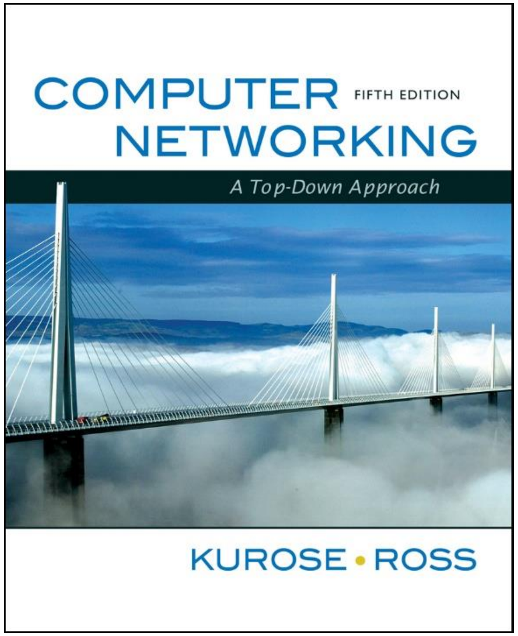
Chapter 8
Network Security



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 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) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form 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

All material copyright 1996-2009
J.F Kurose and K.W. Ross, All Rights Reserved

Computer Networking: A Top Down Approach ,
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Chapter 8: 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 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

What is network security?

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

- sender encrypts message
- receiver decrypts message

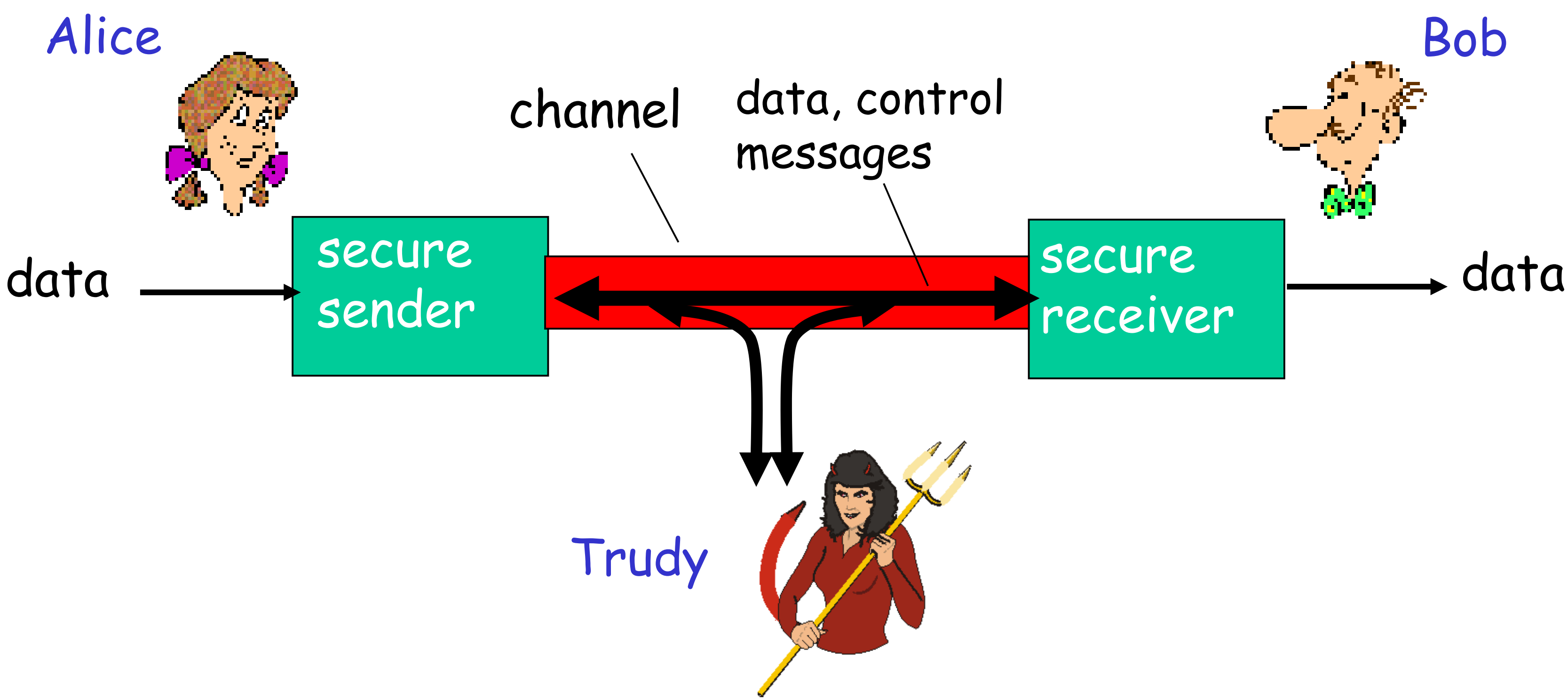
Authentication: sender, receiver want to confirm identity of each other

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?

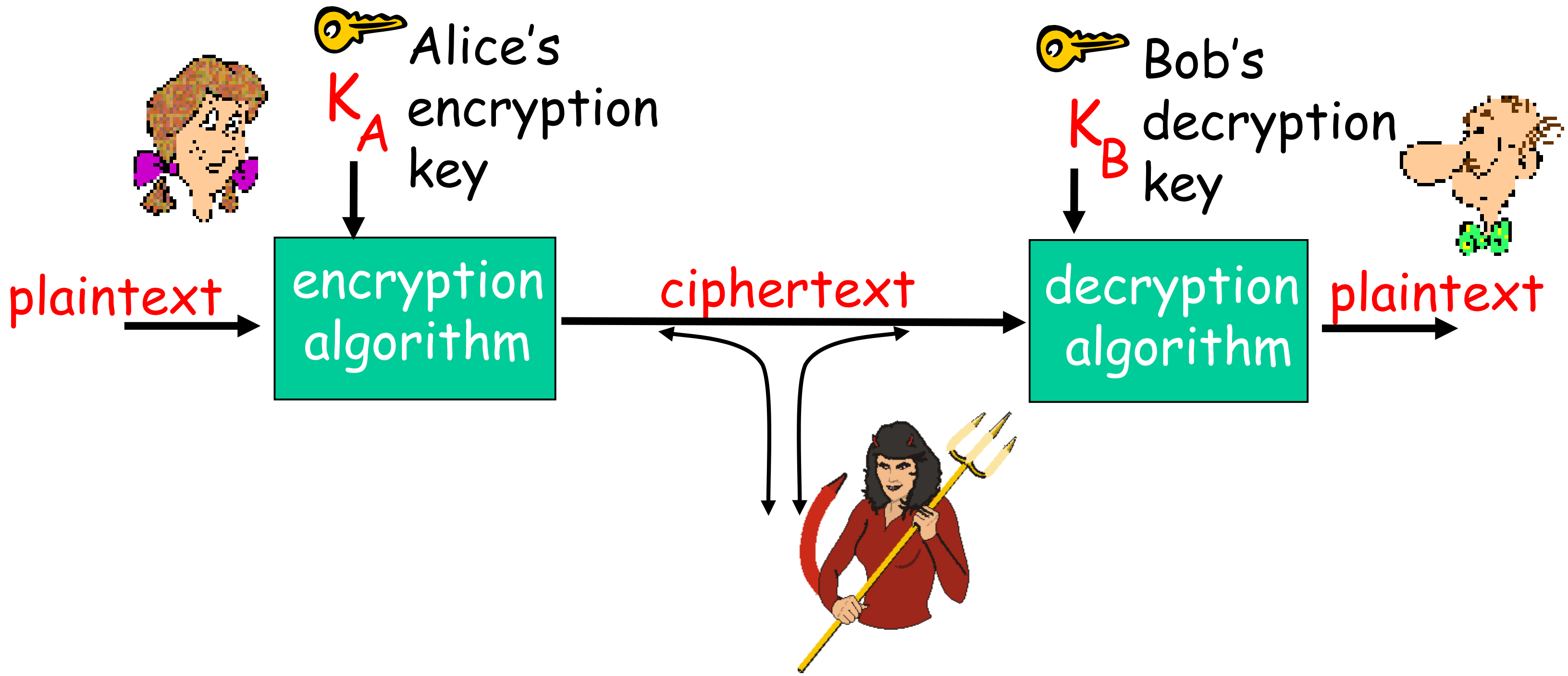
A: A lot! See section 1.6

- *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)

Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

The language of cryptography



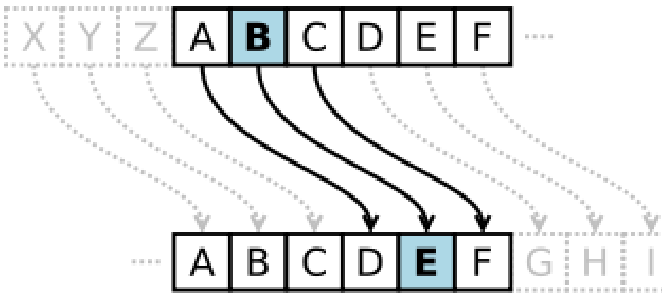
m plaintext message

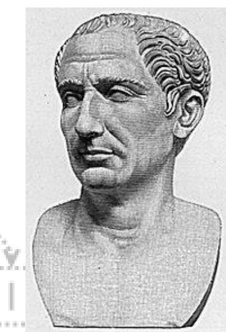
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

历史上的密码

- 从凯撒密码到Enigma






Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

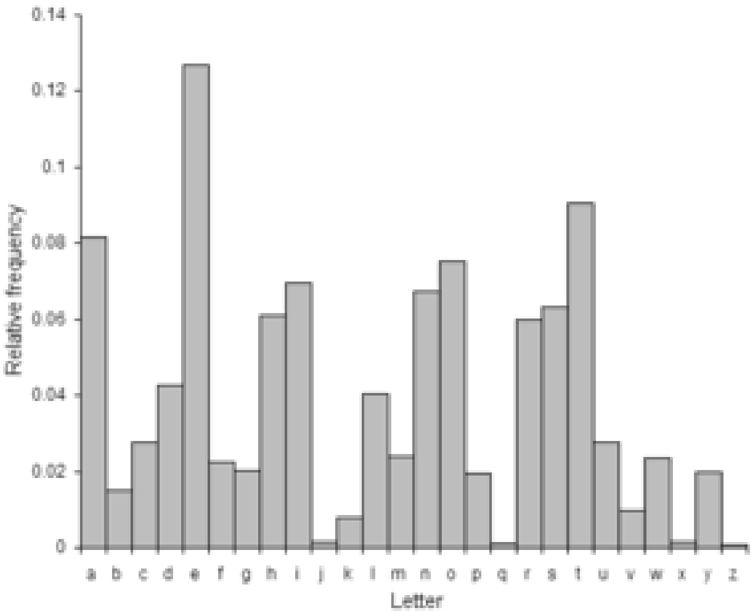
```
 plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

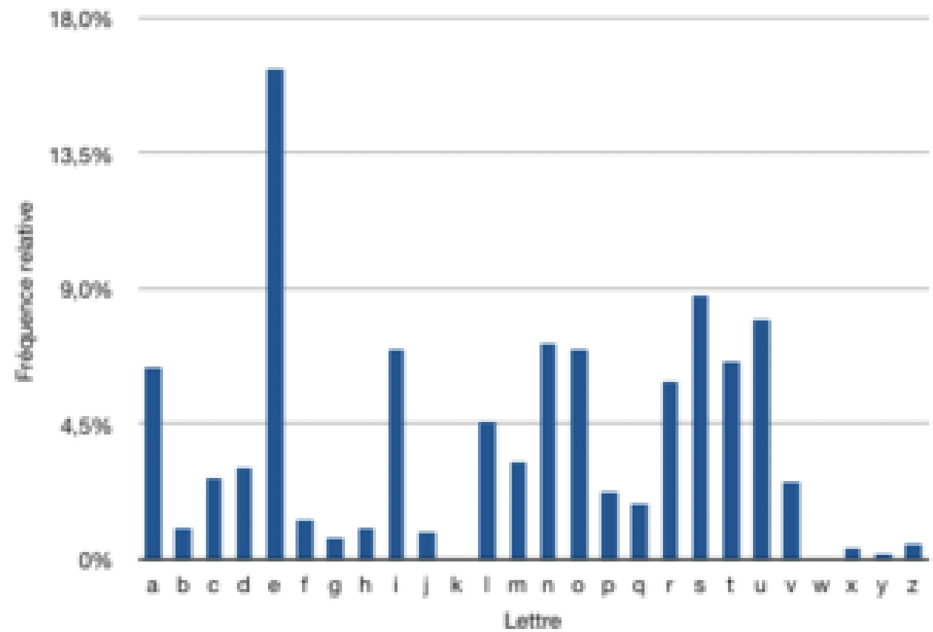
```
 Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Key: the mapping from the set of 26 letters to the set of 26 letters

Statistical analysis



English



French

Polyalphabetic encryption

- n monoalphabetic cyphers, $M_1, M_2, \ldots, M_n$
- Cycling pattern:
  - e.g., n=4, M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2;
- For each new plaintext symbol, use subsequent monoalphabetic pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4
- Key: the n ciphers and the cyclic pattern

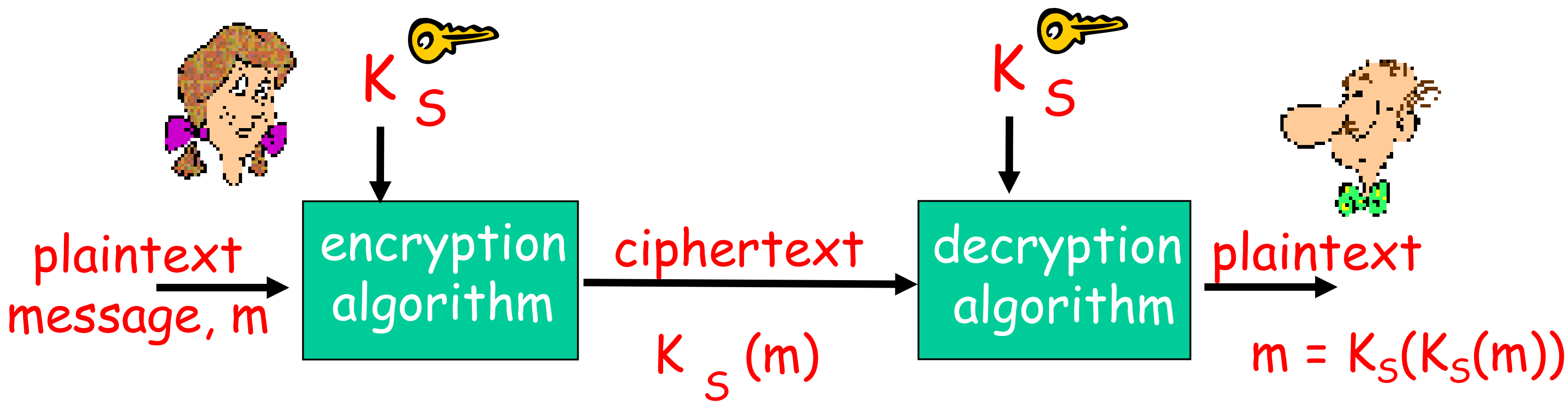
Breaking an encryption scheme

- Cipher-text only attack: Trudy has ciphertext that she can analyze
- Two approaches:
  - Search through all keys: must be able to differentiate resulting plaintext from gibberish
  - Statistical analysis
- Known-plaintext attack: trudy has some plaintext corresponding to some ciphertext
  - eg, in monoalphabetic cipher, trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack: trudy can get the cyphertext for some chosen plaintext

Types of Cryptography

- Crypto often uses keys:
  - Algorithm is known to everyone
  - Only "keys" are secret
- Public key cryptography
  - Involves the use of two keys
- Symmetric key cryptography
  - Involves the use one key
- Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

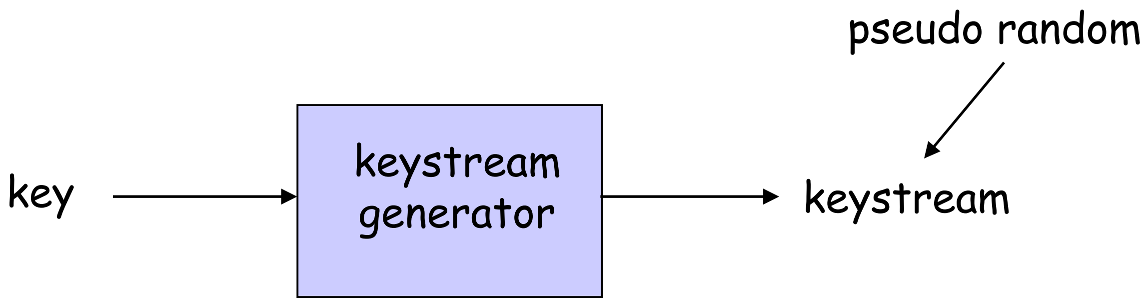
- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

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

Stream Ciphers



- Combine each bit of keystream with bit of plaintext to get bit of ciphertext
- m(i) = ith bit of message
- ks(i) = ith bit of keystream
- c(i) = ith bit of ciphertext
- c(i) = ks(i) $\oplus$ m(i)   ($\oplus$ = exclusive or)
- m(i) = ks(i) $\oplus$ c(i)

RC4 Stream Cipher

- RC4 is a popular stream cipher
  - Extensively analyzed and considered good
  - Key can be from 1 to 256 bytes
  - Used in WEP for 802.11
  - Can be used in SSL

Block ciphers

- Message to be encrypted is processed in blocks of k bits (e.g., 64-bit blocks).
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext

Example with k=3:

input	output
000	110
001	111
010	101
011	100

input	output
100	011
101	010
110	000
111	001

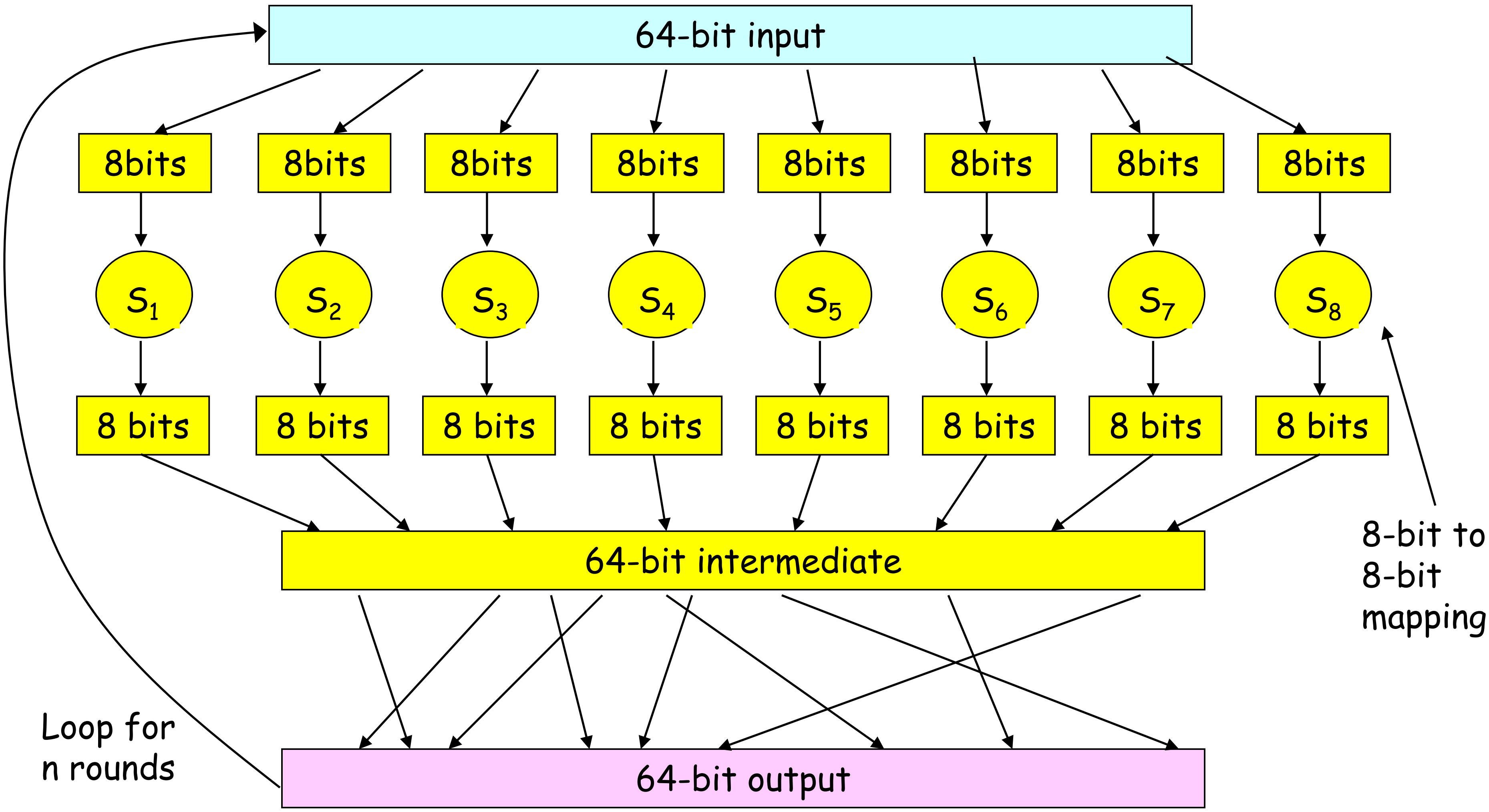
What is the ciphertext for 010110001111 ?

Block ciphers

- How many possible mappings are there for k=3?
  - How many 3-bit inputs?
  - How many permutations of the 3-bit inputs?
  - Answer: 40,320 ; not very many!
- In general, $2^k!$ mappings;   huge for k=64
- Problem:
  - Table approach requires table with 2^{64} entries, each entry with 64 bits
- Table too big: instead use function that simulates a randomly permuted table

Prototype function

From Kaufman et al



Why rounds in prototpe?

- If only a single round, then one bit of input affects at most 8 bits of output.
- In 2nd round, the 8 affected bits get scattered and inputted into multiple substitution boxes.
- How many rounds?
    - How many times do you need to shuffle cards
    - Becomes less efficient as n increases

Encrypting a large message

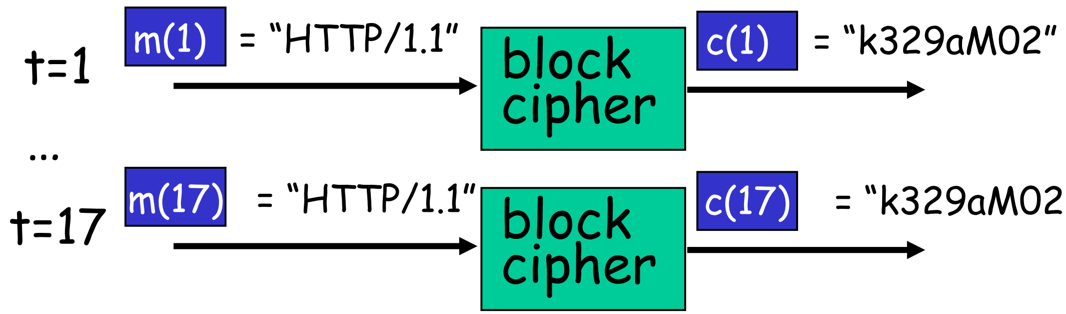
- Why not just break message in 64-bit blocks, encrypt each block separately?
  - If same block of plaintext appears twice, will give same cyphertext.
- How about:
  - Generate random 64-bit number r(i) for each plaintext block m(i)
  - Calculate $c(i) = K_S(m(i) \oplus r(i))$
  - Transmit c(i), r(i), i=1,2,…
  - At receiver: $m(i) = K_S(c(i)) \oplus r(i)$
  - Problem: inefficient, need to send c(i) and r(i)

Cipher Block Chaining (CBC)

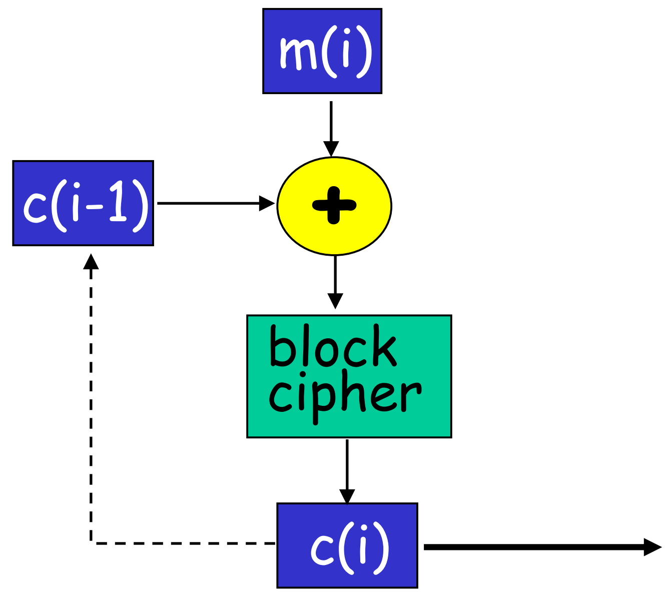
- CBC generates its own random numbers
  - Have encryption of current block depend on result of previous block
  - $c(i) = K_S(\ m(i) \oplus c(i-1)\ )$
  - $m(i) = K_S(\ c(i)) \oplus c(i-1)$
- How do we encrypt first block?
  - Initialization vector (IV): random block = c(0)
  - IV does not have to be secret
- Change IV for each message (or session)
  - Guarantees that even if the same message is sent repeatedly, the ciphertext will be completely different each time

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?



Symmetric key crypto: DES

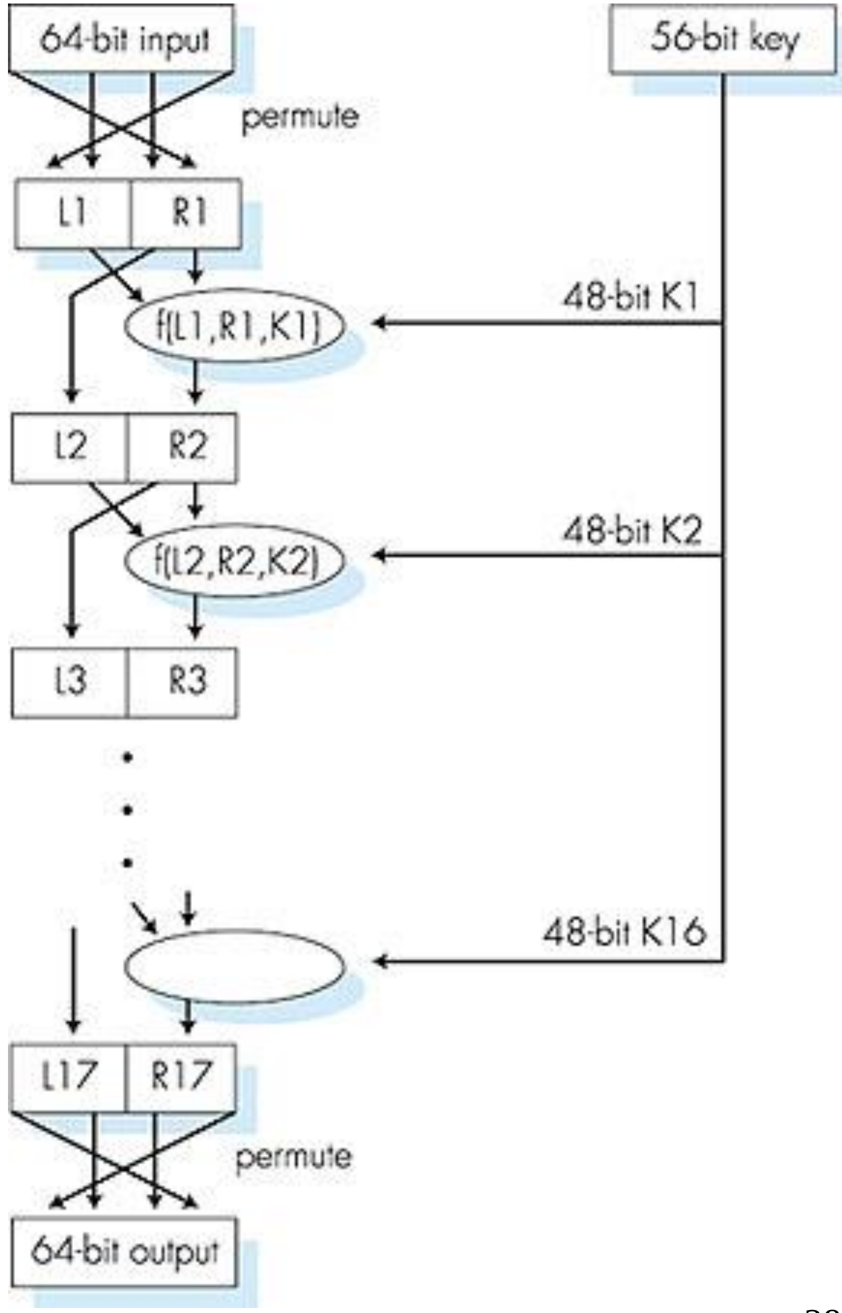
DES: Data Encryption Standard

- 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 (actually encrypt, decrypt, encrypt)

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

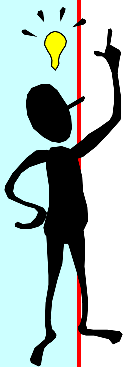
- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

symmetric key crypto

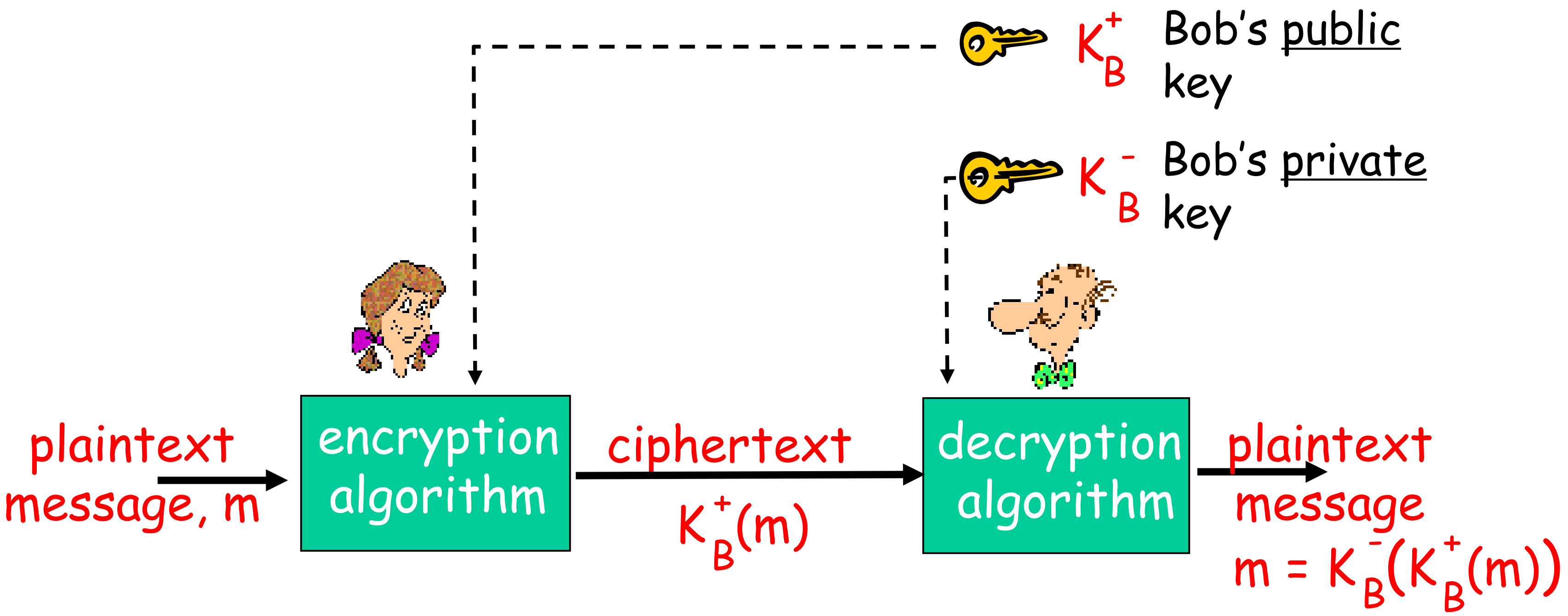
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

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

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

- A message is a bit pattern.
- A 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 cyphertext).

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 $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, decryption

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

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

   $c = m^e \bmod n$

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$

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).

Encrypting 8-bit messages.

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

	c	c^d	$m = c^d \bmod n$
decrypt:	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,
  $c^d \bmod n = (m^e \bmod n)^d \bmod n$
  $= m^{ed} \bmod n$
  关键！ $= m^{(ed \bmod z)} \bmod n$
  $= m^1 \bmod n$
  $= m$

RSA: another important property

The following property will be *very* useful later:

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

Result is the same!

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

Follows directly from modular arithmetic:

$$
\begin{aligned}
(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\
&= m^{de} \bmod n \\
&= (m^d \bmod n)^e \bmod n
\end{aligned}
$$

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.

Generating RSA keys

- Have to find big primes p and q
- Approach: make good guess then apply testing rules (see Kaufman)

Session keys

- Exponentiation is computationally intensive
- DES is at least 100 times faster than RSA

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

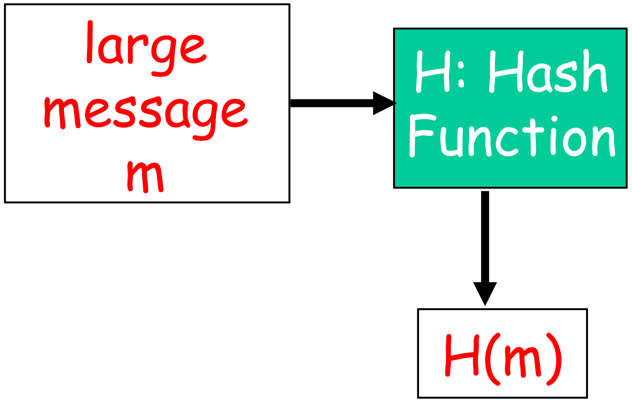
Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

Message Integrity

- Allows communicating parties to verify that received messages are authentic.
  - Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been replayed
  - Sequence of messages is maintained
- Let's first talk about message digests

Message Digests



- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"

- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
  - Seemingly random output

Internet checksum: poor message digest

Internet checksum has some properties of hash function:

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

- But given message with given hash value, it is easy to find another message with same hash value.
- Example: Simplified checksum: add 4-byte chunks at a time:

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

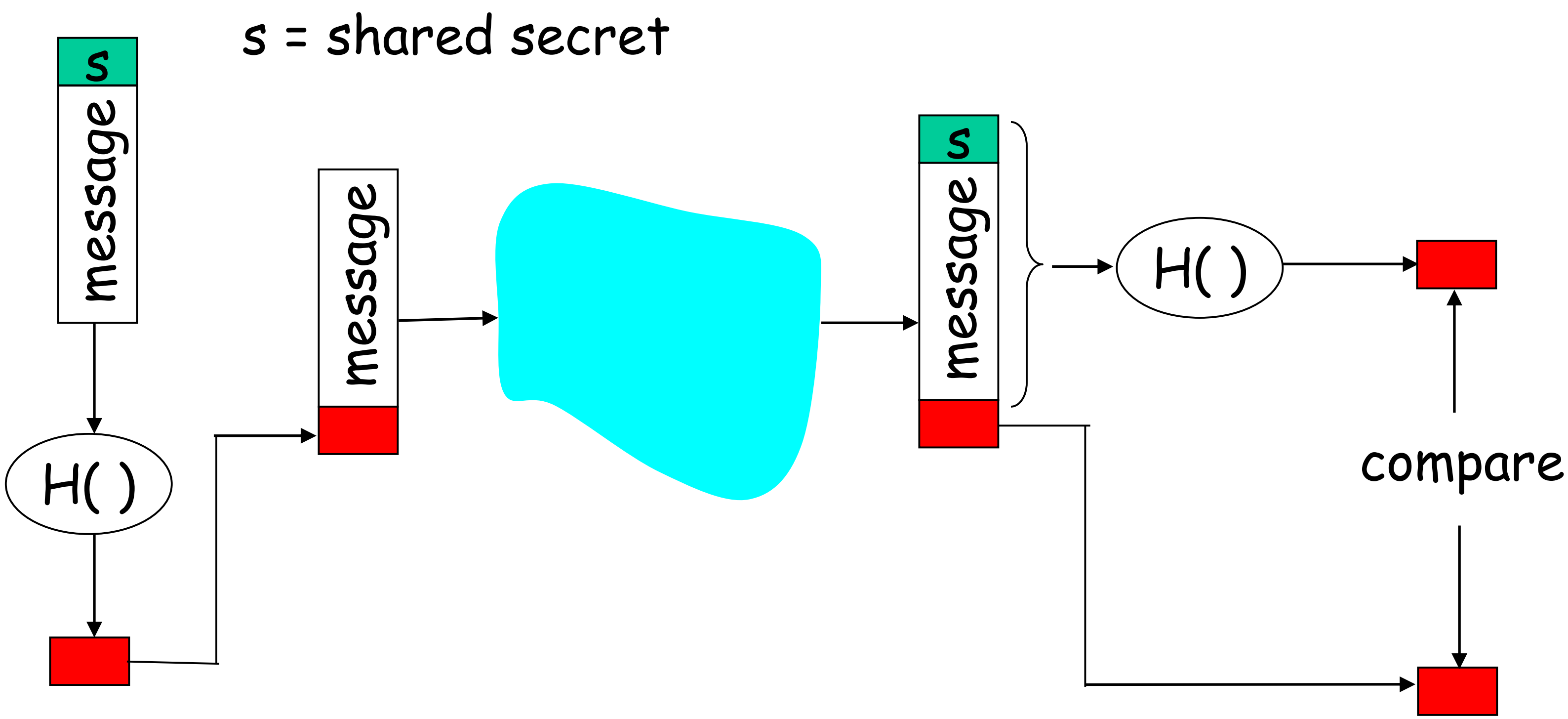
message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Message Authentication Code (MAC)



- ***Authenticates sender***
- ***Verifies message integrity***
- No encryption !
- Also called "keyed hash"
- Notation: $MD_m = H(s||m)$; send $m||MD_m$

HMAC

- Popular MAC standard
- Addresses some subtle security flaws

1. Concatenates secret to front of message.
2. Hashes concatenated message
3. Concatenates the secret to front of digest
4. Hashes the combination again.

Example: OSPF

- Recall that OSPF is an intra-AS routing protocol
- Each router creates map of entire AS (or area) and runs shortest path algorithm over map.
- Router receives link-state advertisements (LSAs) from all other routers in AS.

Attacks:

- Message insertion
- Message deletion
- Message modification

- How do we know if an OSPF message is authentic?

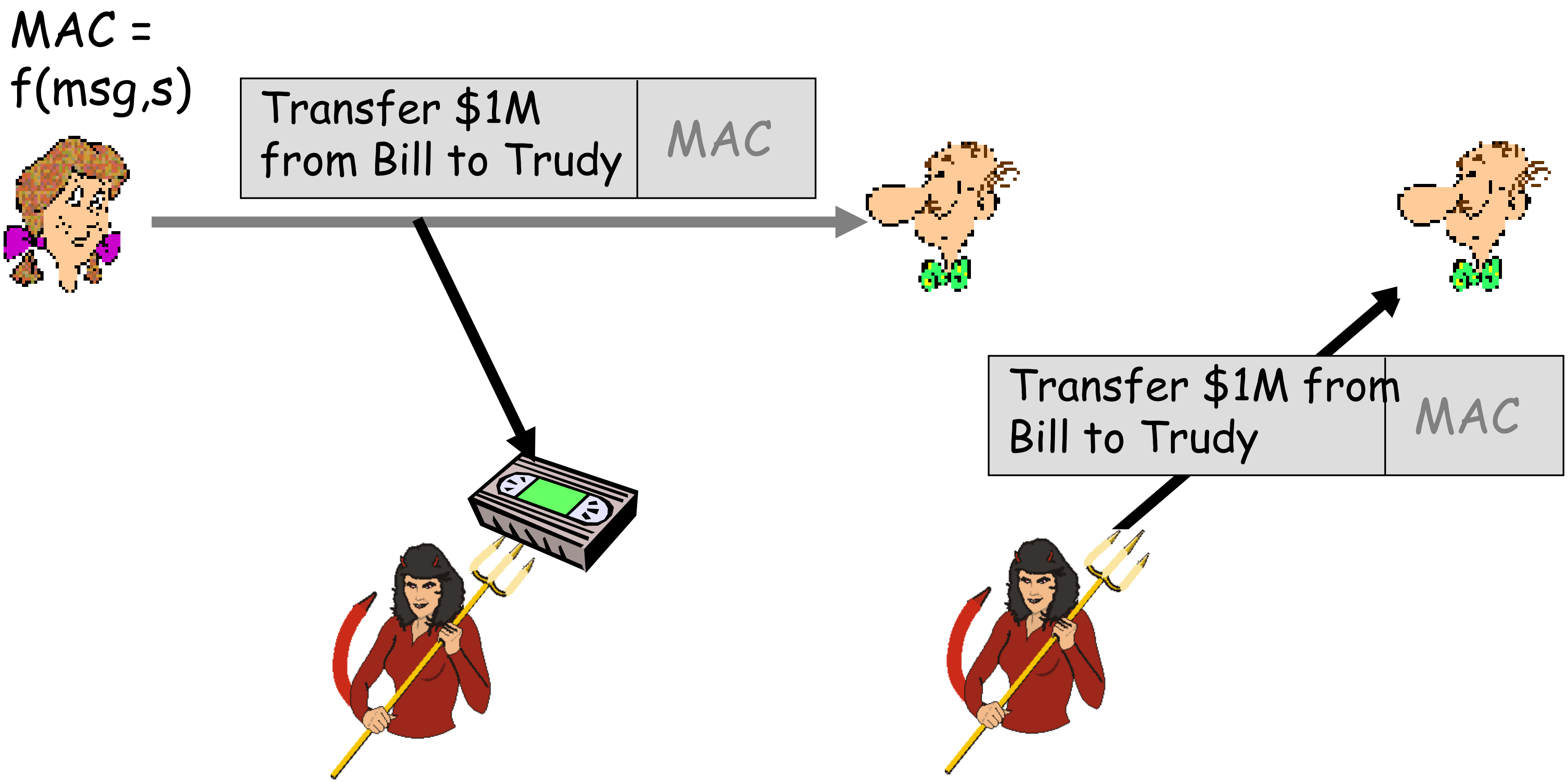
OSPF Authentication

- Within an Autonomous System, routers send OSPF messages to each other.
- OSPF provides authentication choices
  - No authentication
  - Shared password: inserted in clear in 64-bit authentication field in OSPF packet
  - Cryptographic hash
- Cryptographic hash with MD5
  - 64-bit authentication field includes 32-bit sequence number
  - MD5 is run over a concatenation of the OSPF packet and shared secret key
  - MD5 hash then appended to OSPF packet; encapsulated in IP datagram

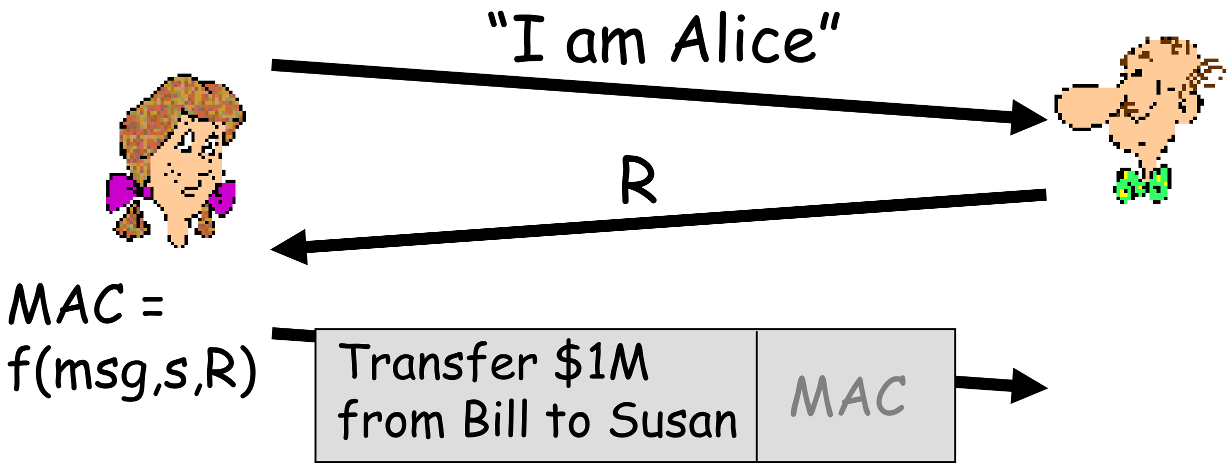
End-point authentication

- Want to be sure of the originator of the message - *end-point authentication*.
- Assuming Alice and Bob have a shared secret, will MAC provide end-point authentication.
  - We do know that Alice created the message.
  - But did she send it?

Playback attack



Defending against playback attack: nonce



Digital Signatures

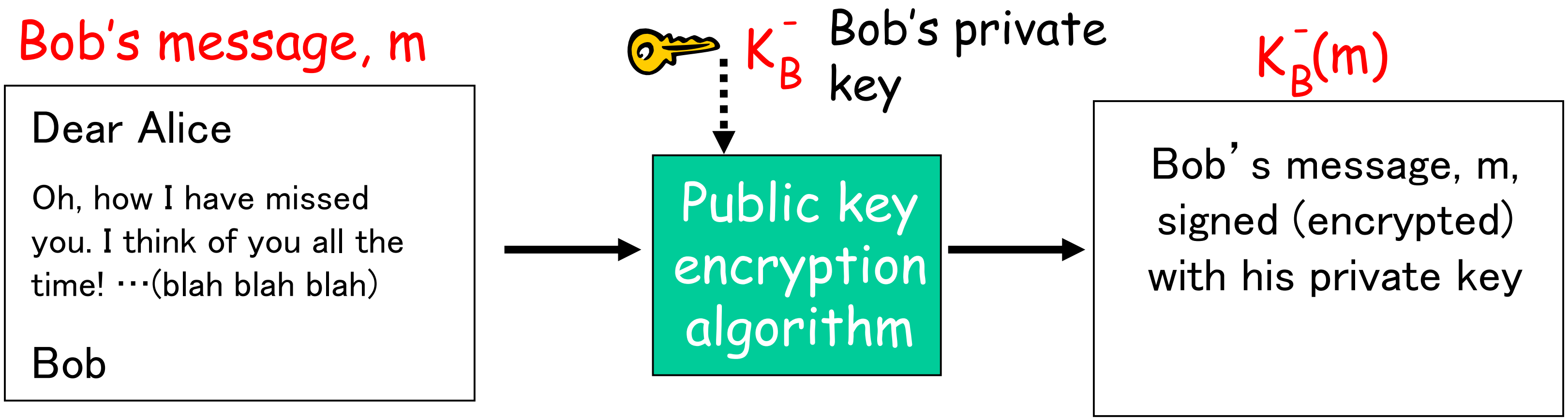
Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- Goal is similar to that of a MAC, except now use public-key cryptography
- 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:

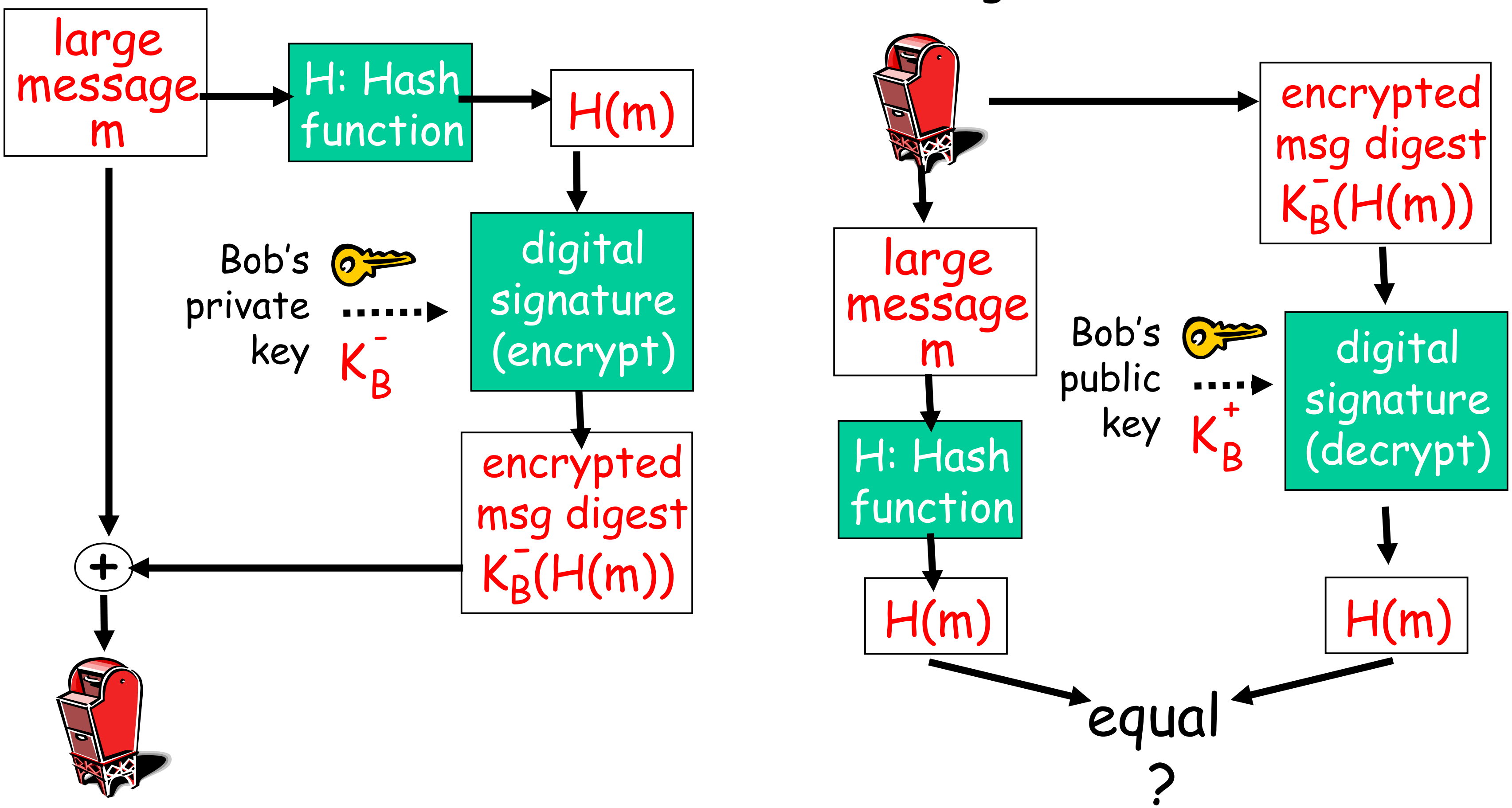
- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:



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:

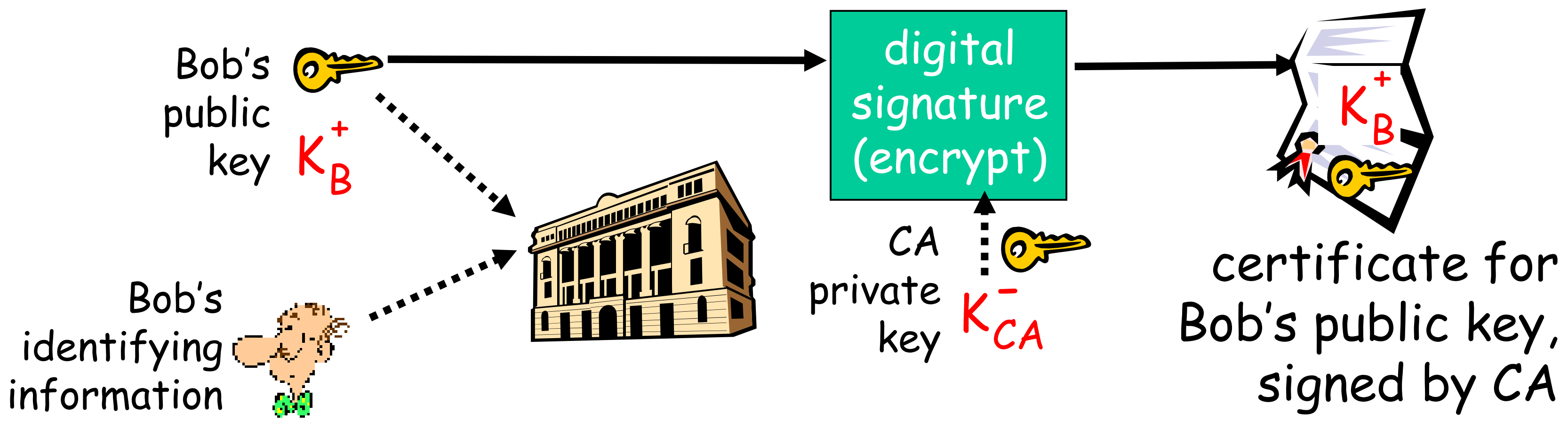
- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

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 pizzas to Bob.
  - Bob doesn't even like Pepperoni

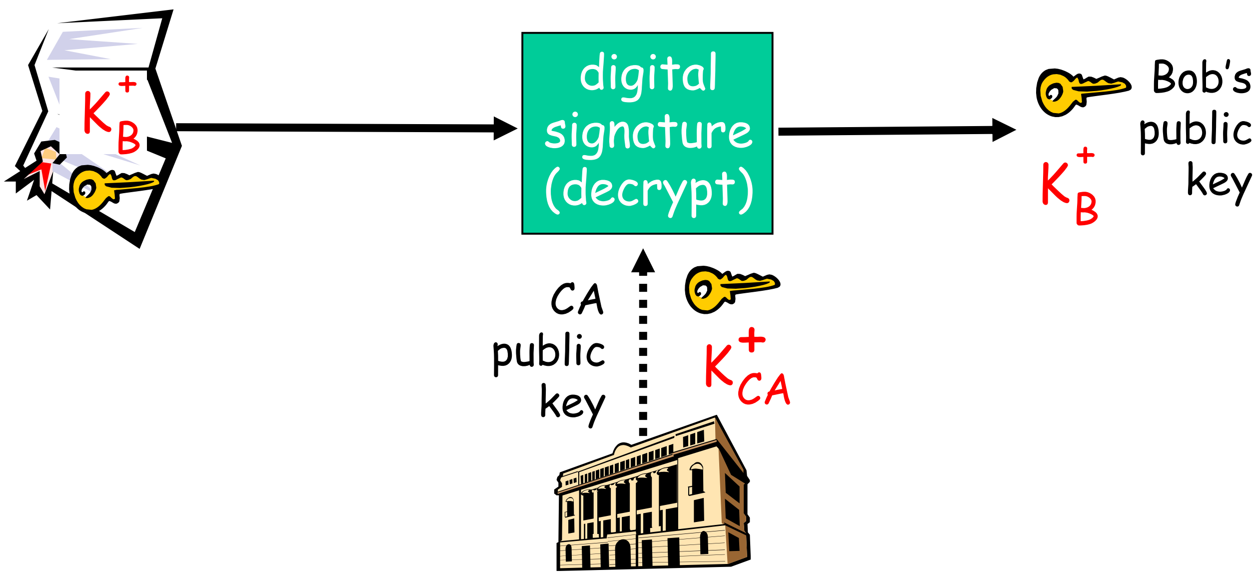
Certification Authorities

- 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"



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



Certificates: summary

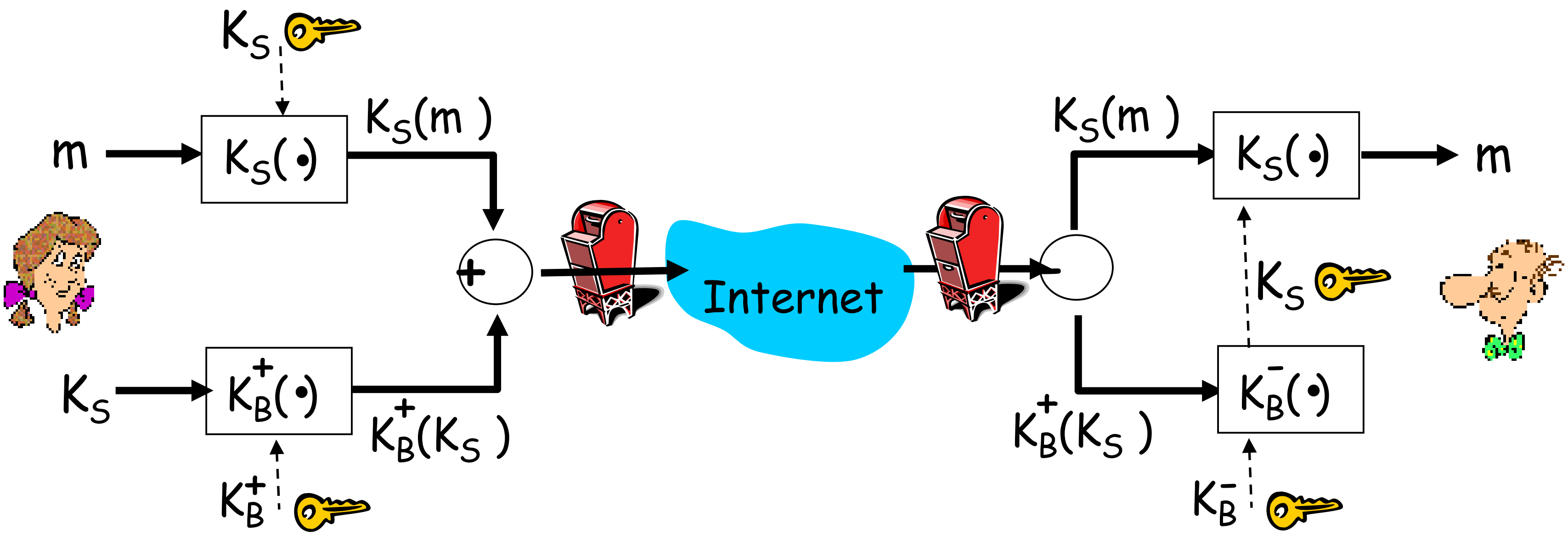
- Primary standard X.509 (RFC 2459)
- Certificate contains:
  - Issuer name
  - Entity name, address, domain name, etc.
  - Entity's public key
  - Digital signature (signed with issuer's private key)
- Public-Key Infrastructure (PKI)
  - Certificates and certification authorities
  - Often considered "heavy"

Chapter 8 roadmap

8.1 What is network security?
8.2 Principles of cryptography
8.3 Message integrity
8.4 Securing e-mail
8.5 Securing TCP connections: SSL
8.6 Network layer security: IPsec
8.7 Securing wireless LANs
8.8 Operational security: firewalls and IDS

Secure e-mail

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

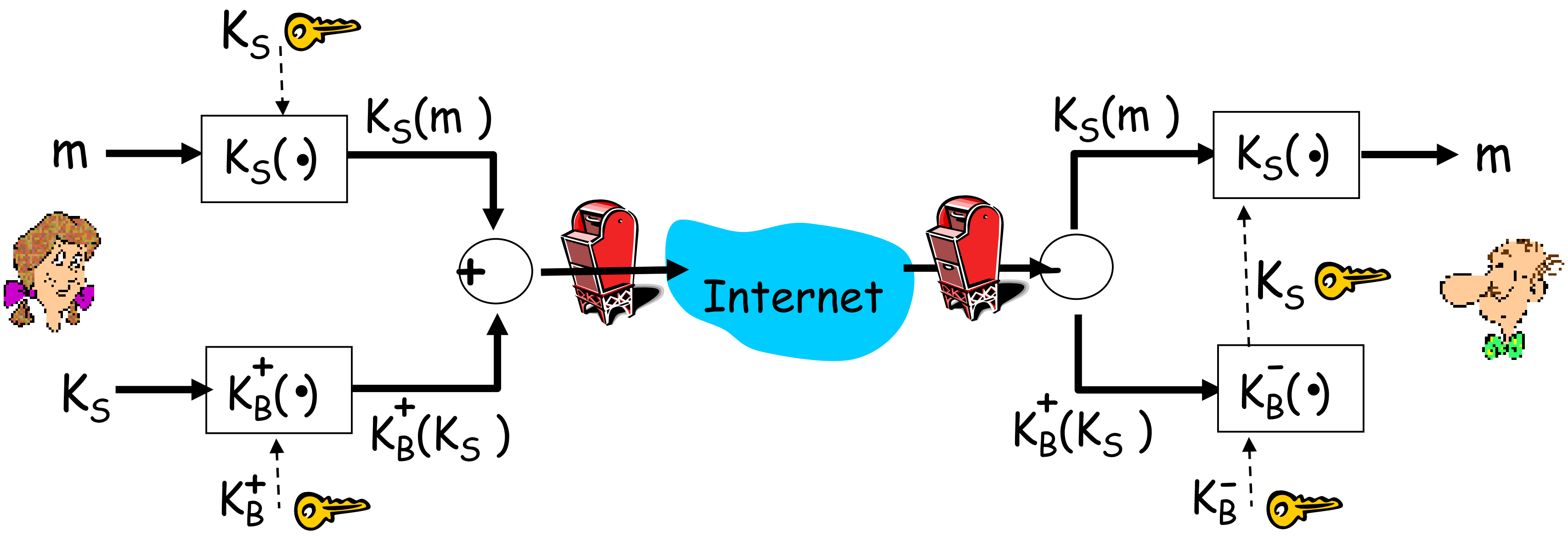


Alice:

- 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.

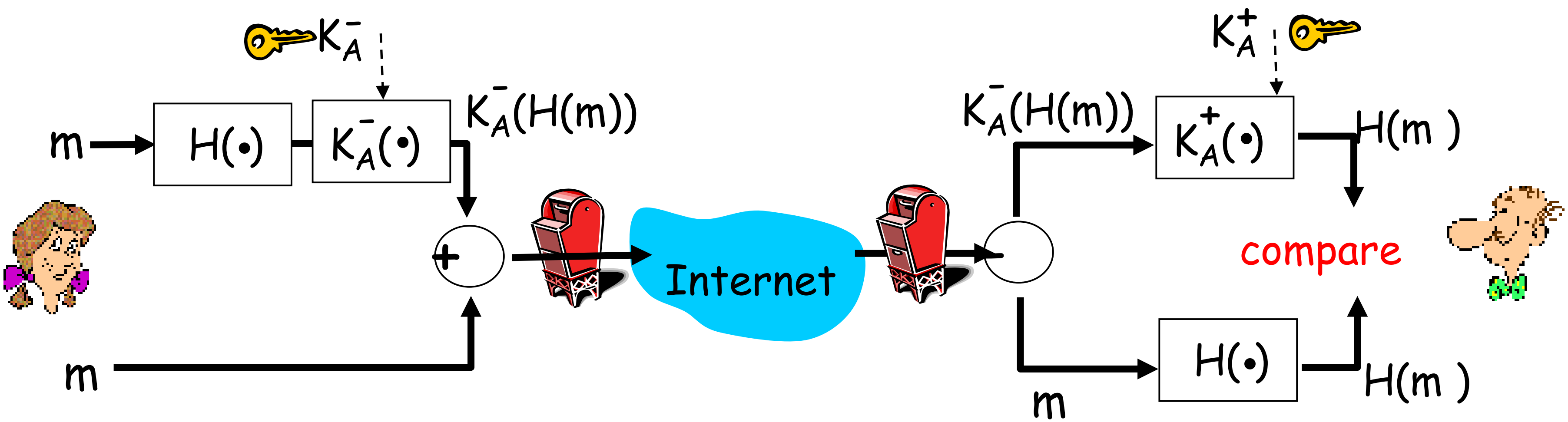


Bob:

- 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)

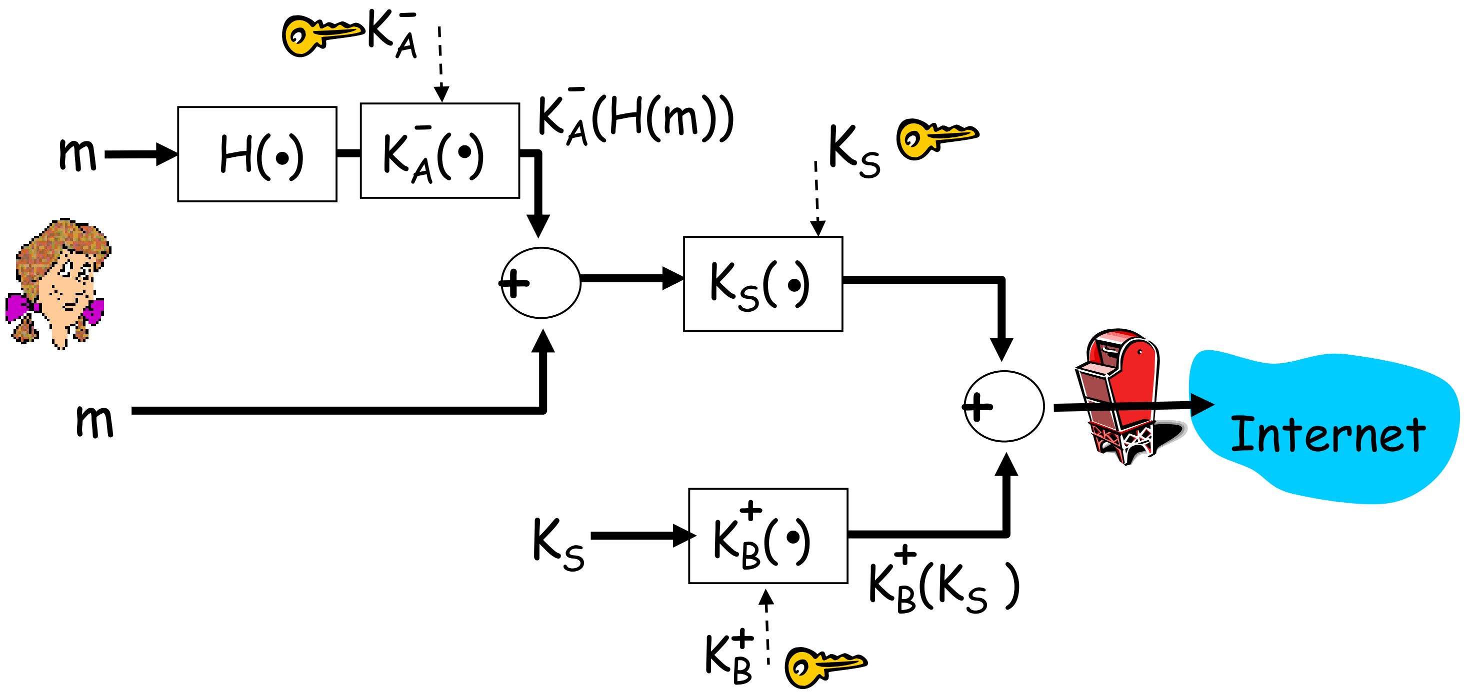
- Alice wants to provide sender authentication message integrity.



- 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

思考题

- 假设你为智慧家庭应用设计通信协议，保护数据的安全
  - 物联网设备（传感器、摄像头）：ID_1-ID_n
  - 网关(路由器)：ID_0
  - 数据云
  - 客户端（浏览器，APP）
- 需要满足数据安全性
  - 保密
  - 身份认证
  - 抵御重放攻击
  - 轻量级

思考题

- 给每个物联网设备配置一个密钥
  - 设备i的密钥为ki
  - 网关设备的密钥为k0
  - 网关设备配置所有终端的密钥ki
- 数据云记录每一个家庭网关的密钥k0
- 用户在数据云中注册账号并关联到对应的网关上

思考题

- 家庭网关对物联网终端设备的鉴别
  - 定义数据格式：$ID_i||E_{ki}(ID_i||Data)$
  - 网关接收到上述消息后，通过ID_i查找到密钥ki，解密得到消息$ID_i||Data$
  - 对比解密得到的ID_i以及数据报文头的ID_i；若一致，则鉴别通过
  - 获得数据Data
- 物联网终端设备对网关的鉴别采用同样的方式
- 注意：上述身份鉴别协议同时完成了数据加密与数据完整性保护，是一种“轻量级”的安全协议

思考题

- 家庭网关上传数据到数据云
  - 家庭网关汇聚数据：DATA=ID_0||Data…ID_n||Data
  - 产生会话密钥k，加密数据E_{k0}(k)以及Ek（ID0,Time,DATA）
  - 发送数据：ID_0|| E_{k0}(k)|| E_k（ID_0,Time, DATA）
- 数据云接收数据
  - 留作大家思考

思考题

- 留作大家思考
  - 数据云下发指令
  - APP与数据云的交互
  - 是否可以使用公钥体系?
  - 什么情况下需要使用公钥证书?