College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

Encryption

Session 9

INST 346

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

Stream and Block Ciphers

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- 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; ..
  - random initialization
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern

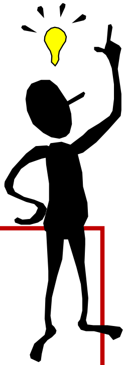
AES: Advanced Encryption Standard

- 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

Public Key Cryptography

symmetric key crypto

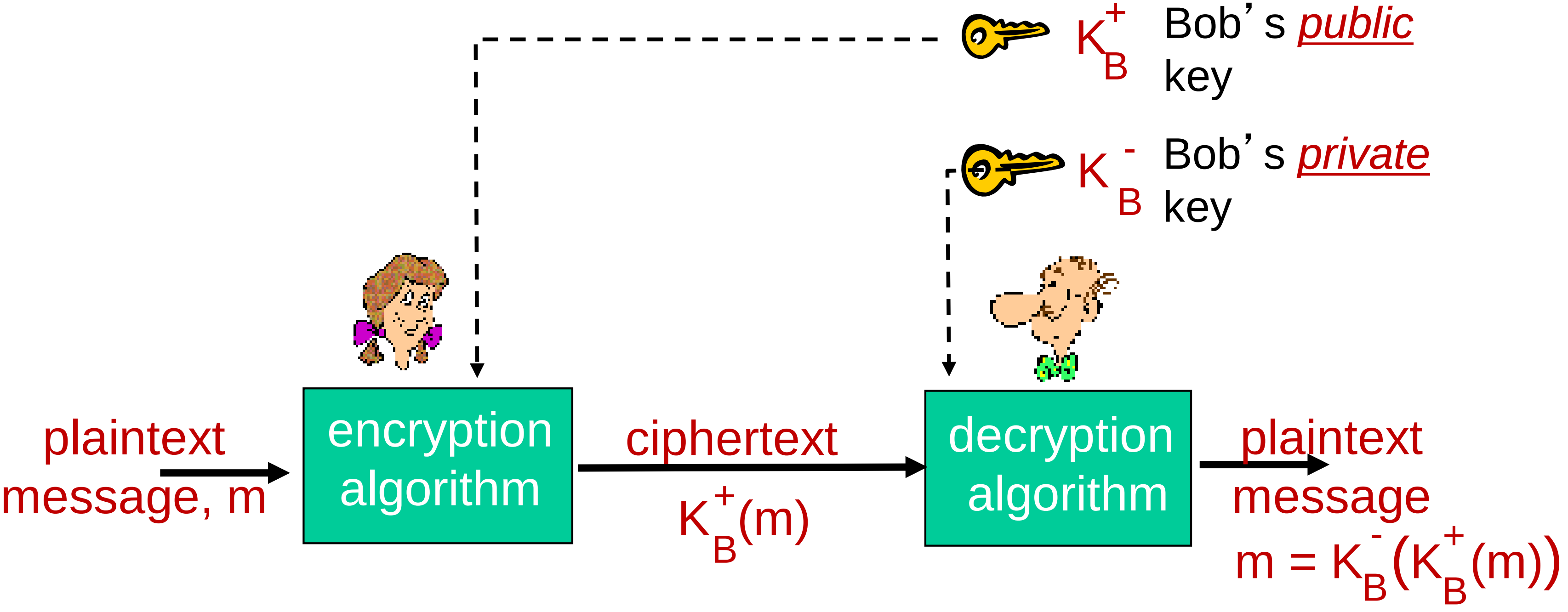
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?



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



K_B^+ Bob’s *public* key

K_B^- Bob’s *private* key

plaintext message, m

encryption algorithm

ciphertext

$K_B^+(m)$

decryption algorithm

plaintext message

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

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

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

   $m = c^d \bmod n$

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

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: $\underbrace{\text{bit pattern}}_{\text{0000l000}}$ $\underbrace{m}_{12}$ $\underbrace{m^e}_{24832}$ $\underbrace{c = m^e \bmod n}_{17}$

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

RSA: an important property

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

result is the same!

Why 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

Digital signatures

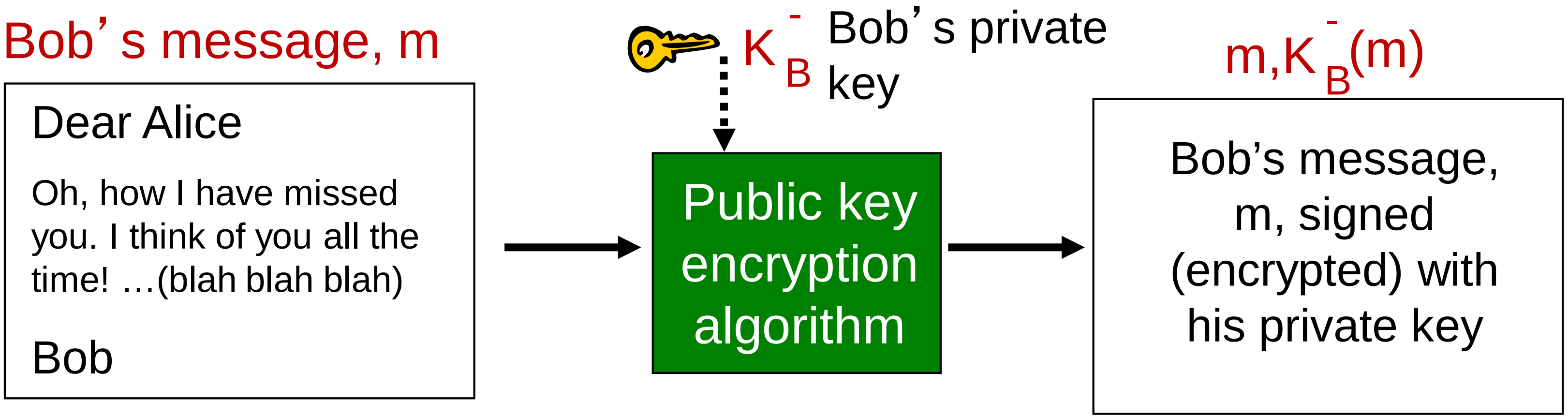
cryptographic technique analogous to hand-written signatures:

- 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, $K_B^-(m)$



Bob’s message, m

Dear Alice

Oh, how I have missed you. I think of you all the time! …(blah blah blah)

Bob

K_B^- Bob’s private key

Public key encryption algorithm

$m,K_B^-(m)$

Bob’s message, m, signed (encrypted) with his private key

In practice, this is done more efficiently on message digests

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 '

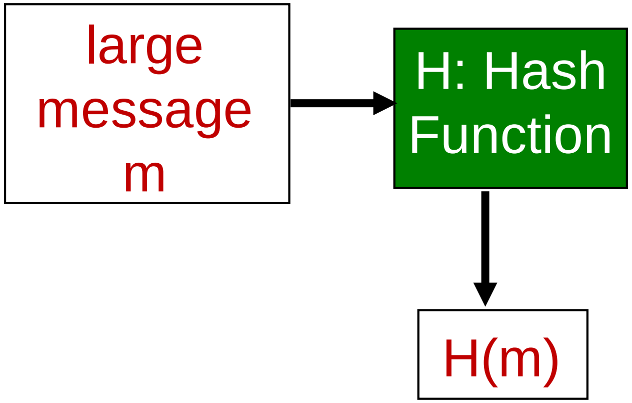
non-repudiation:

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

Message digests

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)

TCP 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 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!

Widely used hash functions

- MD5 (RFC 1321) has known vulnerabilities
  - computes 128-bit message digest in 4-step process
- SHA-1 is widely used but is deprecated
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest
  - Collision attack with 1000 GPUs in a month
- SHA-2 and SHA-3 are now available
  - Also standardized by NIST
  - More secure, but slower (in software)

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today’s class?