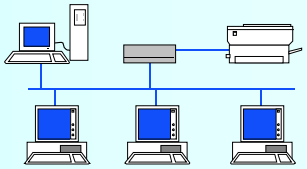


Distributed Systems
600.437
Security

Department of Computer Science
The Johns Hopkins University

Security

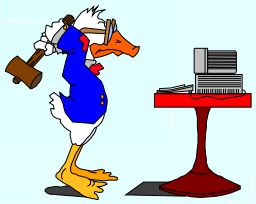
Lecture 12

Applied Cryptography by Bruce Schneier (www.counterpane.com)

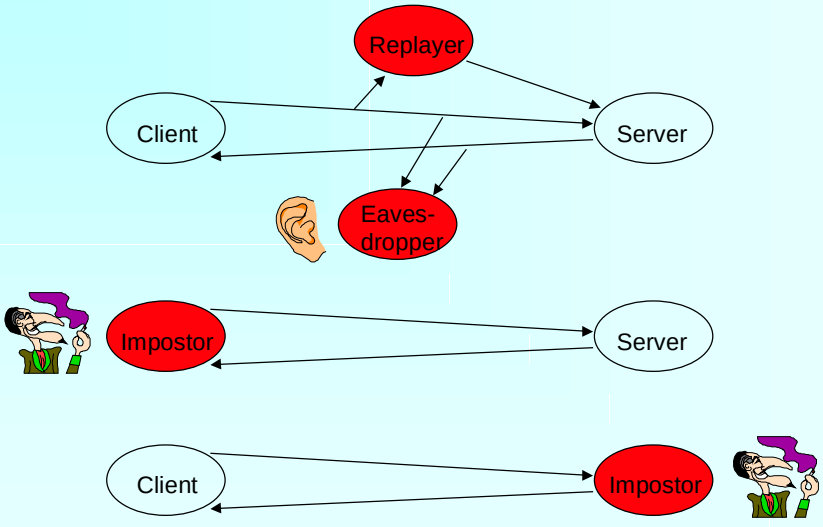
www.securityfocus.com, www.ssh.fi/tech/crypto/

Threats in Computer Systems

- **Leakage** - acquisition of information by unauthorized recipients.
- **Tampering** - the unauthorized alteration of data and programs.
- **Resource stealing.**
- **Vandalism**: interference with the proper operation of a system.

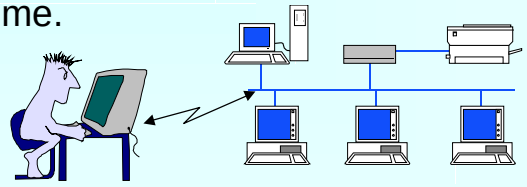


Distributed Systems Problems



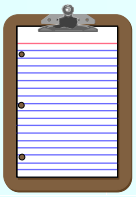
Threats in Distributed Systems

- **Eavesdropping** - obtaining copies of messages without authority.
- **Masquerading** - sending/receiving messages using the identity of others.
- **Message tampering** - altering message content on its way in a store & forward net.
- **Replaying** - Storing messages and sending them at a later time.

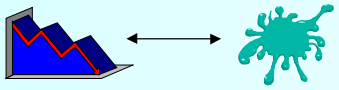


Security Requirements

- Secure communication channels to avoid eavesdropping.
- Authentication protocols for mutually suspicious clients and servers.
  - Servers must be satisfied that clients are who they claim to be.
  - Clients must be satisfied that they get the service from authentic servers.
- Ensure that communication is fresh.



Security Mechanisms



- Cryptography.
  - Secret-key encryption (DES) - secure and computationally cheap.
  - Public-key encryption (RSA) - secure and convenient, no need for secure channel for distributing the keys.
- Authentication services.
  - Kerberos.
- Access control mechanisms.
  - Access control lists.

A strict policy enforcement is important !!

Secret-key Encryption

1. Acquire Key
2. $f(key, mess) \rightarrow Cmess$
3. send Cmess

1. Acquire Key

- The functions define a family of encryption methods. The key selects a specific method.

Secret-key Encryption

1. Acquire Key
2. $f(key, mess) \rightarrow Cmess$
3. send Cmess

1. Acquire Key

2. receive Cmess
3. $f^{-1}(key, Cmess) \rightarrow mess$

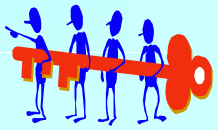
- The functions define a family of encryption methods. The key selects a specific method.
- There should be a secure method for distributing the keys.
- The key must not be compromised.

Data Encryption Standard (DES)

- Maps a 64 bits of plain input to a 64 bits of encrypted output using 56 bits key.
- The basic function is identical to the inverse function.
- No known successful attack strategy (even when matching pieces of plain and encrypted text are available).
- Developed by IBM, adopted as a standard by US government (1977).

Public-key Encryption

1. Compute KeyE, KeyD
2. Publish KeyE

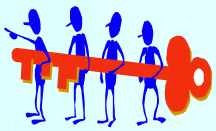


- KeyE is freely distributed.
- KeyD is known only to the receiver.

Public-key Encryption

1. Compute KeyE, KeyD
2. Publish KeyE

1. Acquire KeyE
2. $Enc(keyE, mess) \rightarrow Cmess$
3. send Cmess



- KeyE is freely distributed.
- KeyD is known only to the receiver.

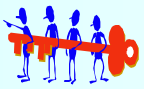
Public-key Encryption

1. Compute KeyE, KeyD
2. Publish KeyE

1. Acquire KeyE
2. $Enc(keyE, mess) \rightarrow Cmess$
3. send Cmess

2. receive Cmess
3. $Dec(keyD, Cmess) \rightarrow mess$

- KeyE is freely distributed.
- KeyD is known only to the receiver.



- Several parties can encrypt using the same key, without being able to read each other messages.

RSA

- Based on the difficulty of finding factors of large numbers.
- No known successful attack strategies.
- Invented by Rivest, Shamir & Adelman 1978.
- Needs two large prime numbers (10^{100} and more).
- *mess = Enc(Dec(mess)) = Dec(Enc(mess))*

RSA Example

P = 5, Q=11 the two prime numbers.
N = P*Q = 55
k < log(N)= 5 the length of mess.

Z = (P-1)*(Q-1) = 40
pick e which has no common factor with Z: e=13
pick d such that e*d = 1 mod Z : d=37

$$Cmess = mess^{e} \bmod(N) \qquad Cmess = mess^{13} \bmod(55)$$

$$mess = Cmess^{d} \bmod(N) \qquad mess = Cmess^{37} \bmod(55)$$

Diffie-Hellman

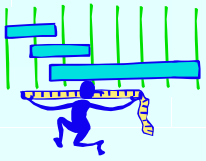
- Based on the difficulty of computing discrete logarithms of large numbers.
- No known successful attack strategies.
- Invented by Whitfield Diffie and Hellman 1976.
- Needs 2 large numbers, one prime(P), one (Q) a primitive root of P.

Diffie-Hellman Example

- Party 1 and 2 exchange P and Q publicly.
- Party 1 computes $A^* = Q^A \ mod(P)$ for secret *A* (known only to itself) and Party 2 computes $B^* = Q^B \ mod(P)$ for secret B. Then they exchange A* and B*.
- The Secret Key is computed by each:
- $DHS = (B^*)^A \ mod \ (P)$
- $DHS = (A^*)^B \ mod \ (P)$

Comparison and Combination

- RSA is more convenient because there is no need to distribute keys.
- DES is within two orders of magnitude faster.
- A viable combination is to distribute the secret keys using RSA, and then, for the bulk data to use DES.
- Similar combination is implemented in the Pretty Good Privacy (PGP) method.



Authentication Server

- Based on the fact that there exists a trustworthy authentication server.
- The authentication server provides a secure way for pair of processes to obtain secret keys.
- Needham & Schroeder suggested two mechanisms to construct such a server:
  - Authentication with secret keys.
  - Authentication with public keys.
- Kerberos is based on the secret-key method.

Secret-key Authentication

1. $A \rightarrow S: A, B, N_A$
2. $S \rightarrow A: \{N_A, B, K_{AB}, \{K_{AB}, A, timestamp\}_{K_B}\}_{K_A}$
3. $A \rightarrow B: \{K_{AB}, A, timestamp\}_{K_B}$
4. $B \rightarrow A: \{N_B\}_{K_{AB}}$
5. $A \rightarrow B: \{N_B - 1\}_{K_{AB}}$

A,B : the processes.
S : the server
N : a nonce
K : a key.

Only the server knows K_A and K_B

Digital Signatures

- Enable to verify that a message was originally produced by the signatory.
- Enable to verify that a message content was not subsequently altered.

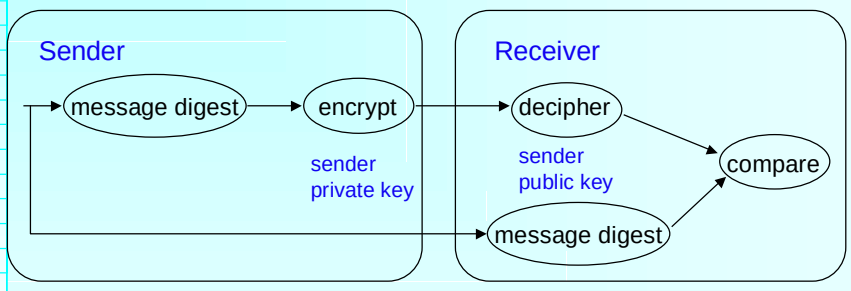
Handwritten signatures cannot entirely provide that.

Hard to detect forged signatures.
Hard to prevent alteration of the document.

Digital Signature with Public Key

- For each message we add a signature which is constructed as follows:
  - Compute a *digest function* of the message (like hash function) to reduce length.
  - Encrypt the result using our private key.
- The receiver
  - Deciphers the signature using our public key.
  - Computes the *digest function* on the document.
  - Compares the two to validate the document.

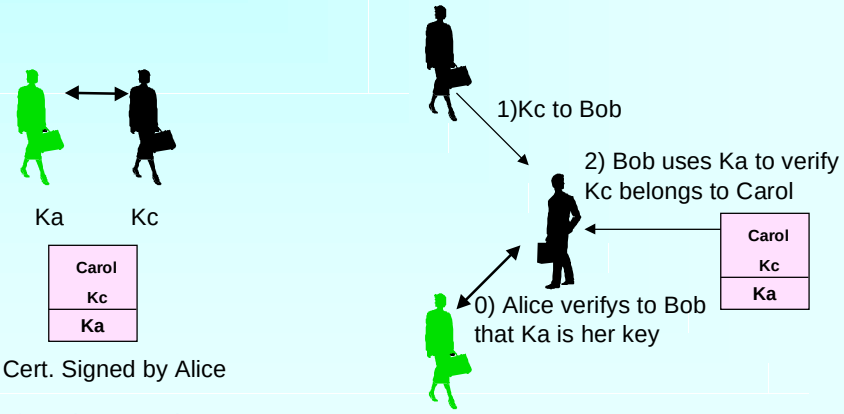
Digital Signature with Public Key



What if we would need to prevent others from reading the message?

Certificates

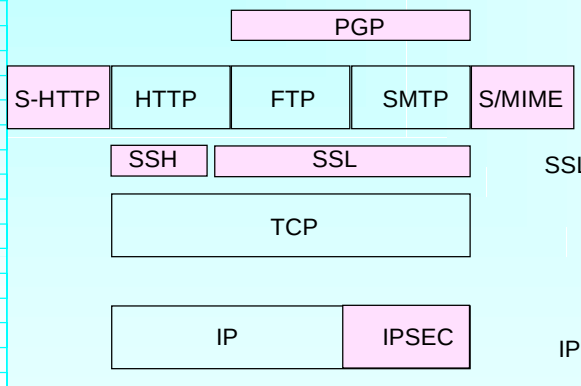
- Problem….How do I know who a particular public key belongs to?



Trust!

- Essential question is “Who do you Trust?”
  - Bank, Credit Card Issuer, State Govt., Feds, Hopkins?
- Second most important question…
  - Theory vs. Practice

Security Mechanisms at Various Levels



SSL - Secure Socket Layer uses public key to distribute private key that is used to encrypt content.

IPSEC - Packet level security for IP.

Course Evaluations!!!

Instructor: Yair Amir

Course Number: CS437/337

Course Title: Distributed Systems

TAs: Yan Gu and Sam Lang