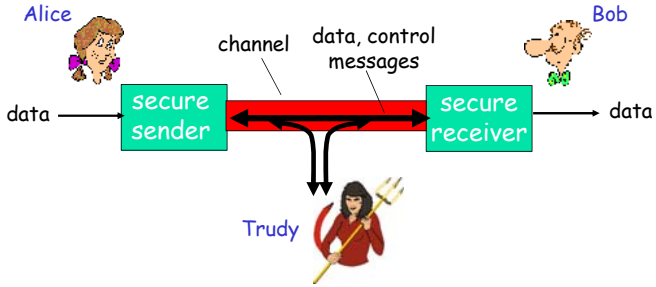
Principles of Network Security

Kai Shen

The Network Security Model

- Bob and Alice want to communicate "securely".
- Trudy (the adversary) has access to the channel.



Who might Bob and Alice be?

- Web browser/server for electronic transactions (e.g., on-line purchases/banking)
- DNS servers
- routers exchanging routing table updates
- … well, *real-life* Bobs and Alices!

What can an adversary do?

- eavesdrop: understand the content of messages
- actively changing messages
- impersonation: fake (spoof) identity
- denial of service: prevent service from being used by others (e.g., by overloading resources)

What is Network Security?

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

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

Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards)

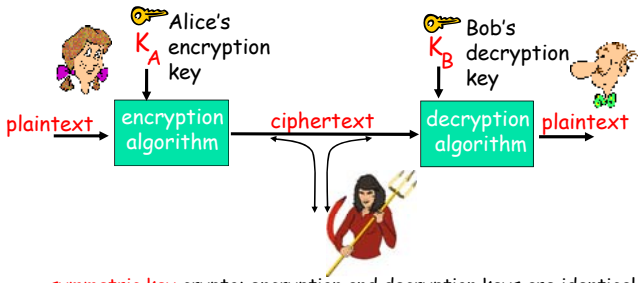
Access and Availability: services must be accessible and available to (and only to) legitimate users.

Principles of Network Security

- Confidentiality: cryptography
- Authentication
- Integrity

The Language of Cryptography

First goal of cryptography: confidentiality.



- symmetric key crypto: encryption and decryption keys are identical. (both are secret)
- public key crypto: encryption key is public, decryption key is secret.

Symmetric Key Cryptography: Monoalphabetic Cipher

Monoalphabetic cipher: substitute one letter for another.

```
plaintext:  abcdefghijklmnopqrstuvwxyz
            |                       |
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

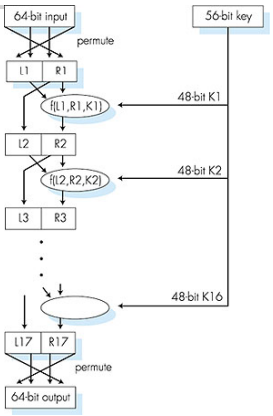
Example: `Plaintext: bob. i love you. alice`
`ciphertext: nkn. s gktc wky. mgsbc`

Q1: How hard to break this simple cipher?
- brute force?
- other?

Q2: How to make it more difficult to break?

Symmetric Key Cryptography: DES

- DES: Data Encryption Standard
  - US encryption standard [NIST 1993]
  - 56-bit symmetric key, 64-bit plaintext input
  - encryption: initial permutation ⇒ 16 "rounds", each using different 48 bits of key ⇒ final permutation
  - decryption: reverse operation using the same key
- How secure is DES?
  - DES Challenge (1999): 56-bit-key-encrypted phrase decrypted (brute force) in 22 hours 15 minutes
- Making DES more secure:
  - use three keys sequentially (3-DES)
  - use more bits



AES: Advanced Encryption Standard

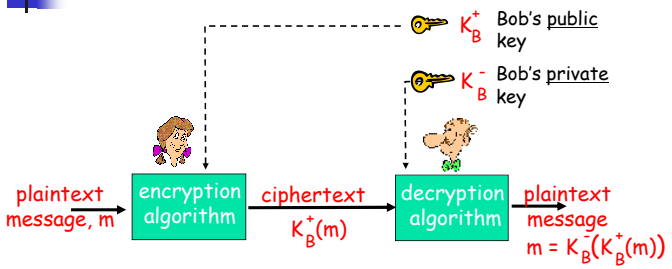
- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for 128-bit AES

Public Key Cryptography

symmetric key cryptography
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place? (particularly difficult if Trudy is eavesdropping on all communication)

public key cryptography
- encryption key is different from decryption key
- encryption key is public, known to everyone, also called public key
- decryption key is secret, known only to receiver, also called private key

Public Key Cryptography



Principle for choosing the public/private key pair:
One should not be able to derive the private key from the public key.

Public Key Cryptography: RSA
(Ron Rivest, Adi Shamir and Len Adleman)

- Choosing keys:
  - Choose two large prime numbers p, q. (e.g., 1024 bits each)
  - Compute n = pq, z = (p-1)(q-1)
  - Choose e (with e<n) that has no common factors with z.
  - Choose d such that ed-1 is exactly divisible by z.
  - Public key is (n,e). Private key is (n,d).
- To encrypt a message, m (<n): do $c = m^e \bmod n$
- To decrypt a received ciphertext, c: do $m = c^d \bmod n$
- Reason: for any m (relatively prime with n)
  - $m^z \bmod n = 1$; therefore $m^{ed-1} \bmod n = 1$
- Another property: $(m^d \bmod n)^e \bmod n = m$
- RSA is much slower than the symmetric key cryptos.

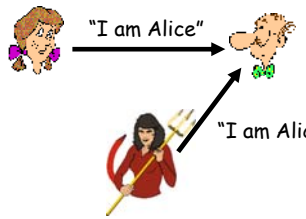
Principles of Network Security

- Confidentiality: cryptography
- Authentication
- Integrity

Authentication: version 1.0

Authentication: Bob wants Alice to "prove" her identity to him.

Protocol ap1.0: Alice says "I am Alice".

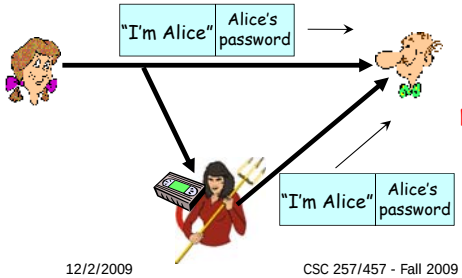


Failure scenario??

Trudy can simply declare herself to be Alice

Authentication: version 2.0

Protocol ap2.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

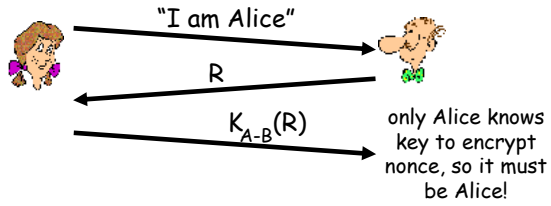
playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: version 3.0

Goal: avoid playback attack

Nonce: number (R) used only *once-in-a-lifetime*

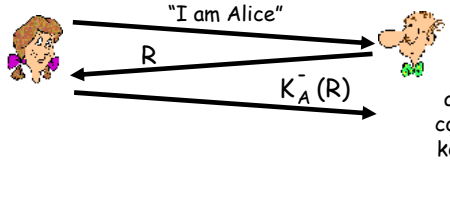
ap3.0: Bob sends Alice a nonce, R. Alice must return R, encrypted with shared secret key



Authentication: version 4.0

ap3.0 requires shared symmetric key. Key distribution can be a problem.

ap4.0: use nonce, public key cryptography.



Principles of Network Security

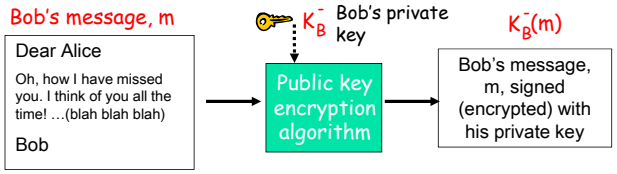
- Confidentiality: cryptography
- Authentication
- Integrity

Integrity

- Digital Signatures:
  - Cryptographic technique to ensure document integrity.
  - analogous to hand-written signatures.
- sender (Bob) digitally signs document, establishing he is document owner/creator.
- the recipient (Alice) receives the document and the digital signatures.
- the recipient can be sure that the document is
  - verifiable: Bob signed the document.
  - nonforgeable: the document hasn't been changed since Bob signed it.

Digital Signatures

- Bob signs m by encrypting with his private key, creating a digital signature $K_B^-(m)$

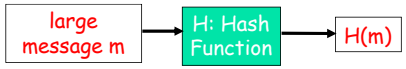


- Suppose Alice receives msg m and its digital signature $K_B^-(m)$
- Alice applies Bob's public key K_B^+ to $K_B^-(m)$ then checks whether $K_B^+(K_B^-(m)) = m$.
- If so, whoever signed m must have used Bob's private key.

Problem: computationally expensive to public-key-encrypt long messages.

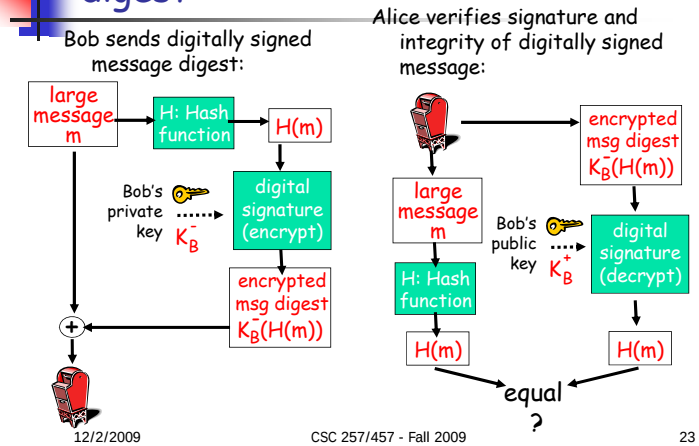
Message Digests

- apply a hash function H to m, get a much smaller message digest H(m).



- public-key-encrypt the message digest to generate the digital signature $K_B^-(H(m))$.
- example hash functions?

Digital signature = signed message digest



Message Digests: good/bad hash function

- apply a hash function H to m, get a much smaller message digest H(m).
- public-key-encrypt the message digest to generate the digital signature $K_B^-(H(m))$.

Good/bad has functions?

- Note: given a hash function, it is possible for many messages sharing the same digest.

Internet Checksum: Poor Hash Function for Generating Message Digests

Given a message and its Internet checksum, it is easy to find another message with same checksum.

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 property: given digest x for message m, computationally infeasible to find another message m' that shares the same digest.

Good Hash Functions for Generating Message Digests

- MD5 hash function widely used
  - computes 128-bit message digest in 4-step process.
  - appears difficult to construct message m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Disclaimer

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).