Cryptography, main concepts

- P clear (plain) text, message - readable (intelligible) information
- C ciphertext - encrypted information
- E encryption (enciphering) - transforming clear text into ciphertext
- D decryption (deciphering) - transforming ciphertext back into original cleart text
- encrypting algorithm - a mathematical function having the following form:
  $C = E(P, K_e)$ where K_e encryption key
- decryption algorithm:
  $P = D(C, K_d)$ where K_d encryption key
- attacker (cryptoanalyst, intruder) - person that tries to discover C (compromise the encryption algorithm)
- two entities (users, programs) A and B need to communicate
  - if A has K_e , B has a matching K_d - A and B have a one way private secure communication channel
  - if also B has K_e and A has a matching K_d - A and B have a two way secure communication channel

Security methods, types of attack

- Two methods to achieve security:
  - conventional - encryption algorithm is designed to be rather complex and hard to guess
  - modern - encryption algorithm is made public but the keys are kept secret - the strength of the algorithm depends on the difficulty of determining K_d
- type of attack depends on information available to attacker:
  - ciphertext-only, attacker tries to discover K_d knowing C only
  - known-plaintext, plain text and ciphertext is known
  - chosen-plaintext, the attacker is able to obtain the ciphertext for any plaintext she chooses

Conventional cryptosystems

- the Caesar cipher: every letter is transformed into the third (or some other) in the alphabetical sequence (with wrap around):
  E=M->(M+3) mod 26
  - “advanced operating systems” -> “dgydqfhg rshudwlqj vcvwhpv”
- transformation is linear - the number of keys (shifts) is only 25 - easy to guess
- simple substitution cipher : an alphabet can be mapped to any permutation of letters
  - each permutation is a key - there 26! (> 10^{26}) keys. Exhaustive search is very expensive.
- substitution preserves frequency distribution of the letters of an alphabet - statistical analysis is possible.

Modern cryptosystems

- Symmetric - K_e and K_d are similar (possibly can be easily derived from one another) - not as computationally intensive as asymmetric. Useful if both encryption and decryption is performed by private parties

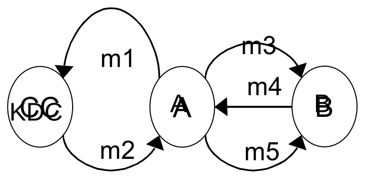
  Need secure channel to exchange keys between communicating parties. Can use insecure channel for encrypted messages transmission
  - Example - Data Encryption Standard (DES)
- asymmetric - K_e and K_d are dissimilar. It is (computationally) hard to derive K_d from K_e. K_e does not need to be kept secret. Computationally expensive and cannot be used for bulk data encryption

  Can use insecure channel for both key and message transmission
  - Example - method of Rivest-Shamir-Adleman (RSA)

Key distribution problem, Symmetric cryptosystems

- If two entities A and B want to communicate they must share K_e/K_d pair. How can the two entities exchange keys if there is no secure channel?
- Key distribution center (KDC) - a server that has a private secure two-way communication channels with A and B and holds a private key for each user
- three approaches
  - centralized - one KDC
  - fully distributed - every node is KDC
  - partially distributed - a group (region) shares a KDC

Key distribution, symmetric systems, centralized KDC



- The following messages are exchanged:
  - m1 = (R_a, ID_a, ID_b) - where R_a - code for request
    ID_a - id of process A
    ID_b - id of process B
  - m2 = E(R_a, ID_a, K_{ab}, C1), K_a) - where
    K_{ab} - secret key for communication between A and B
    C1=E((K_{ab}, ID_a), K_b)
    K_a - private key of A
  - m3 = C1
  - m4 = C2=E(N_r, K_{ab})
    where N_r is random number generated by B
  - m5 = C3=E(N_t, K_{ab})
    where N_t =f(N_r),
    and f is some previously defined function

Authentication types

- Authentication - verifying the identity of an entity before permitting access to the requested resource
  - login authentication - deals with verifying the identity of a user for the system at the time of login
  - One-way authentication of communication entities - verirfyes the identity of one of the two communication entities by the other entity
  - Two-way authentication of communication entities deals with mutual authentication (both communication entities verify each other's identity)

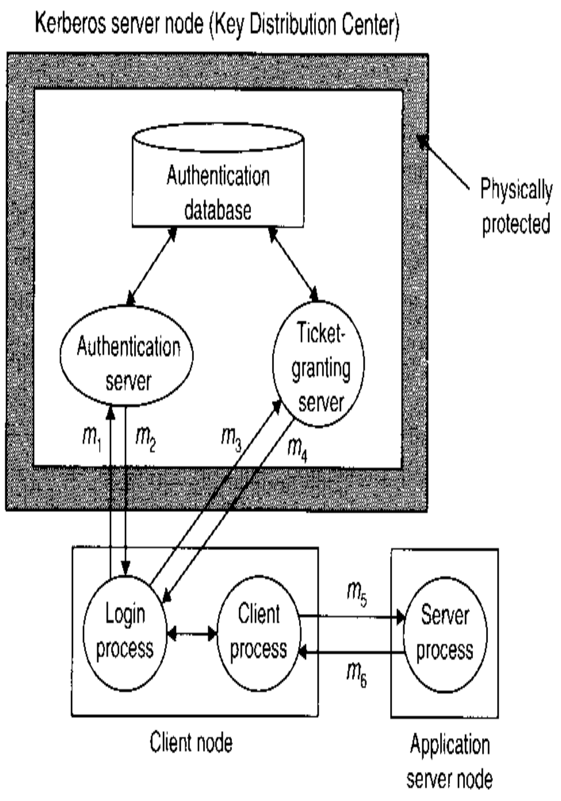
Approaches to Authentication

- approaches to authentication:
  - proof by knowledge - verifying something that can be known only by an authorized entity (principal). Example - typing a password
    - demonstration method - entity claims her identity by supplying the (secret) information
    - challenge-response method - answers correctly to a (challenge) question asked by verifier
  - proof by possession - entity produces an item that can only be possessed by an authorized principal. Example - using a magnetic card or a key.
  - proof by property - system measures some physical characteristic of an entity that is hard to forge

Kerberos

- Kerberos is a network authentication system
- developed at MIT in late eighties
- features:
  - authenticates users on untrusted network.
  - clear password is never send over the network
  - single sign-on facility (user enters password only once)
- Kerberos server - key distribution server, contains:
  - authentication database - contains user IDs and passwords (secret keys) for all users in the system
  - authentication server - verifies user's identity at the time of login
  - ticket-granting server - supplies tickets to clients for permitting access to other servers in the system
- client - runs on public workstations, obtains permission from Kerberos server for the user to access resources
- application server - provides certain service to client after authenticity of the client has been verified by the Kerberos server

Kerberos authentication protocol



Kerberos authentication protocol (cont.)

- A - authentication server, G - ticket-granting server, C - client, S - application server

ID_g = ticket-granting server's identifier	K_g = ticket-granting server's secret key
ID_c = client's identifier	K_1 = ticket-granting ticket session key
ID_s = applicationserver's identifier	K_2 = service-granting ticket session key
N_i = a nonce	T_{si} = starting time of validity of ticket
K_c = client's secret key	T_{ei} = ending time of validity of ticket
K_s = applicationserver's secret key	T_i = a timestamp

$$m_1 = (ID_c, N_1)$$

$$m_2 = C_2 = E((N_1, K_1, C_1), K_c), \text{ where } C_1 = E((ID_c, ID_g, T_{s1}, T_{e1}, K_1), K_g)$$

$$m_3 = (ID_s, N_2, C_1, C_3), \text{ where } C_3 = E((ID_c, T_1), K_1)$$

$$m_4 = C_5 = E((N_2, K_2, C_4), K_1), \text{ where } C_4 = E((ID_c, ID_s, T_{s2}, T_{e2}, K_2), K_s)$$

$$m_5 = (C_4, C_6), \text{ where } C_6 = E((ID_c, T_2), K_2)$$

$$m_6 = C_7 = E(T_3, K_2), \text{ where } T_3 = T_2 + 1$$

C1 - ticket-granting ticket, C4 - service-granting ticket

Problems with Kerberos

- Not effective against password guessing attacks
- an application has to be *Kerberized* (modified to work with Kerberos)
- after authentication the information is transmitted without encryption
- users' passwords are stored in unencrypted from on the Kerberos server
- loose clock synchronization is necessary, appropriate time to live for tickets needs to be configured

Secure shell (SSH)

- Ssh - authentication and encryption system
- Originally developed at Helisinki University of Technology now maintained by SSH Communications Security, Ltd
- features:
  - authentication of users on the untrusted network
  - clear passwords are never sent over the network
  - communication between machines is encrypted - multiple encryption algorithms available (the algorithms may be automatically selected)
  - communication may be (automatically) compressed
  - tunneling and encryption of arbitrary connections
- client-server architecture:
  - ssh server (daemon) - handles authentication, encryption and compression
  - ssh client - handles communication on the client side

SSH communication

- SSH uses both asymmetric (for authentication) and symmetric (for data encryption) cryptosystems.
- Stage 1. connection negotiation:
  - Each machine running SSH has a (asymmetric) key-pair called host key. A server key-pair is generated when a client contacts the server
  - when a client contacts a server, the server sends server and host's public keys.
  - client stores host's public key between sessions. The client compares the host's public key it receives with the one it stores to make sure the host is correct
  - client generates a random number to serve as session key. It encrypts this session key using both the host and server key
  - further communication is encrypted with session key using one of the symmetric data encryption methods, the communication can be optionally compressed

SSH communication (cont.)

- Stage 2. user authentication:
  - multiple authentication methods can be used:
    - a server has a list of *user x machine* pairs from which it accepts connections without further authentication
    - a server possesses a list of pairs *user's pubic key x machine,* the client has a private key and has to authenticate itself by providing a nonce encrypted with this private key
    - password authentication - server stores *login x password* of the client (or can verify it with a password server)
- Stage 3. command execution:
  - after authentication a client requests a command to be executed on a server - may be a shell command