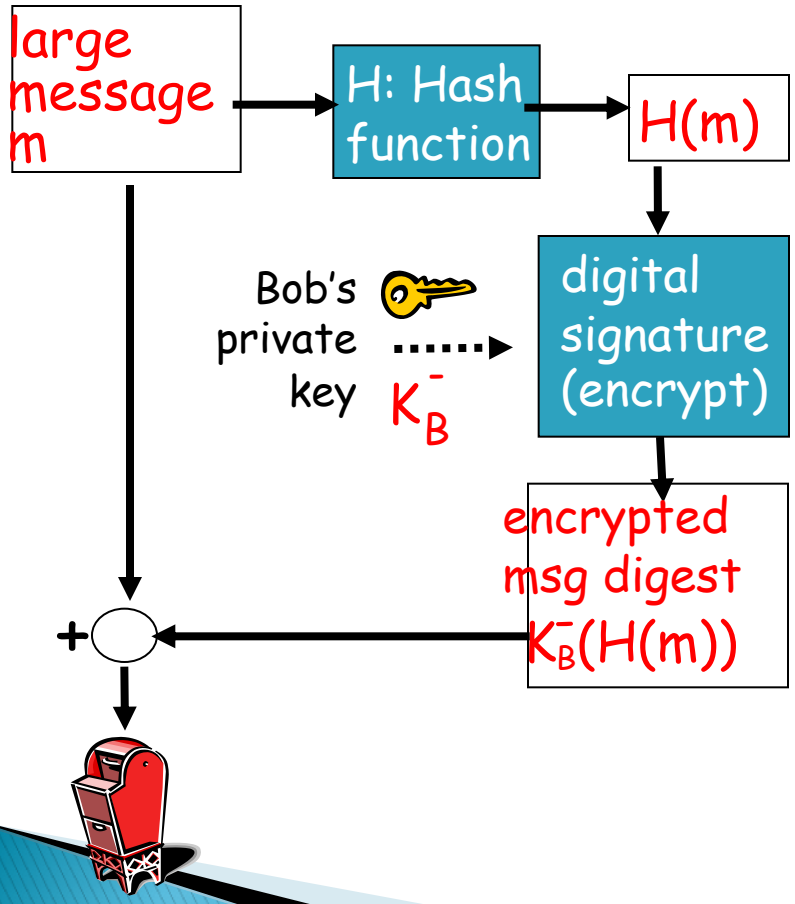
message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 4F 42
	B2 C1 D2 AC

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

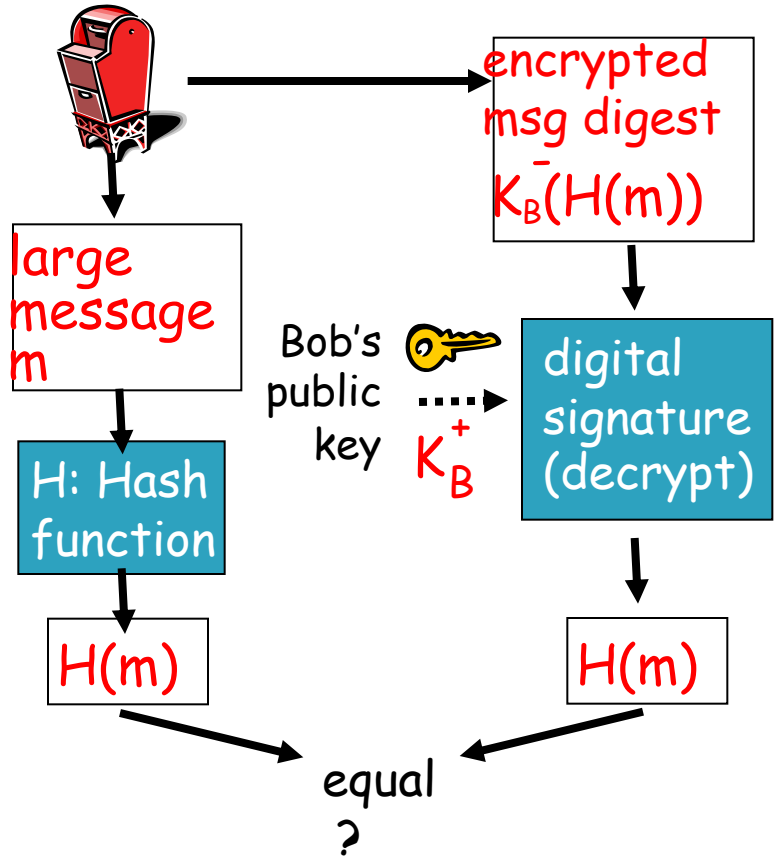
different messages but identical checksums!

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

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:

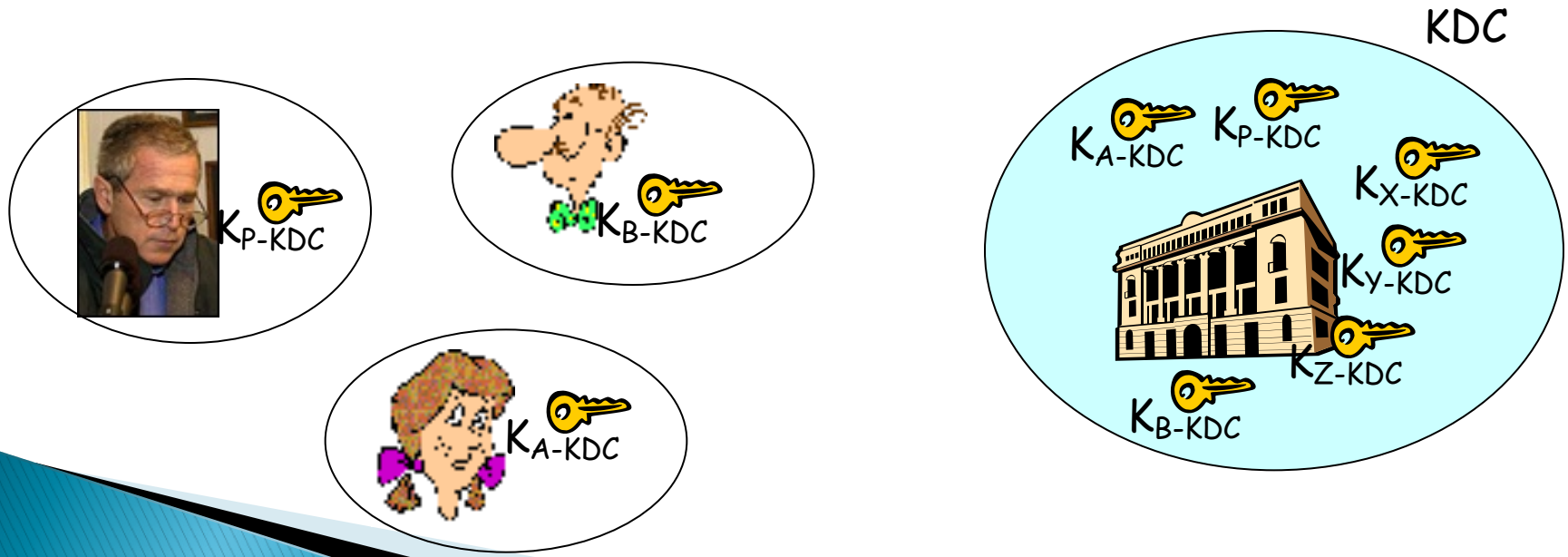
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

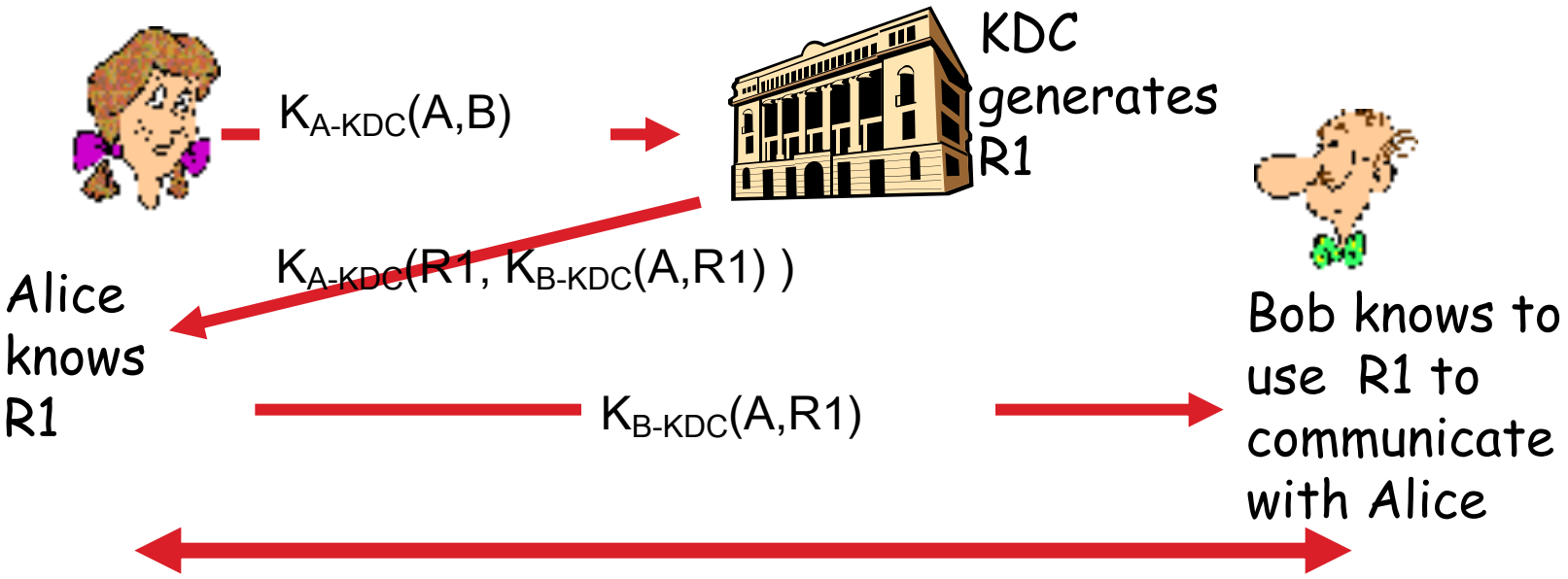
Key Distribution Center (KDC)

- 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.



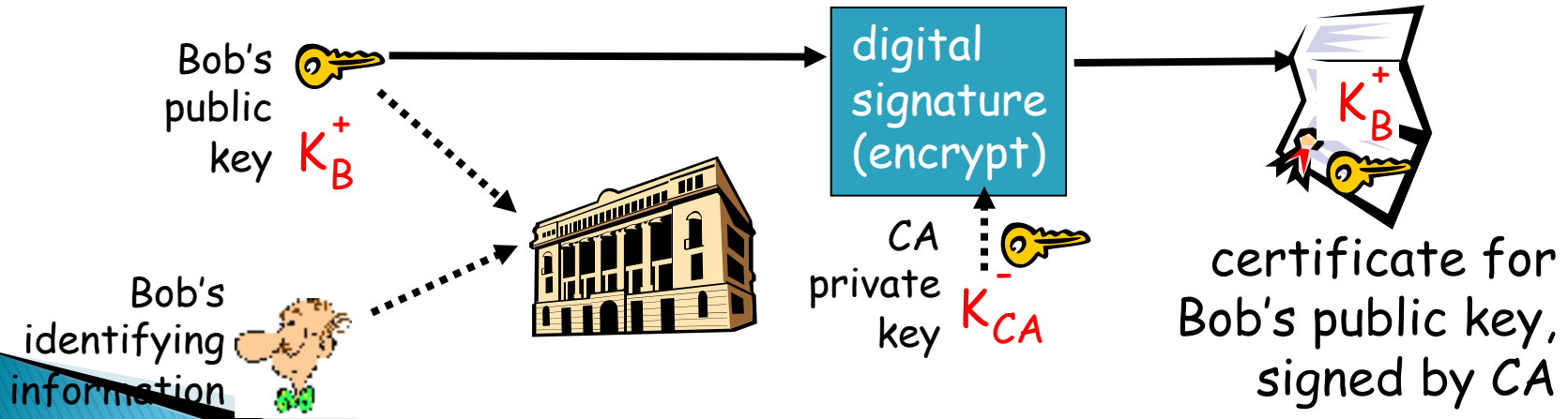
Key Distribution Center (KDC)

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



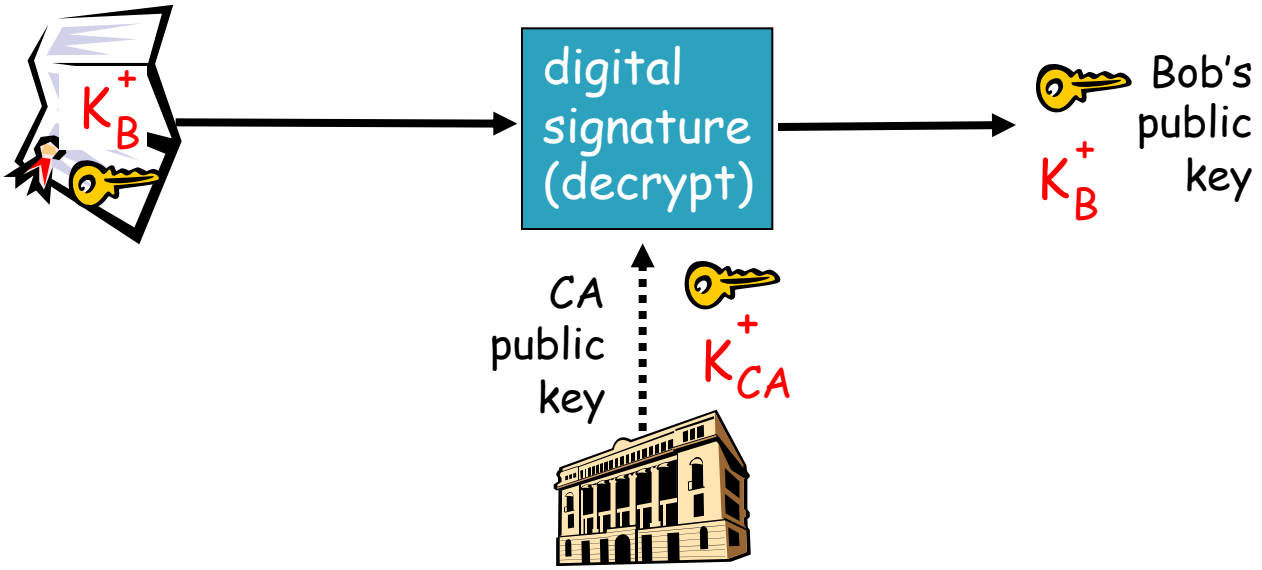
Certification Authorities

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



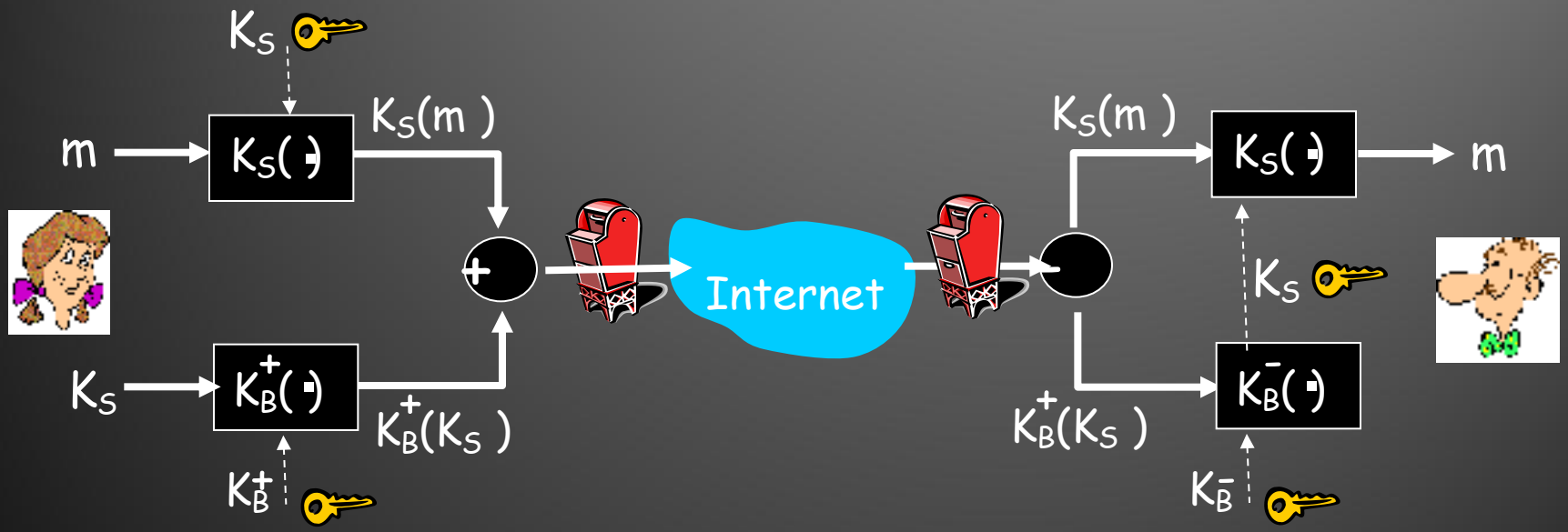
Certification Authorities

- 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



Secure e-mail

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

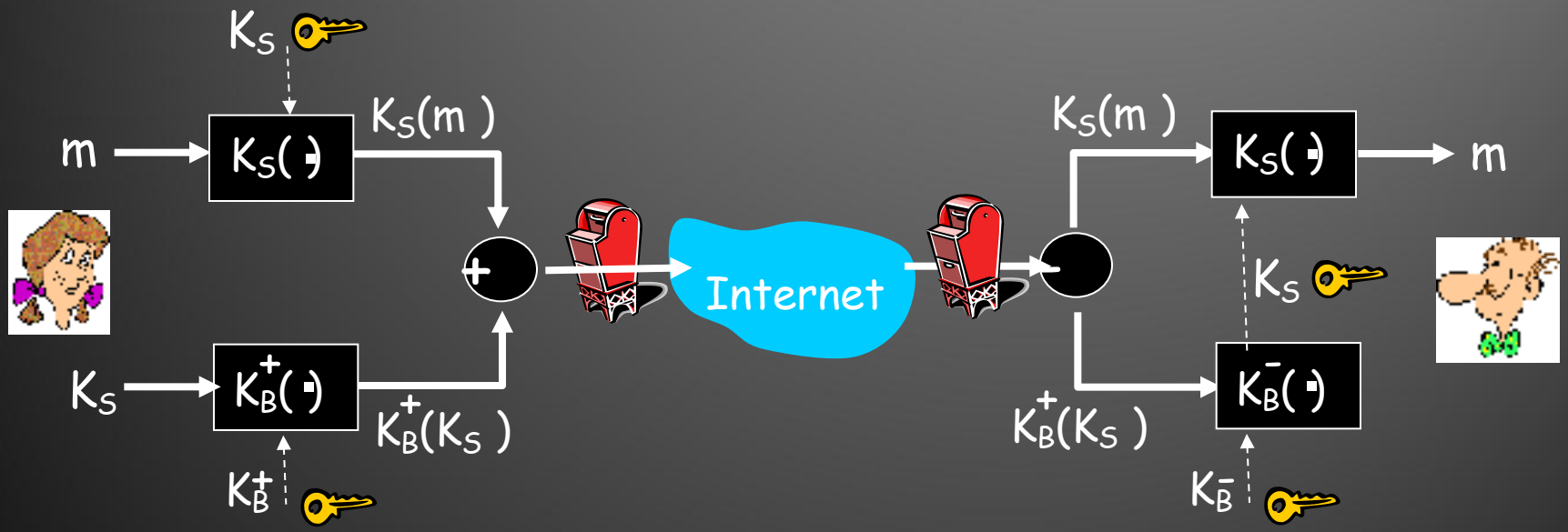


Alice:

- generates random *symmetric* 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.

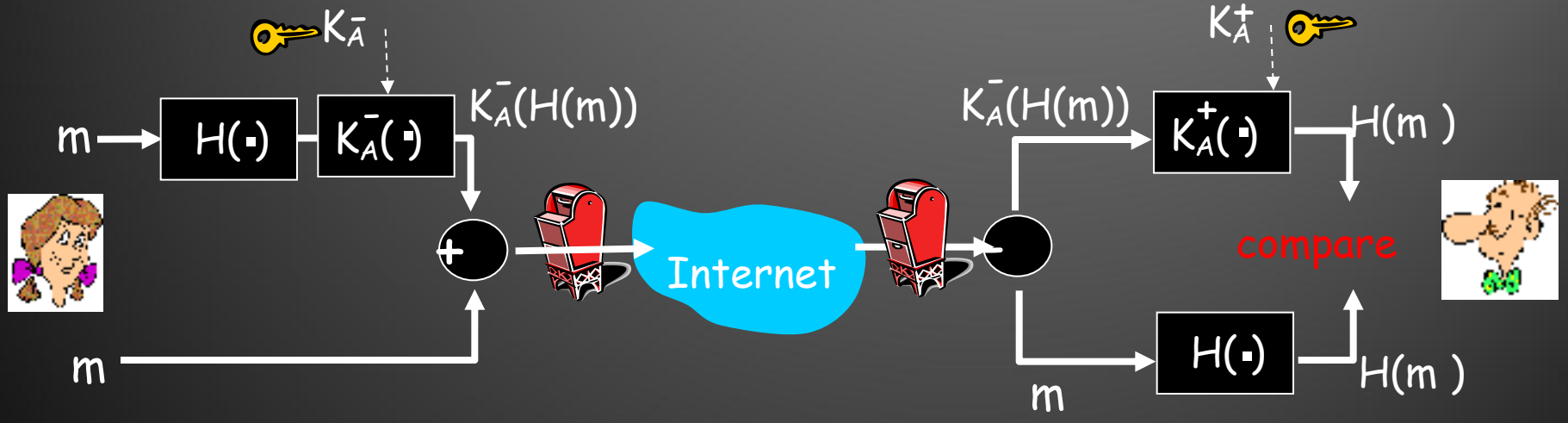


Bob:

- 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)

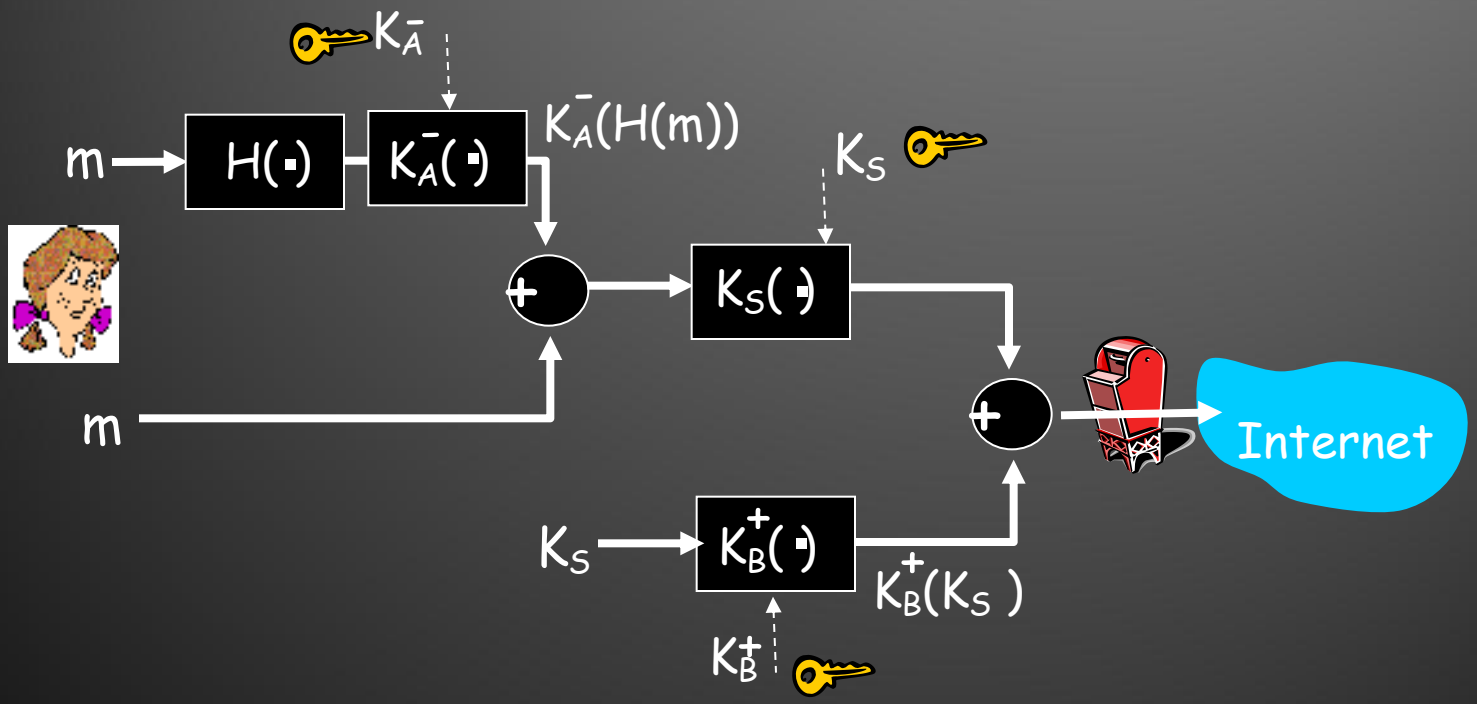
- Alice wants to provide sender authentication, message integrity.



- 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, and 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.

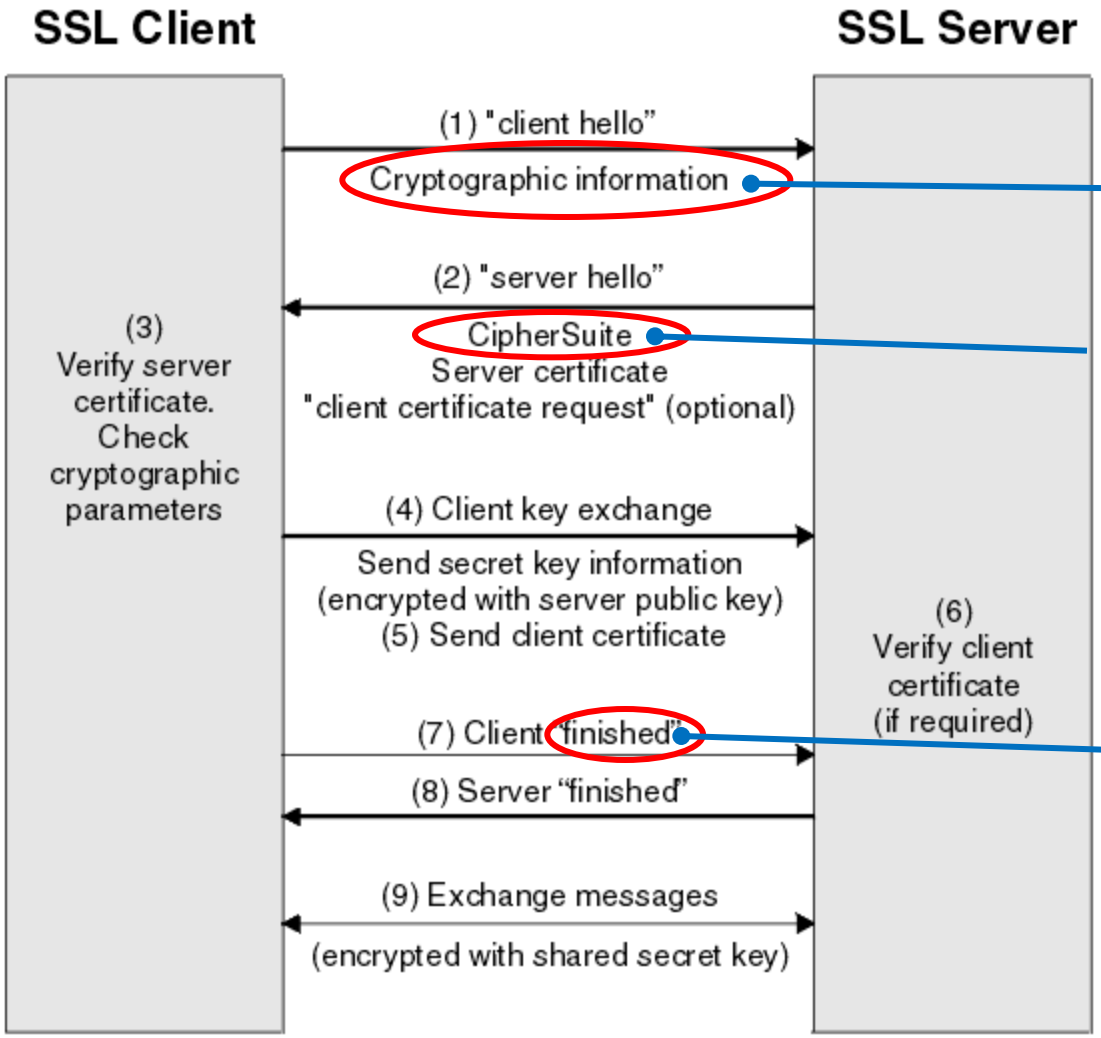
Secure Sockets Layer (SSL)

- Layered on top of TCP/IP protocol suite but under application layer
- Most frequently used to secure communications between browsers and web servers
  - Other applications possible!
- Provides
  - Message authentication
  - Message confidentiality
  - Message integrity

SSL Negotiates Cipher Suites

- Cipher suite: a set of cryptographic algorithms
- Client and Server may support different cipher suites
  - Authentication: RSA, Diffie-Hellmann key exchange
  - Encryption: 56bit DES, 128bit 3DES
  - Message Authentication Code (MAC): MD5, SHA
- The SSL handshake protocol allows the client and server to agree upon a cipher suite to use

SSL Handshake Protocol



Cryptographic information contains the CipherSuite list supported by the client.

The server choses CipherSuite from the list provided by client

"finished" message encrypted with the exchanged secret key

Step 7 and 8 indicate that the SSL handshake is complete

Homework Readings

- ALP: Advanced Linux Programming
  - Chapter 10