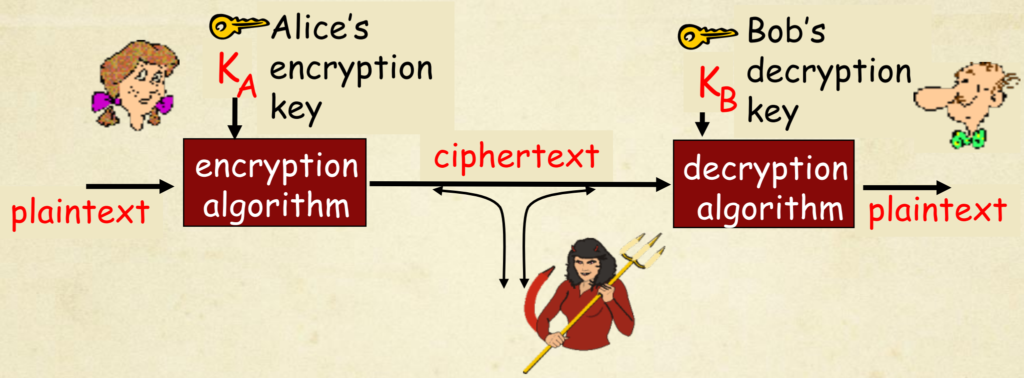
Security

CSC 249
April 10, 2018

Network Security

- Symmetric Key Cryptography
  - Caesar cipher
  - DES and AES
- Public Key Cryptography

Cryptographic Keys



Symmetric key cryptography: sender & receiver keys are *identical* and *secret* (but known by 2 parties)

Public-key cryptography: the encryption key is *public*, the decryption key *secret*, and know only by one party

Symmetric Key Cryptography

- Both parties have the same key
- Use this key to both encrypt and decrypt the message

  → The actions are symmetric
- Early – Caesar Cypher
- Now, two dominant algorithms
  - DES – data encryption standard
  - AES – advanced encryption standard

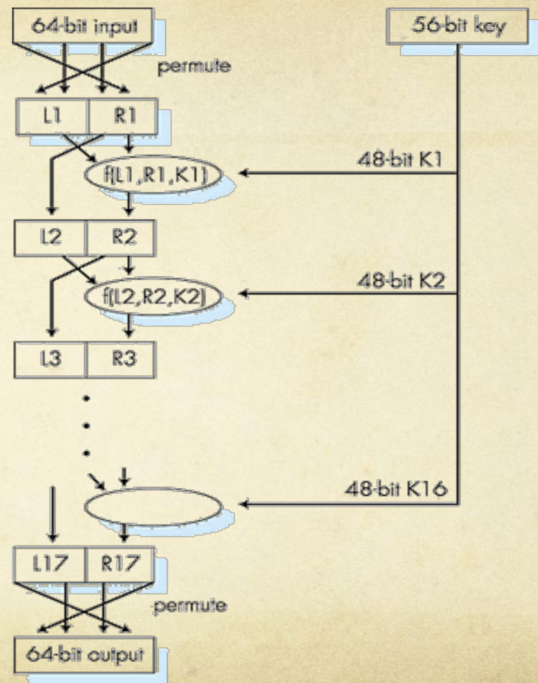
Symmetric key cryptography: DES

DES operation

Initial Permutation

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

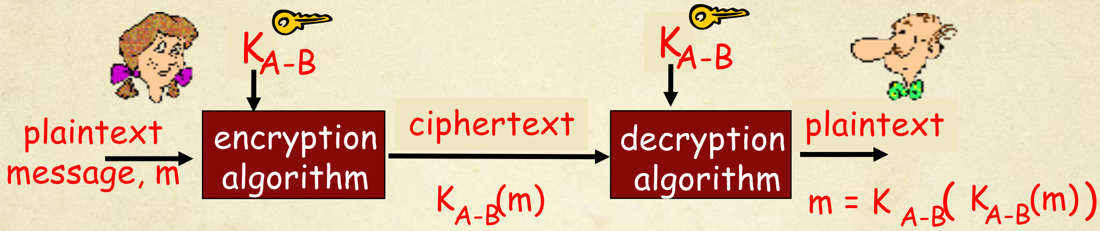
Final permutation



AES: 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

Symmetric Key Cryptography



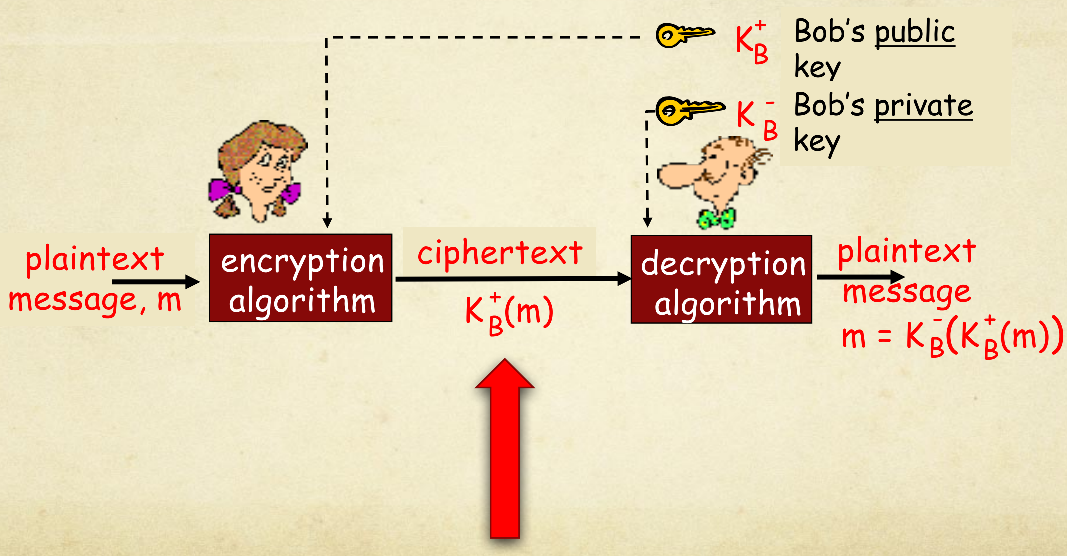
Symmetric key cryptography: Bob and Alice share/know the same (symmetric) key: K

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

- Q: how do Bob and Alice agree on key value?

7

Public Key Cryptography



8

RSA Important Property

The following property defines this method:

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

use public key first, followed by private key

use private key first, followed by public key

9

Public key encryption algorithm

Requirements:

1. need $K_B^-(\bullet)$ and $K_B^+(\bullet)$ 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

10

RSA: Choosing keys (an art)

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

11

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)

Number theory result	$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

12

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		

decrypt:

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

13

* Activity *

- Using RSA, choose p = 3, q = 11. Encode a letter of your choice and send it to a different host to decode.
- Suggestion for e? ... choose e =
- Then z = (p-1)(q-1) =
- Also choose d =
- Thus n =

14

* Activity *

- So we have
  - n = 33, e = 9, d = 9
  - (n,d) & (n, e)
- Encrypt a LETTER and pass it across the room to be decrypted

15

RSA in practice: session keys

- Exponentiation in RSA is computationally intensive
- DES/AES is at least 100 times faster than RSA
- Use public key crypto 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

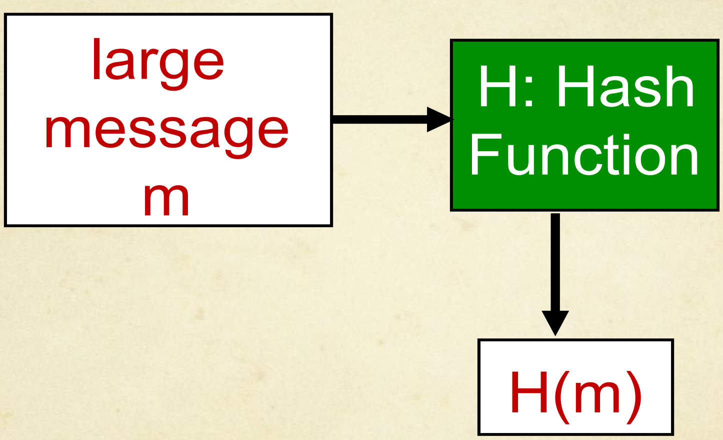
8-16

Next Security Tasks

- Encryption keys are public, so anyone could *claim* to be someone else
  - Need more than public key cryptography
- Ensure message is not corrupted
  - Message integrity with Message Authentication Code (MAC)
- Bind message to sender – end-point authentication
  - Digital signature
- **Use: Cryptographic hash function**

17

Hash Functions

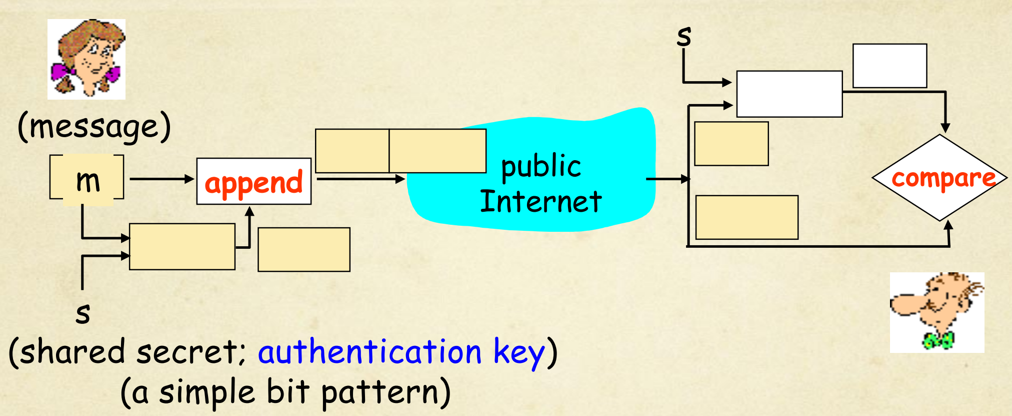


Cryptographic Hash Function

- The ideal cryptographic hash function has four properties:
  1. Easy to compute the hash value for any message, H(m)
  2. Infeasible to generate the message from the hash
  3. Infeasible to modify a message without changing the hash H(m') ≠ H(m)
  4. Infeasible to find two different messages with the same hash H(m1) ≠ H(m2)
- The output is called the *digest*
- Note – there is no encryption here

(1) Message Authentication Code:

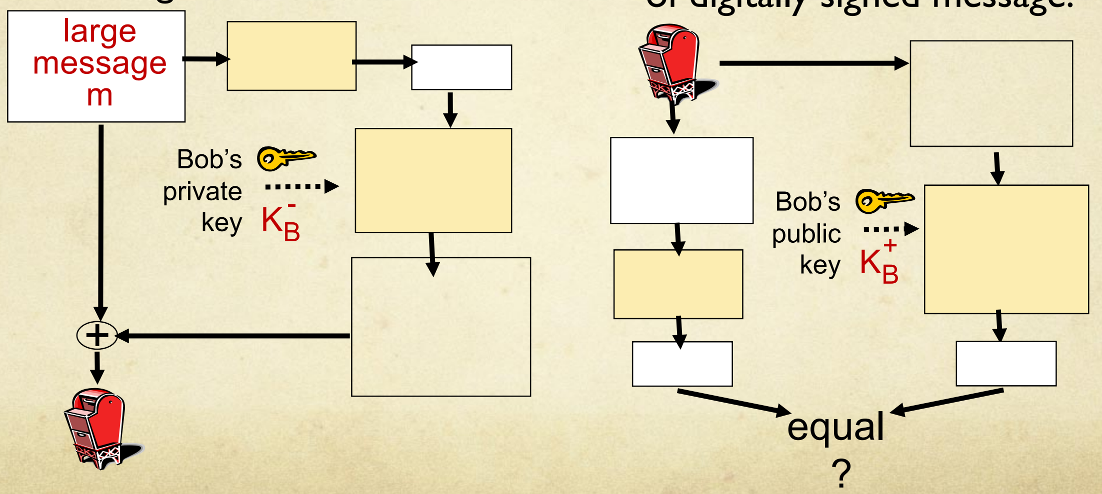
→ Use Shared Secret: H(m+s) = MAC



Digital signature = signed message digest

Bob sends digitally signed message:

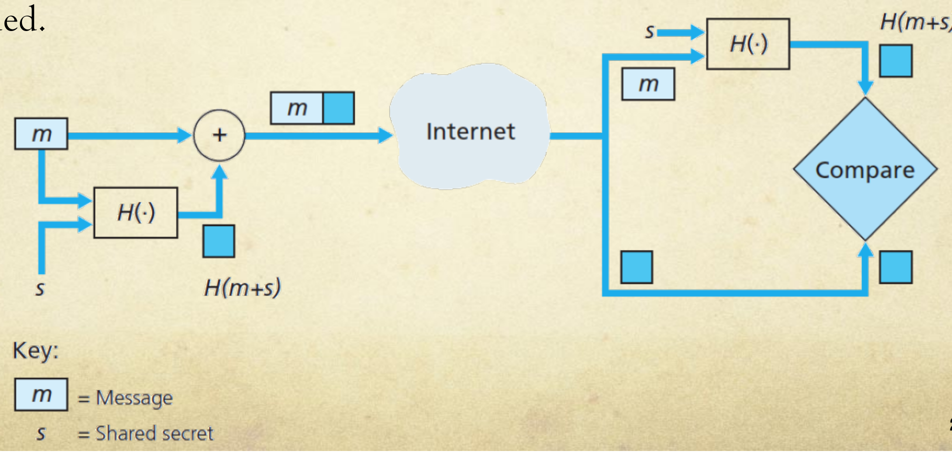
Alice verifies signature, integrity of digitally signed message:



8-21

Task: Integrity + Authentication

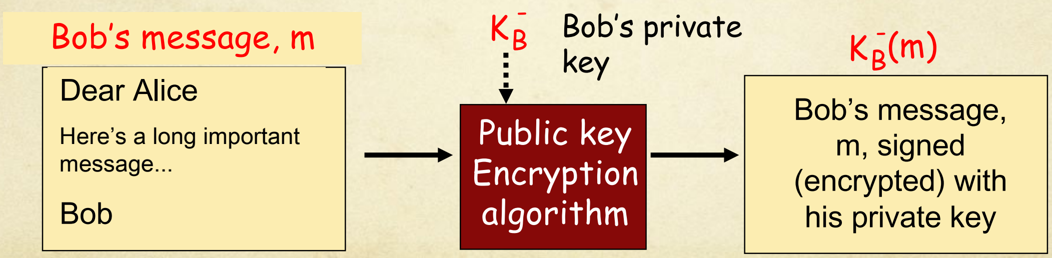
- Suppose Alice and Bob share <u>two</u> secret keys:
  - an authentication key S1 <u>and</u>
  - a symmetric encryption key S2.
- Augment the figure so that both integrity and confidentiality are provided.



22

(2) Digital Signature: Use Public Key Cryptography

- Bob signs m by encrypting it with his private key K_B^-, creating "signed" message, $K_B^-(m)$
- Binds the message to the sender (stronger than H(m+s))



23

Digital Signatures (more)

- Alice verifies m signed by Bob by
  -
  -
- 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.

24