Cryptography as a solution to secure communication

Artem A. Lenskiy

Seoul Tech Society

Overview

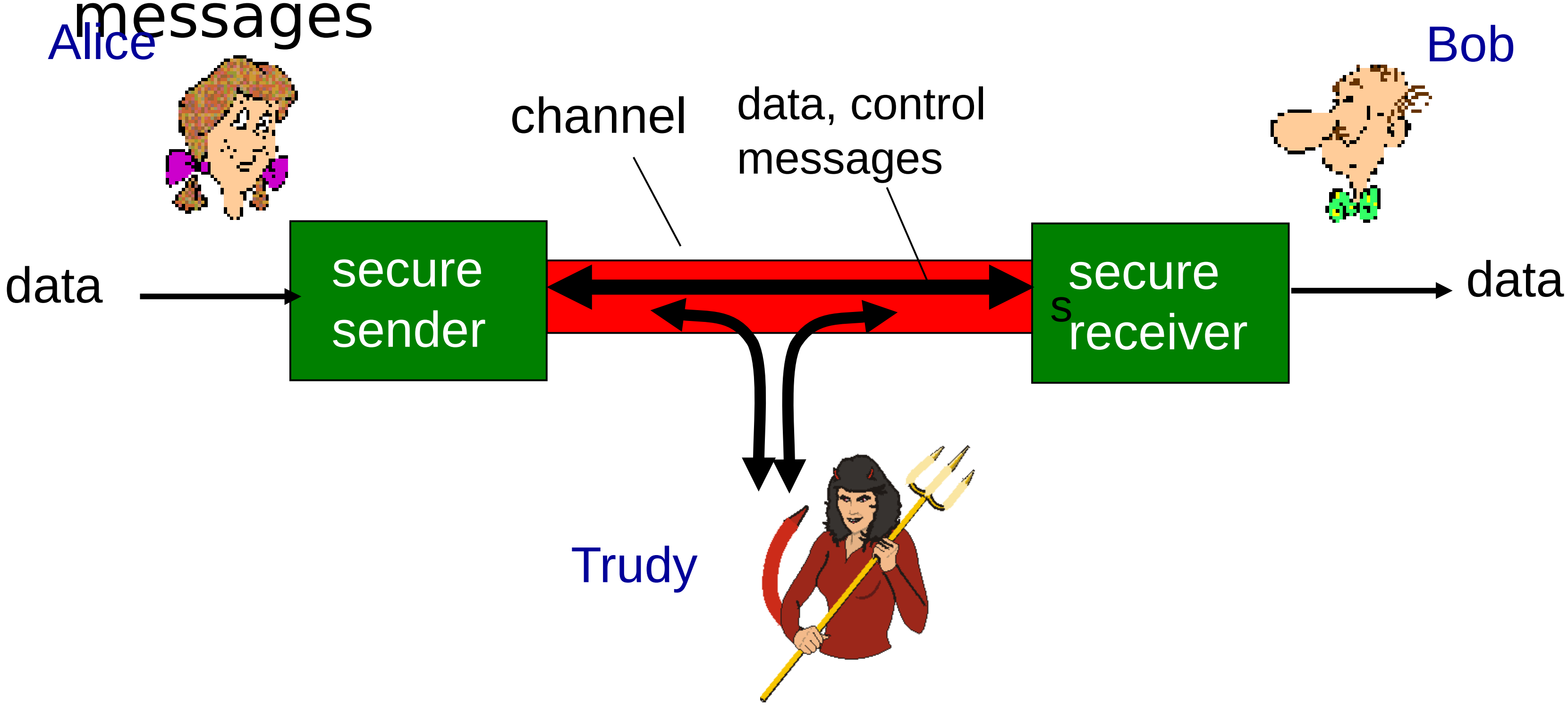
- Why secure communication is important?
- Cryptography Basic Principles
- Intro to symmetrical cryptography (DES)
- Intro to asymmetrical cryptography (RSA)
- Authentication
- Message digest: hash function
- Certification authority
- Secure emails

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
- Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- Availability: services must be accessible and available to users

Friends and enemies: Alice, Bob and Trudy

- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might be 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?

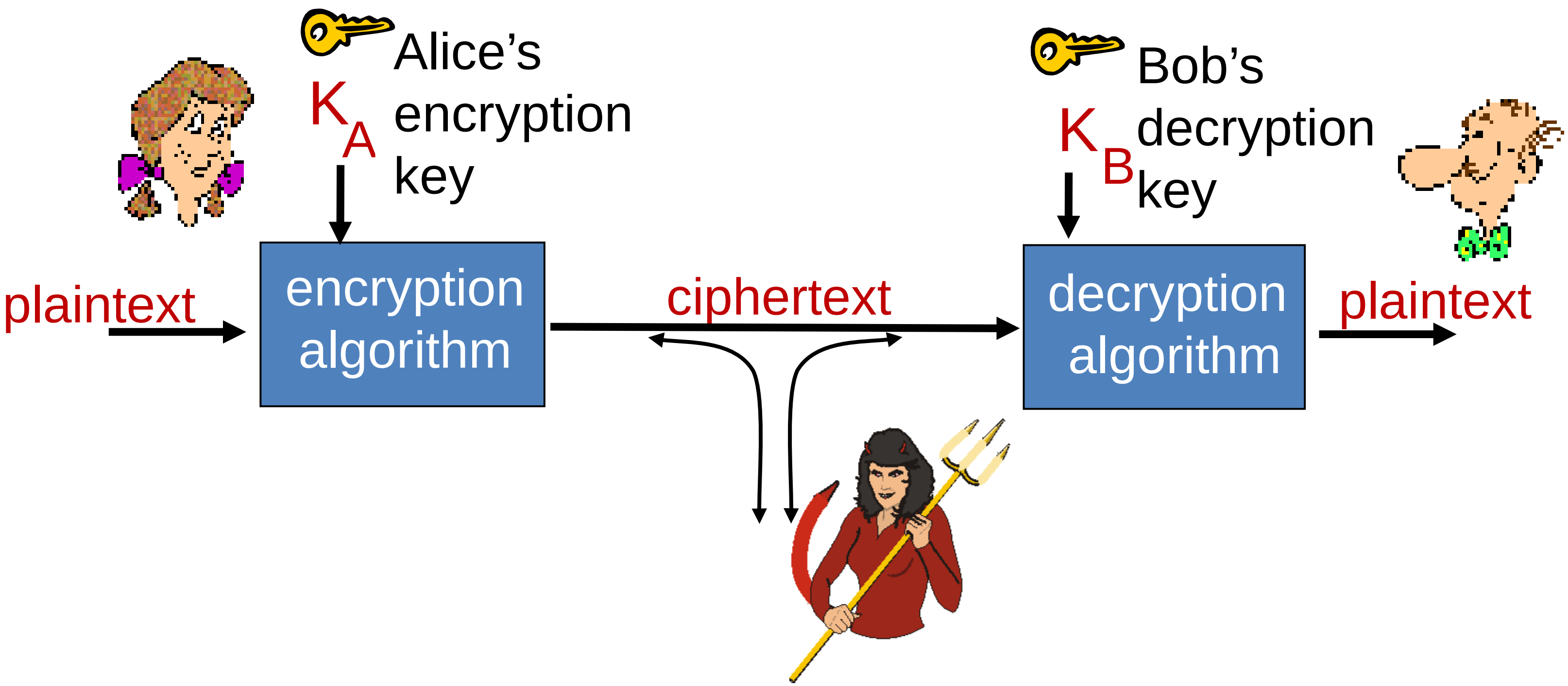
The are bad guys (and girls) out there!

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)

The language of cryptography



m plaintext message

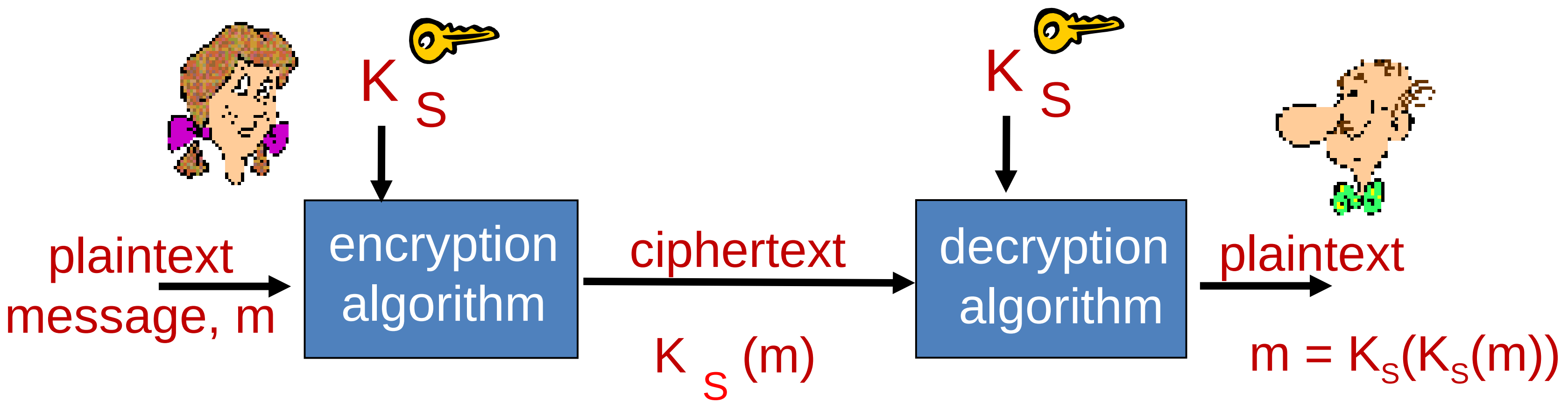
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

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

Symmetric 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

Simple encryption scheme

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
```

Encryption key: mapping from set of 26 letters to set of 26 letters

Symmetric key crypto: DES

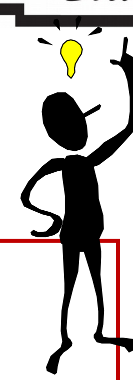
DES: Data Encryption Standard

- 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

Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- 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 crypto

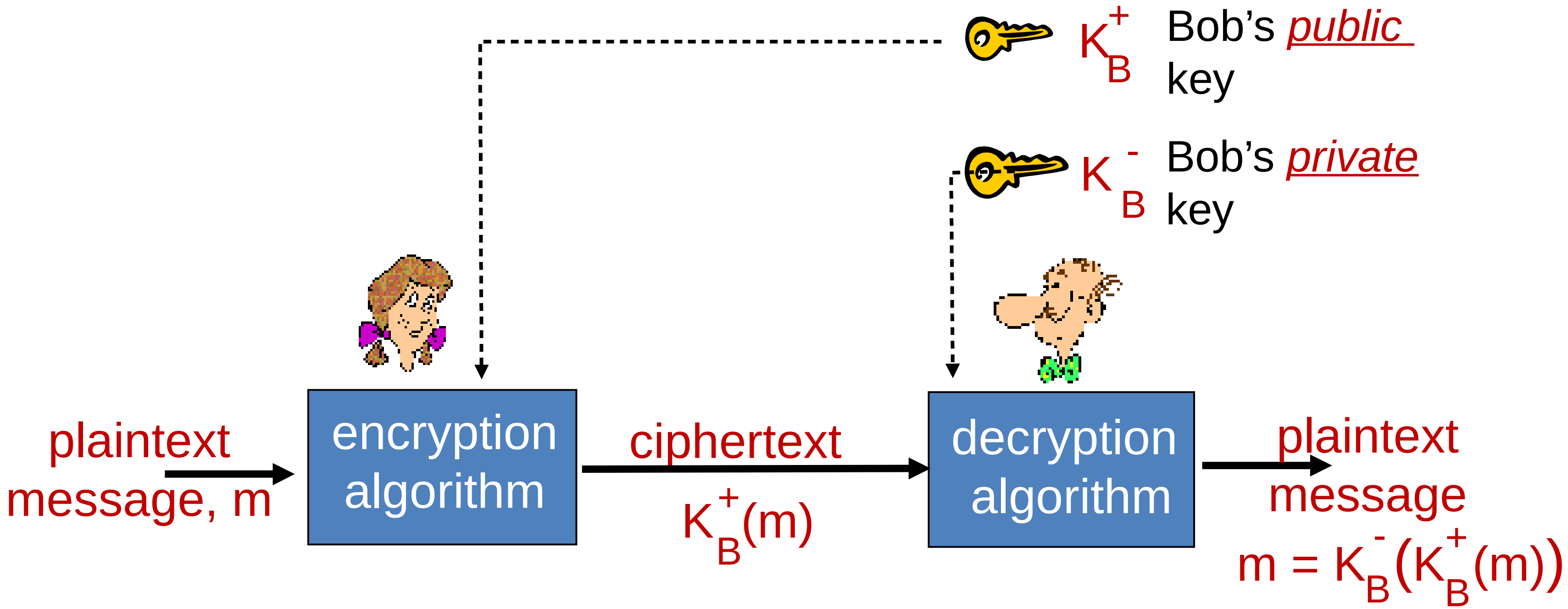
❖ radically different approach [Diffie-Hellman76, RSA78]

❖ sender, receiver do *not* share secret key

❖ *public* encryption key known to *all*

❖ *private* decryption key known only to receiver

Public key cryptography



Public key cryptography 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: important property

The following property will be *very* useful later:

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

result is the same!

RSA in practice: session keys

- RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key cryto to establish secure connection, then establish second key – symmetric session key – for encrypting data

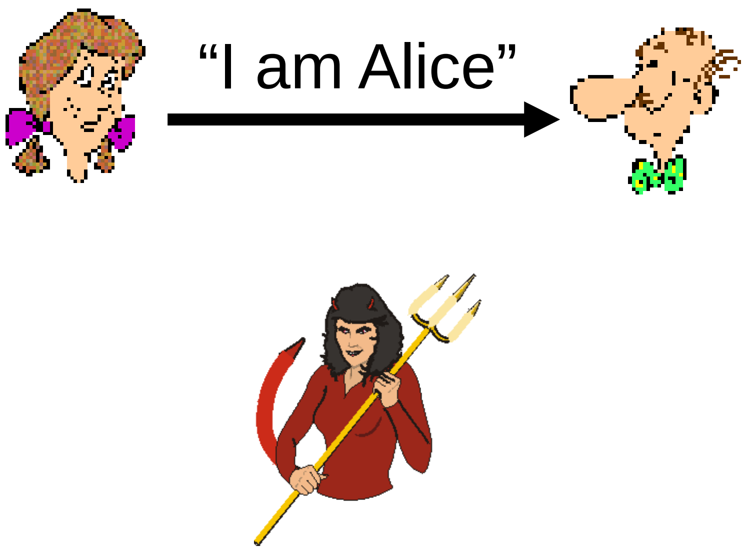
session key, K_S

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

Authentication

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

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

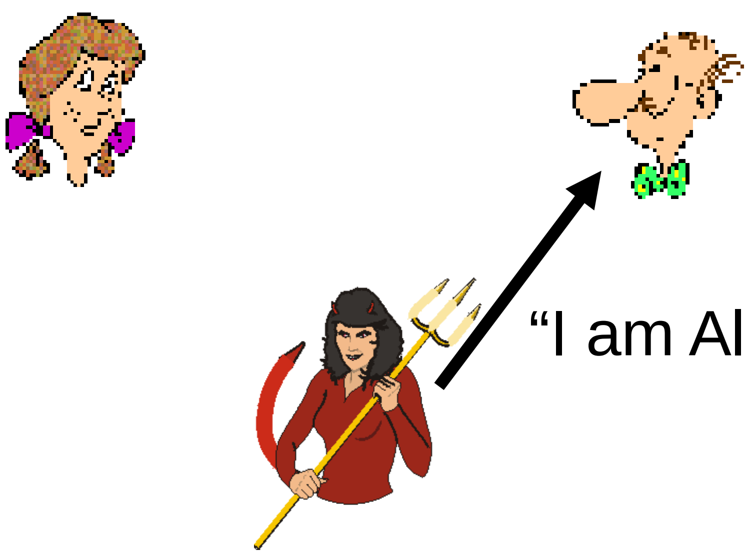


Failure scenario??

Authentication

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

<u>*Protocol ap1.0:*</u> Alice says "I am Alice"



"I am Alice"

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

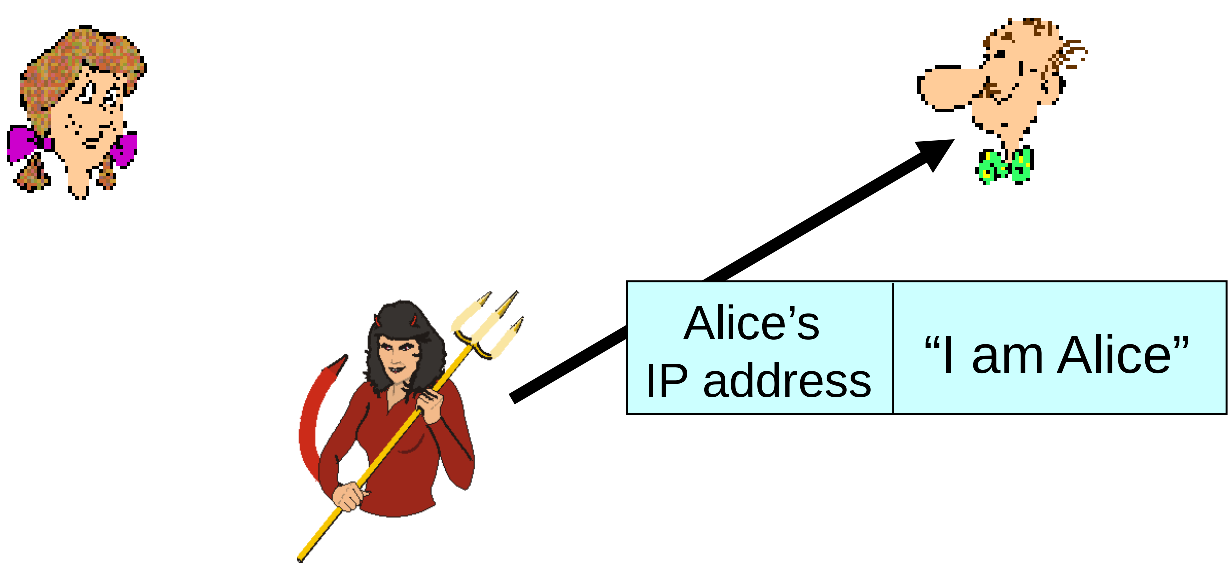
Authentication: another try

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

Failure scenario??

Authentication: another try

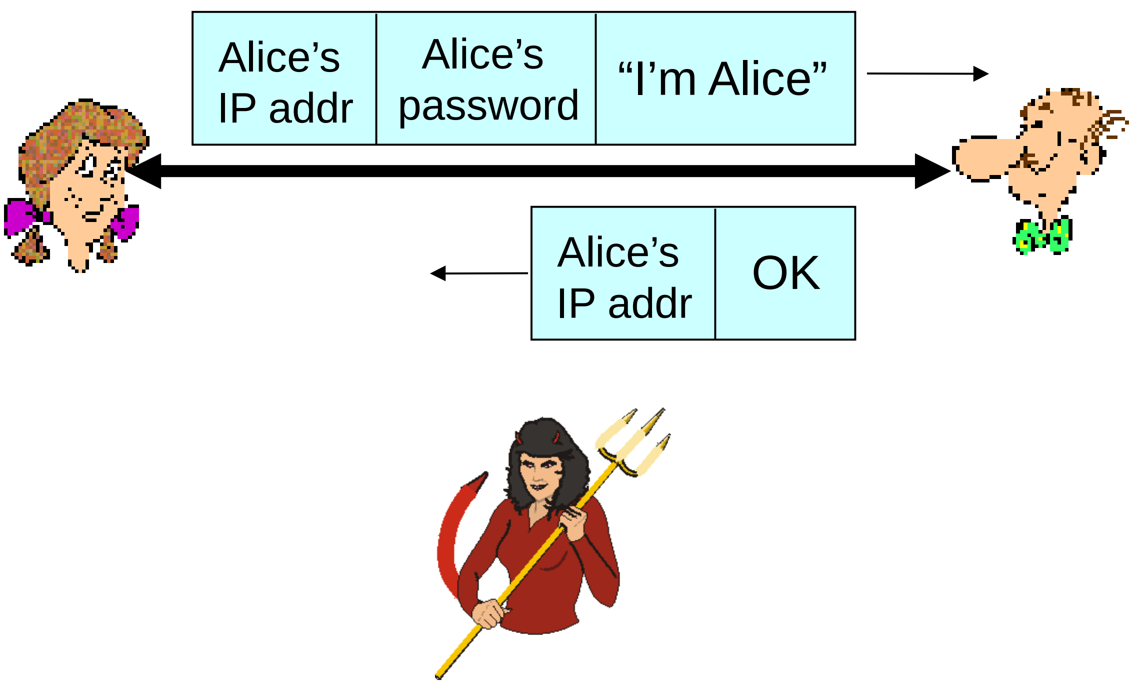
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



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

Authentication: another try

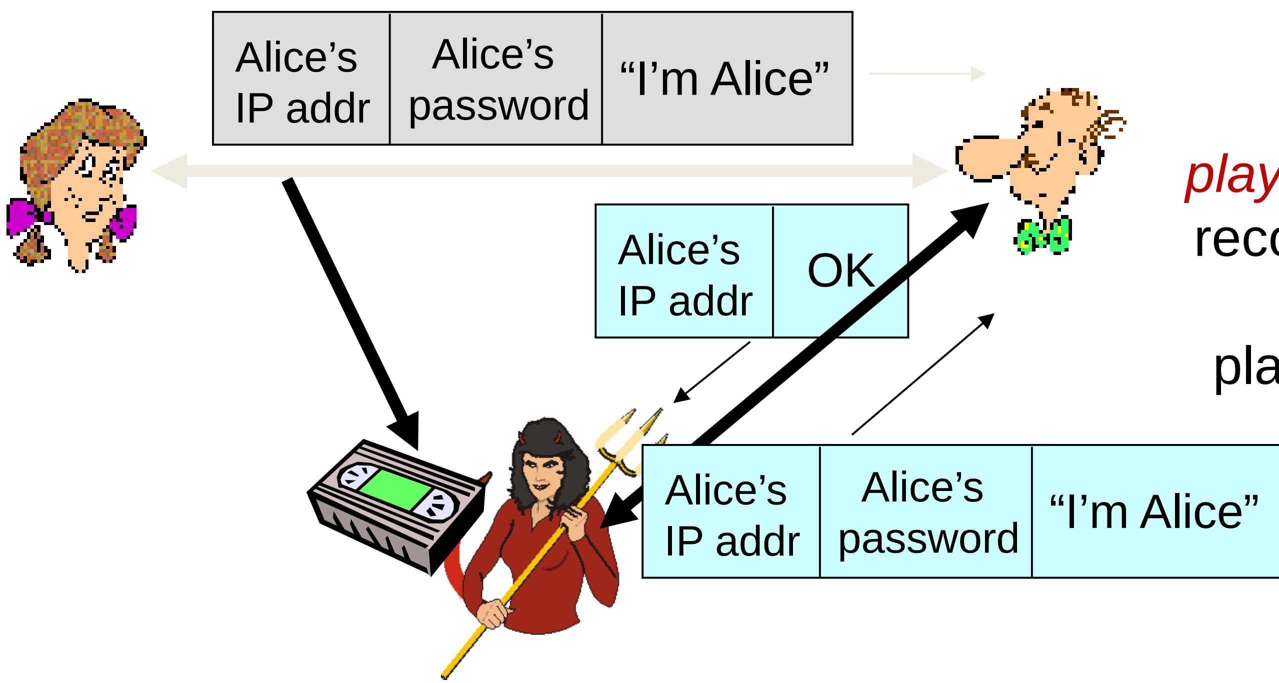
Protocol ap3.0: Alice says “I am Alice” and sends her secret password to “prove” it.



Failure scenario??

Authentication: another try

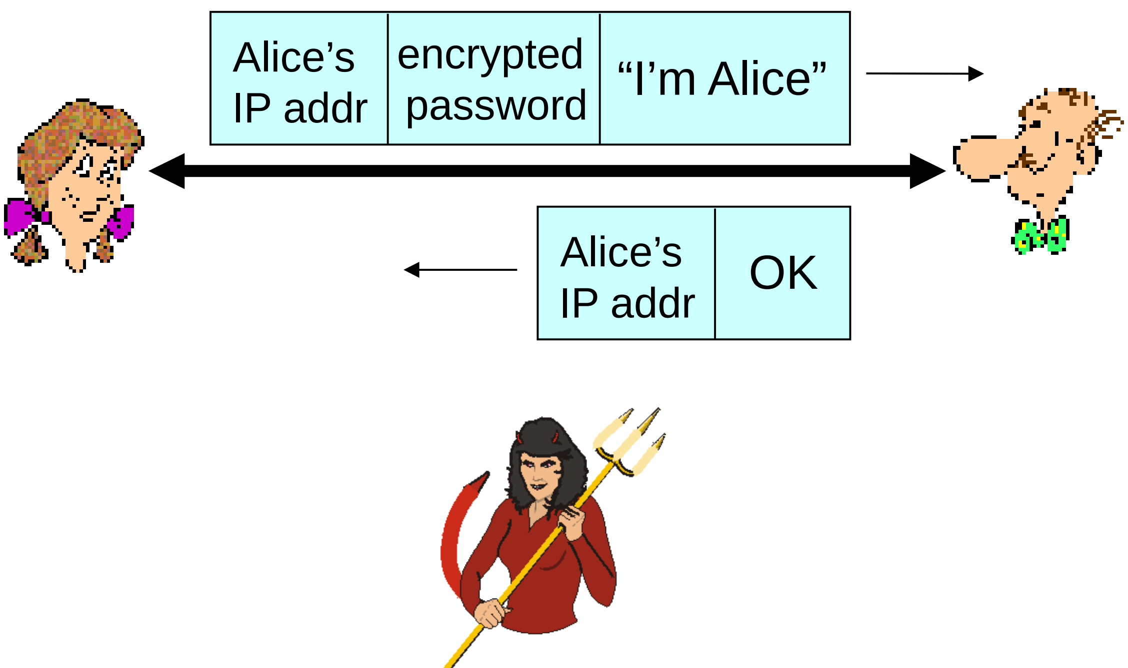
Protocol ap3.0: Alice says “I am Alice” and sends he
secret password to “prove” i



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

Authentication: another try

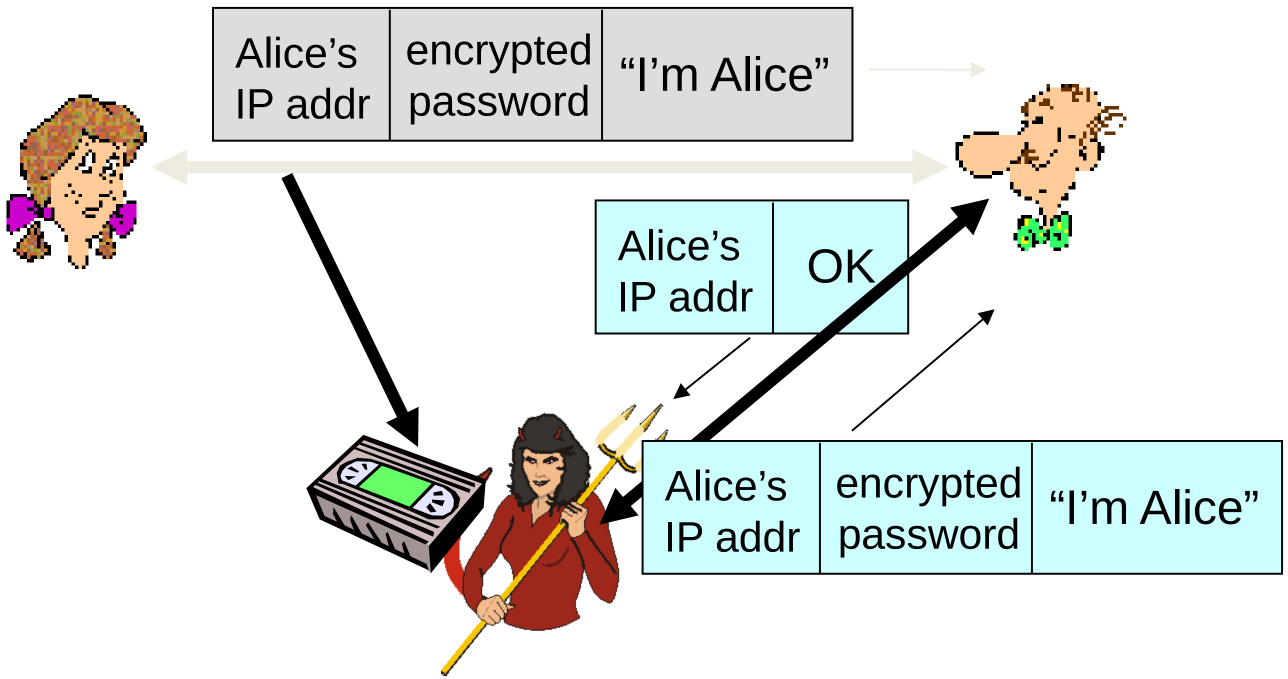
Protocol ap3.1: Alice says “I am Alice” and sends he
encrypted secret password to “prove” i



Failure scenario??

Authentication: another try

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



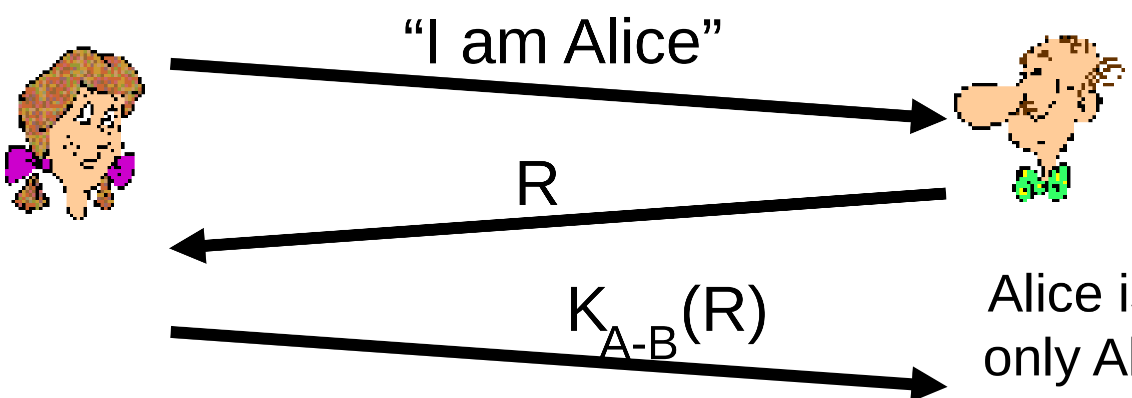
record
and
playback
still works!

Authentication: 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



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

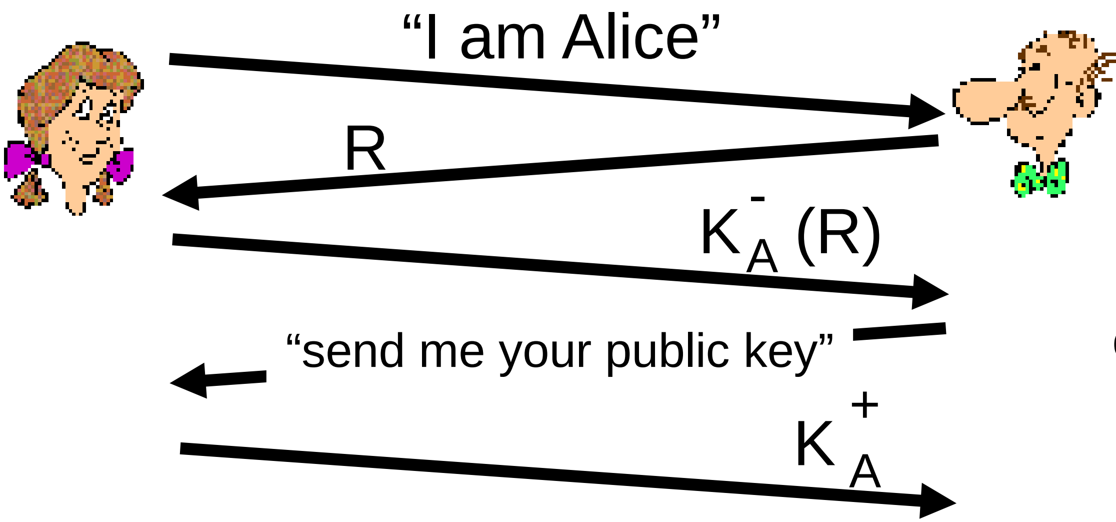
Failures, drawbacks?

Authentication: ap 5.0

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

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

Message integrity: Digital signatures

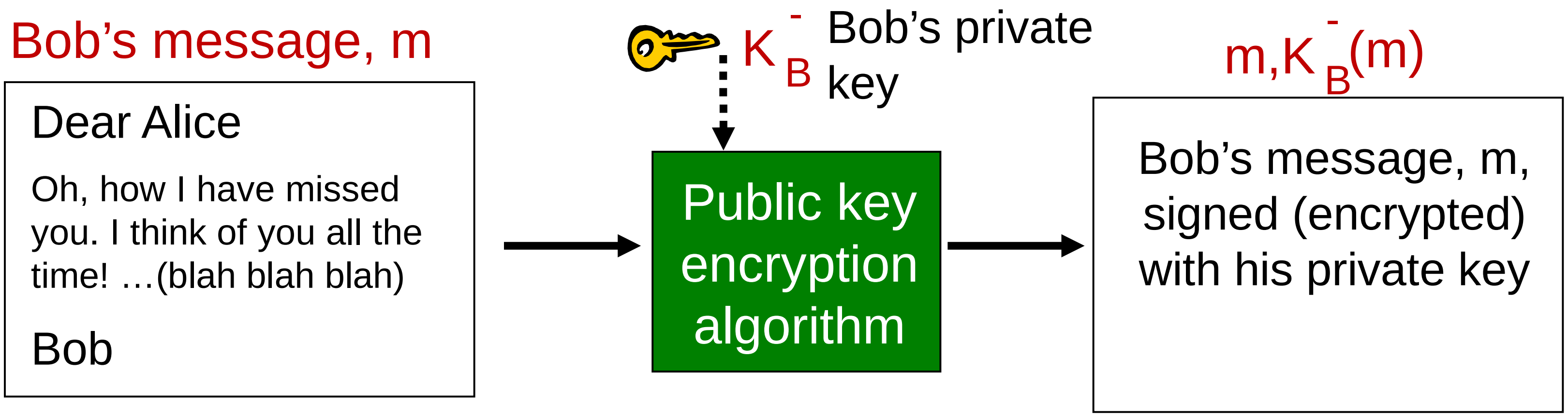
cryptographic technique analogous to hand-written signatures:

- 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

- 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.

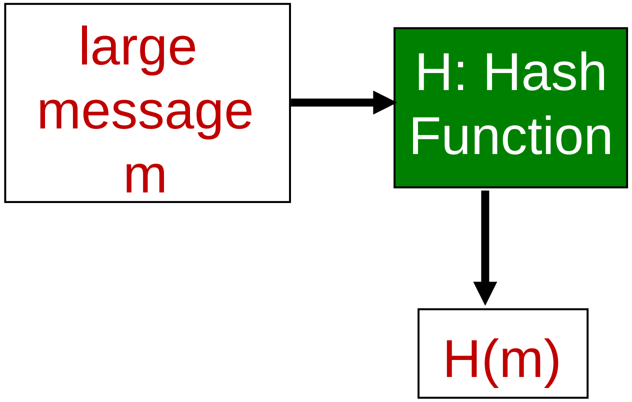
Alice thus verifies that:

- 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 digest



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)

Checksum: poor crypto function

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

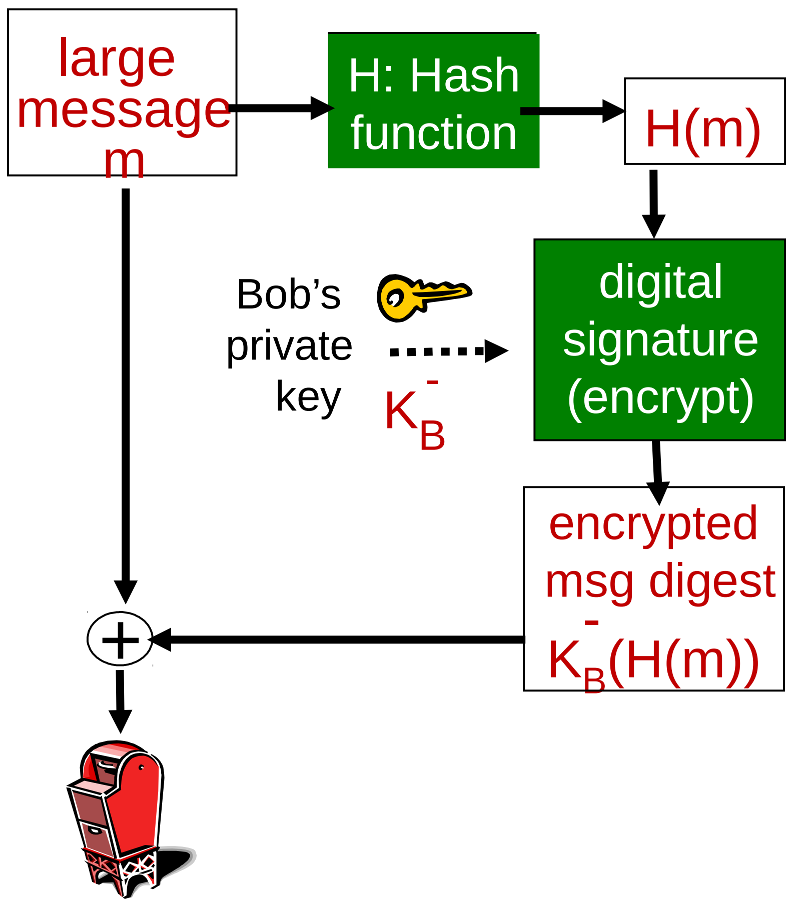
But given message with given hash value, it is easy to find another message with same hash value:

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**

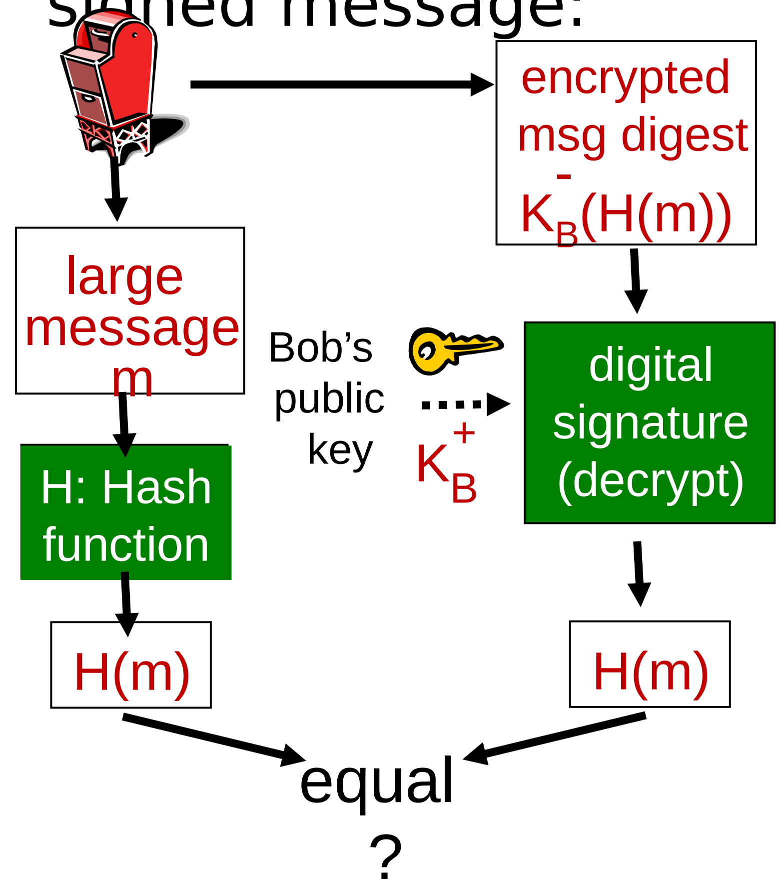
different messages
but identical checksums!

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:

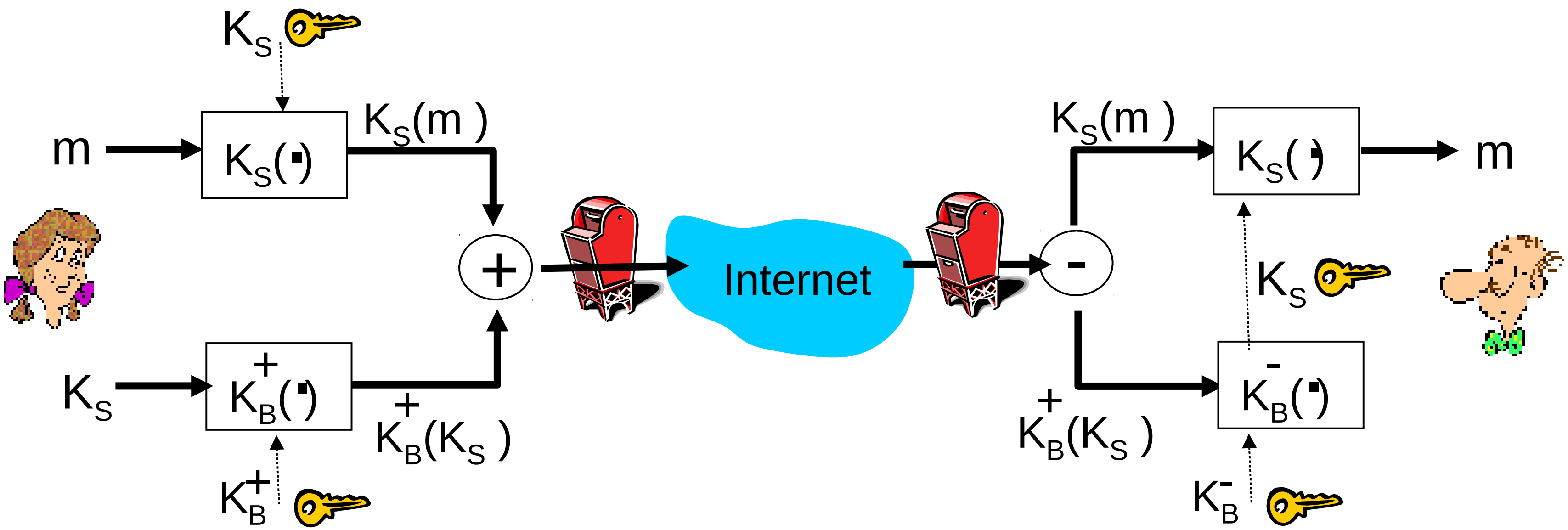


Hash function algorithms

- MD5 hash function widely used (RFC 1321 http://tools.ietf.org/html/rfc1321)
  - 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 (secure hash algorithm) is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Secure email (sender)

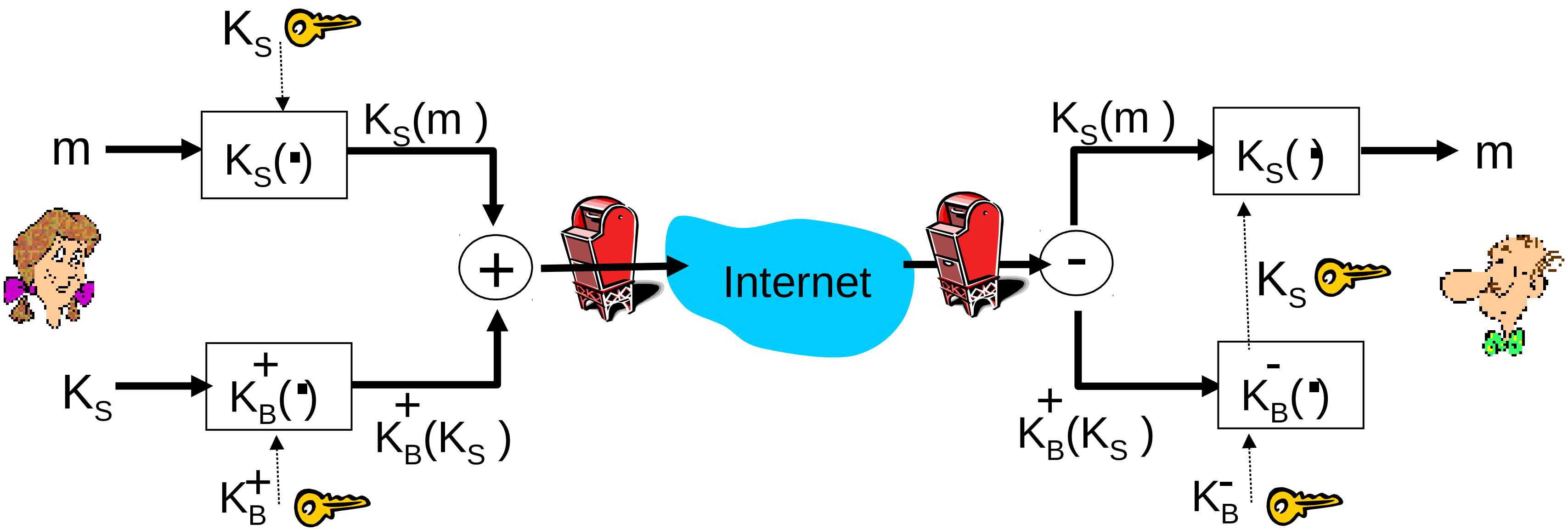
- Alice wants to send confidential e-mail, m, to Bob.



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 email (receiver)

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

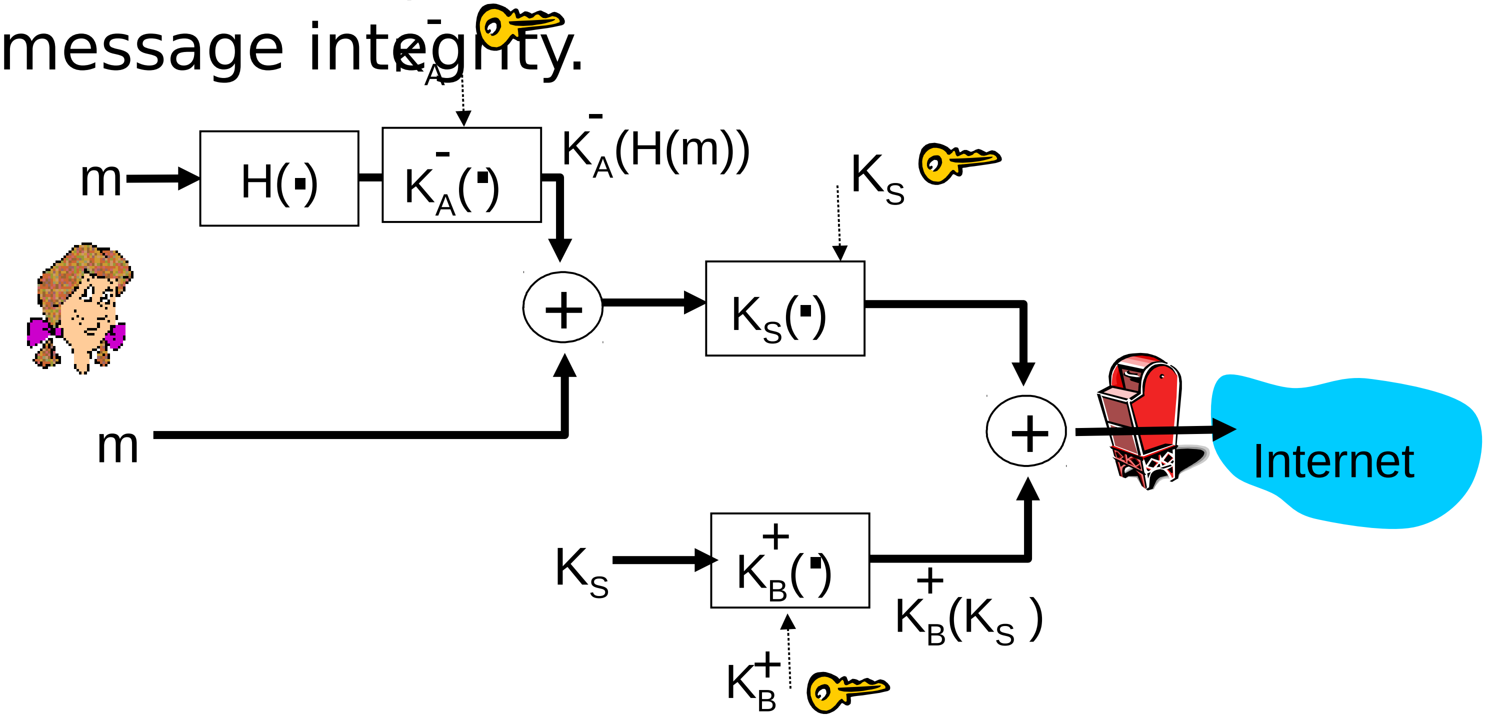


Bob:

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

Secure email (PGP)

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



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