3. Routers

- A networking device that connects multiple networks together and forwards data packets from one network to another
  - Routers enable you to create a secure network by assigning it a passphrase
  - An additional layer of security: the router provides you the capability to specify the unique media access control (MAC) address of each legitimate device connected to the network and restrict access to any other device that attempts to connect to the network

4. Encryption

- The process of scrambling messages or data in such a way that only authorized parties can read it
- Used to protect billions of online transactions each day, enabling consumers to order more than $300 billion in merchandise online and banks to route $40 trillion in financial transactions each year

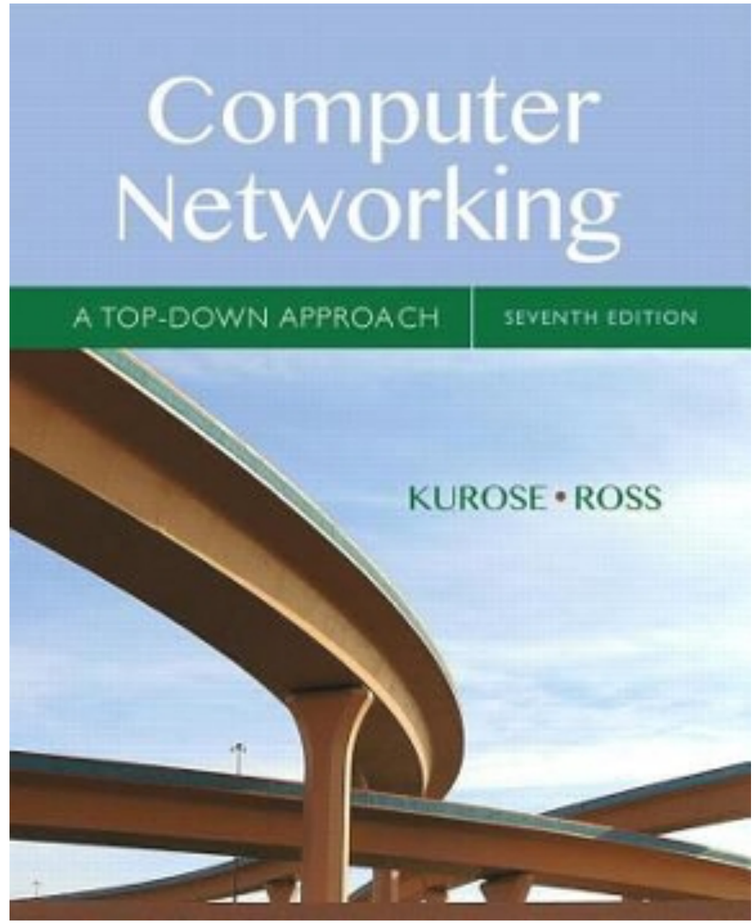
Introduction to Cryptography

Please Watch Video for more details!

Reference

J. Kurose and K. Ross, **Computer Networking: A Top-Down Approach**, Seventh Edition.

Chapter 8: Network Security



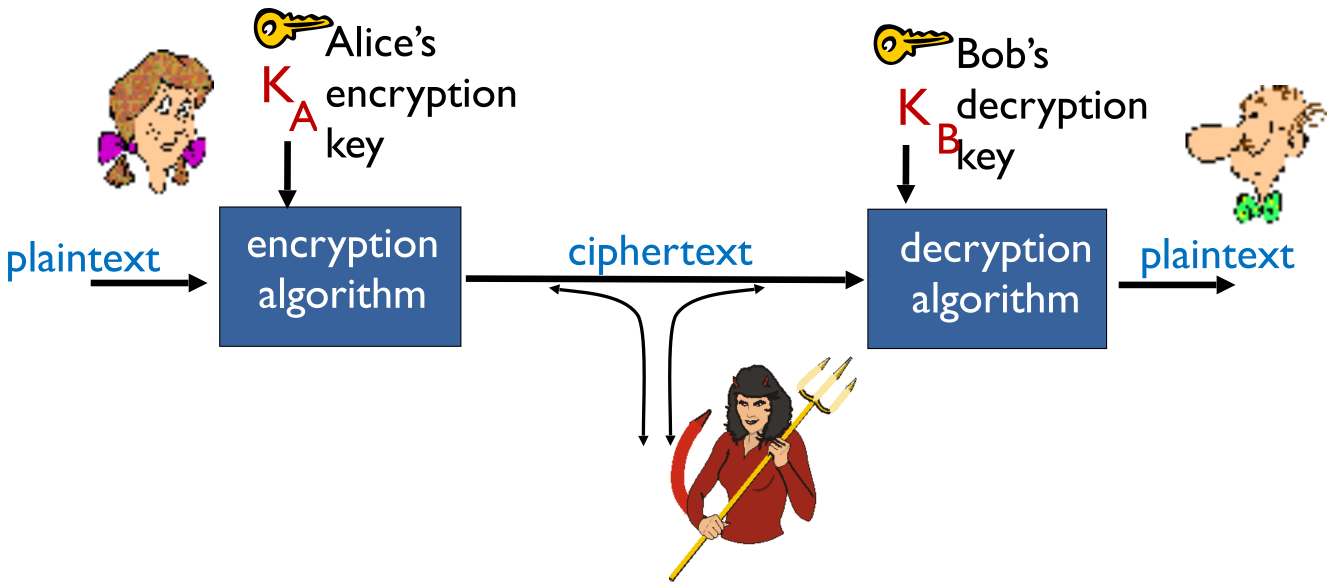
Reference

L. Buttyan and J.-P. Hubaux, **Security and Cooperation in Wireless Networks**, Cambridge University Press.

Appendix A: Introduction to Cryptographic Algorithm and Protocols

Contents

The Language of Cryptography



m plaintext message

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

$m = K_B(K_A(m))$

Basic Classification Encryption Schemes

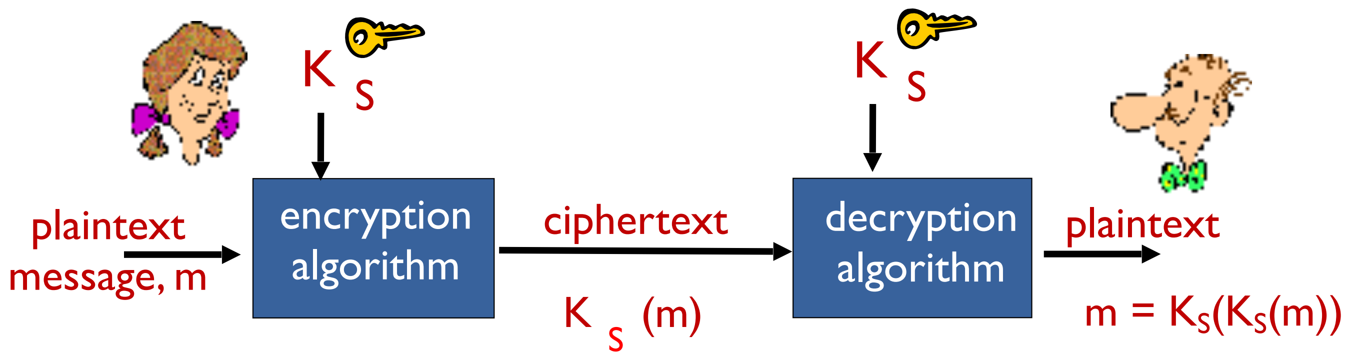
- Symmetric-key encryption
  - It is easy to compute K’ from K (and vice versa)
  - Usually K’ = K
  - Two main types:
    - **Stream ciphers** – operate on individual characters of the plaintext
    - **Block ciphers** – process the plaintext in larger blocks of characters
- Asymmetric-key encryption
  - it is hard (computationally infeasible) to compute K’ from K
  - K can be made public (i.e., public-key cryptography)

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?

Contents

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?

Simple Encryption Scheme

Substitution Cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

e.g.:

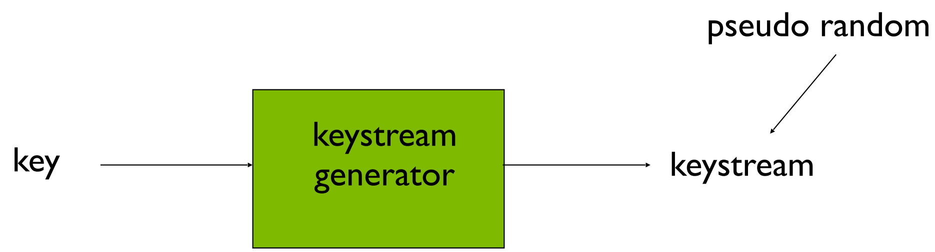
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc

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

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

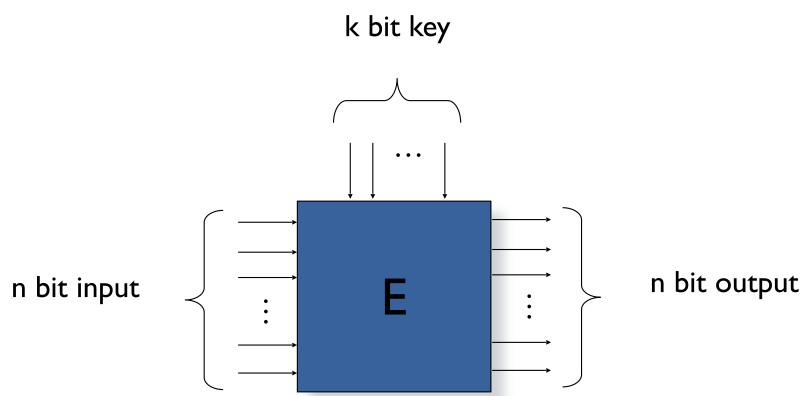
Stream Ciphers



- Combine each bit of keystream with bit of plaintext to get bit of ciphertext
- $m(i)$ = i^{th} bit of message
- $k_s(i)$ = i^{th} bit of keystream
- $c(i)$ = i^{th} bit of ciphertext
- $c(i) = k_s(i) \oplus m(i)$ ($\oplus$ = exclusive or)
- $m(i) = k_s(i) \oplus c(i)$

Block Ciphers

An *n* bit block cipher is a function E: $\{0, 1\}^n \times \{0, 1\}^k \Longrightarrow \{0, 1\}^n$, such that for each $K \in \{0,1\}^k$, $E(x, K) = E_K(x)$ is an invertible mapping from $\{0,1\}^n$ to $\{0,1\}^n$



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 (Number of Possible Key)

- 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

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

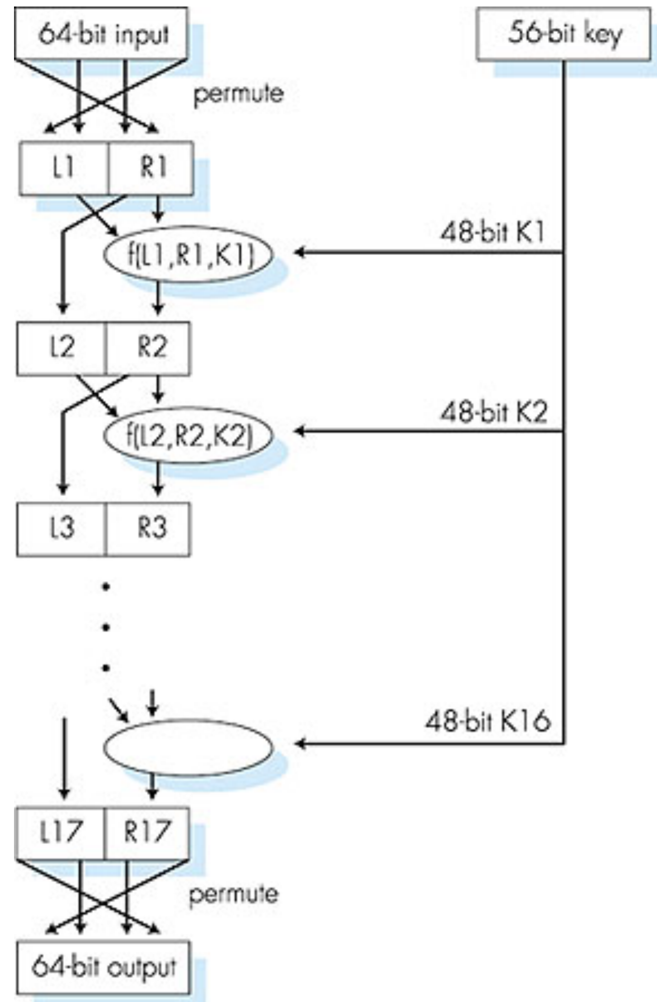
Symmetric Key Crypto: DES

DES operation

initial permutation

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

final permutation



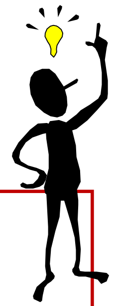
AES: Advanced Encryption Standard

- 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

Contents

- The Language of Cryptography
- Symmetric Key Cryptography
- Asymmetric Key Cryptography
- Message Integrity (MAC)
- Digital Signature

Public Key Cryptography



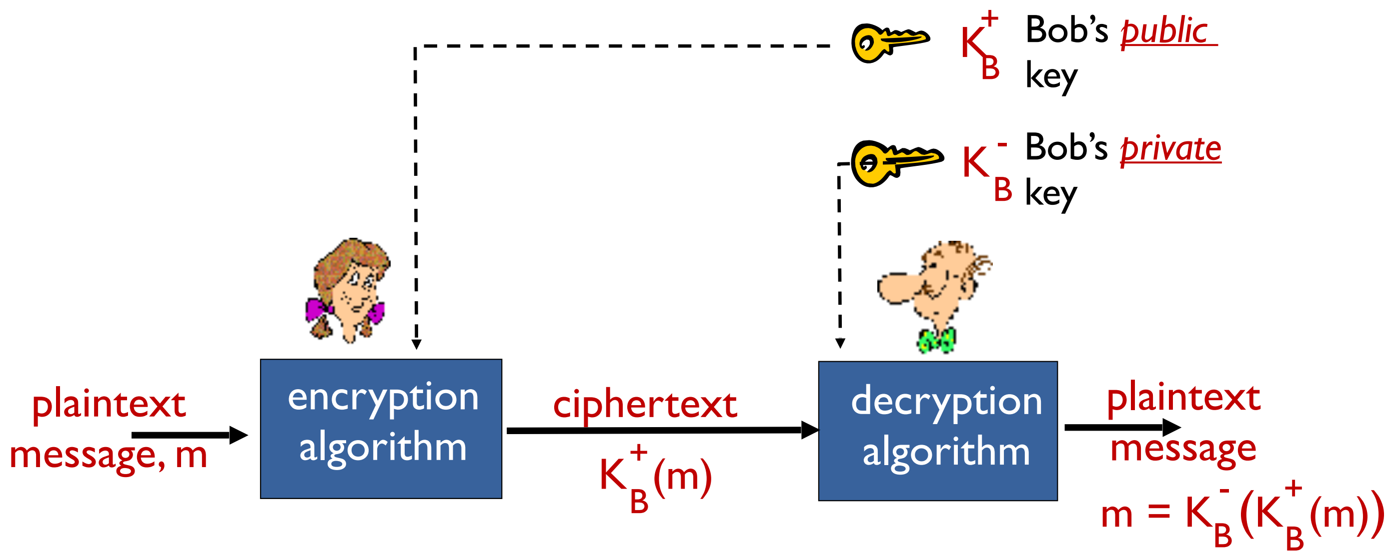
symmetric key crypto

- 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 encryption algorithms

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

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:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \rightarrow x^d \bmod 10 = 6$

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 $\underbrace{(n,e)}_{K_B^+}$. *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.

encrypt: $\underbrace{\text{bit pattern}}_{00001100}$ $\quad \underbrace{m}_{12}$ $\quad \underbrace{m^e}_{24832}$ $\quad \underbrace{c = m^e \bmod n}_{17}$

decrypt: $\underbrace{c}_{17}$ $\quad \underbrace{c^d}_{481968572106750915091411825223071697}$ $\quad \underbrace{m = c^d \bmod n}_{12}$

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,

$$
\begin{aligned}
c^d \bmod n &= (m^e \bmod n)^d \bmod n \\
&= m^{ed} \bmod n \\
&= m^{(ed \bmod z)} \bmod n \\
&= m^1 \bmod n \\
&= m
\end{aligned}
$$

RSA: Another Important Property

The following property will be *very* useful later:

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

result is the same!

Why Public Key and Private key can be exchanged over a message?

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

RSA in Practice: Session Keys

- Exponentiation in RSA is computationally intensive
- DES 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

Contents

Message Integrity

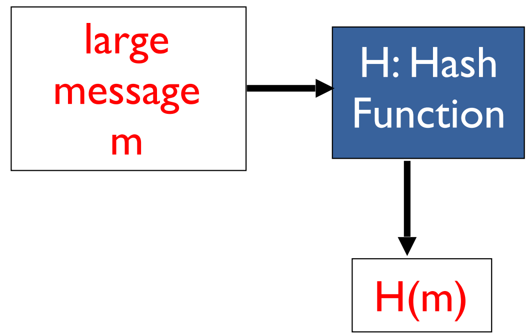
- Allows communicating parties to verify that received messages are authentic.
  - Content of message has not been altered
  - Source of message is who/what you think it is
  - Message has not been replayed
  - Sequence of messages is maintained
- Let's first talk about message digests

Message 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"

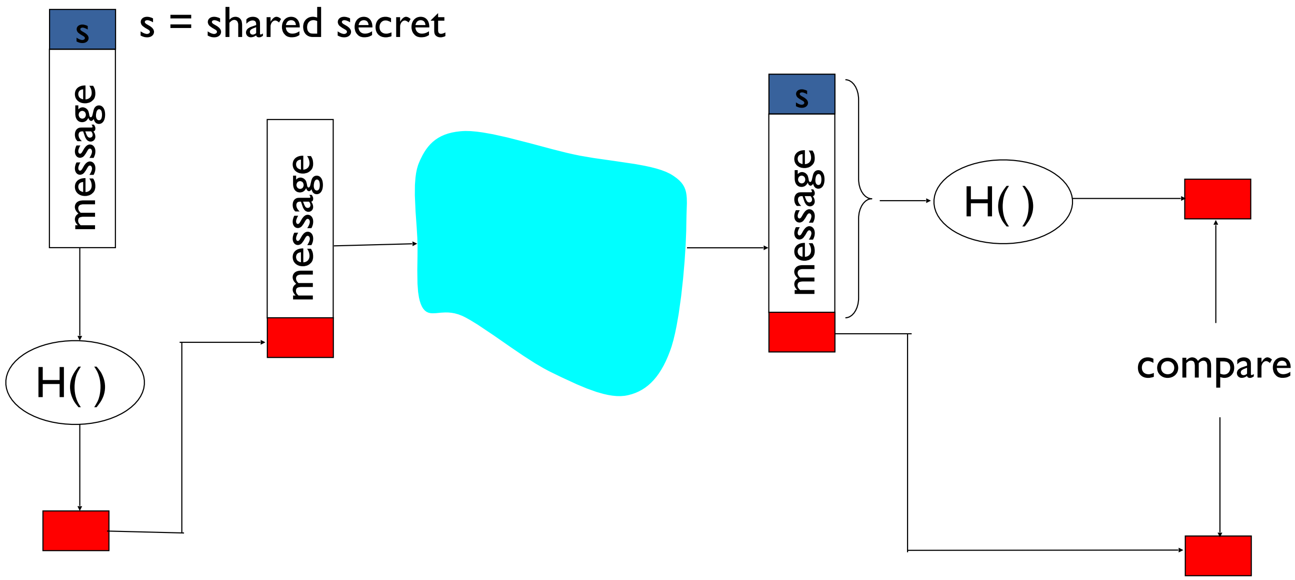


- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
  - Seemingly random output

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 $m||MD_m$

Contents

Digital Signatures

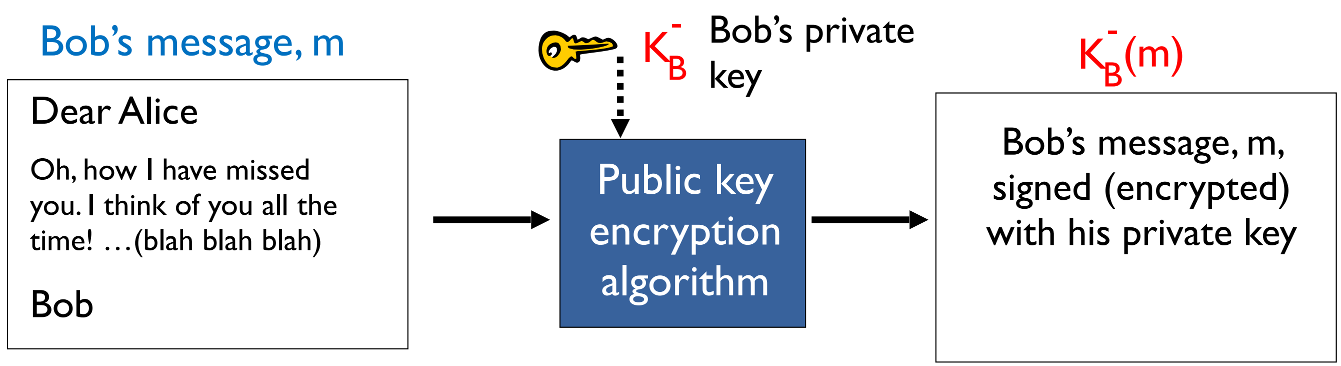
Cryptographic technique analogous to hand-written signatures.

- 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 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:

1. Bob signed m.
2. No one else signed m.
3. Bob signed m and not m'.

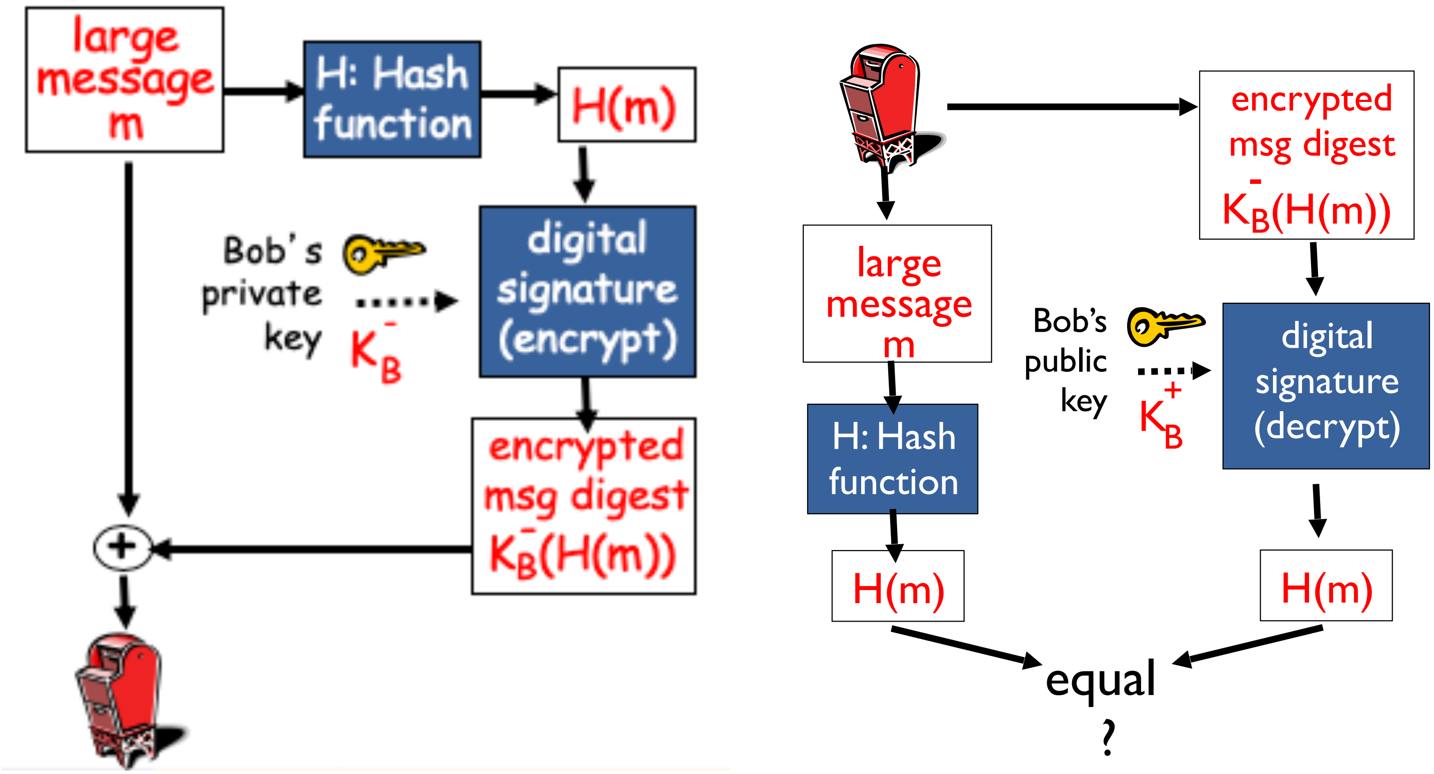
Non-repudiation:

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

Digital Signature = Signed Message Digest

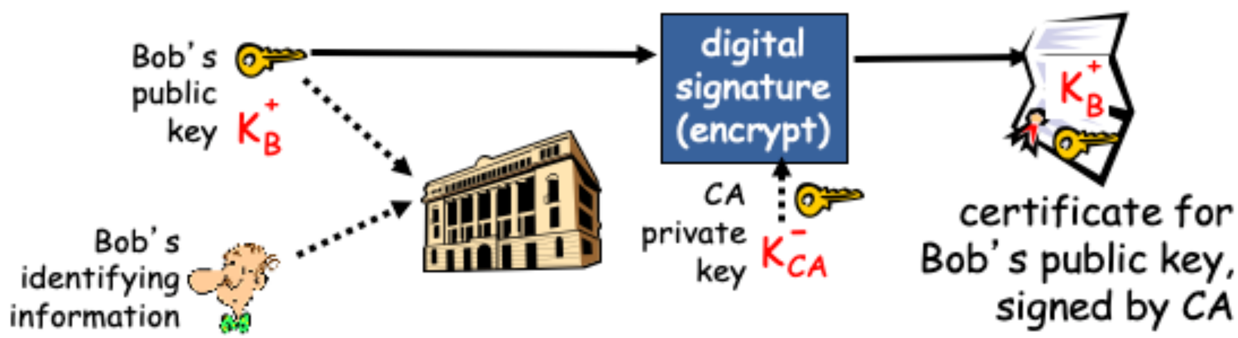
Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:



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