KERBEROS

Distributed OS Concepts & designs
by Pradeep K. Sinha

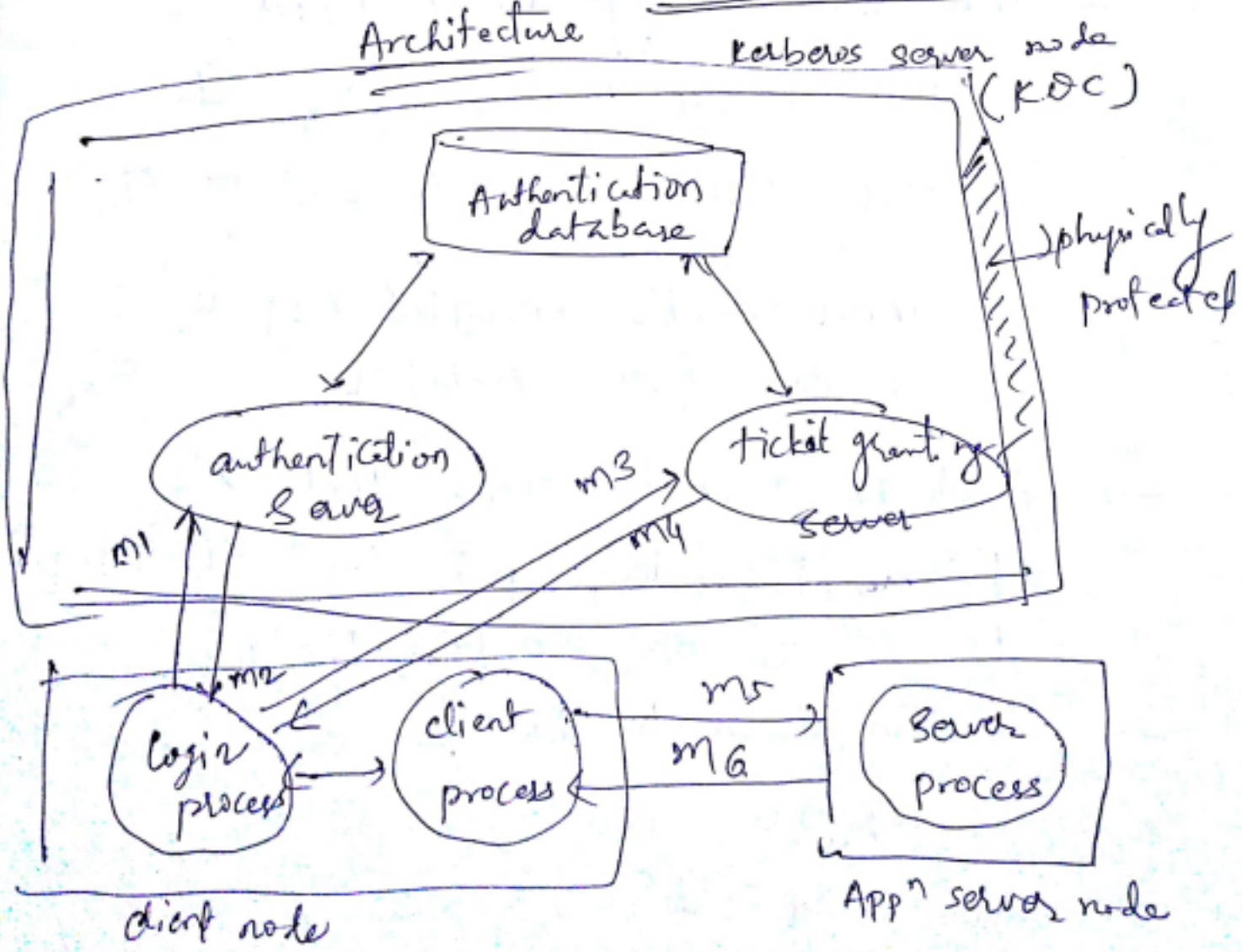
- Kerberos
- X.509 authentication services.

→ Kerberos is a nw authentication system developed by MIT.

- of several authentication sys. developed till date, Kerberos is most popular & it is continuing till date.

(Kerberos protocol)

Architecture



$m_1 = (ID_c, N_1)$

$m_2 = c_2 = E((N_1, K_1, G), K_c)$

$c_1 = E_{K_g}((ID_c, ID_g, T_{s1}, T_{e1}, K_1), K_g)$

$m_3 = (ID_s, N_2, G, c_3)$ (authenticator)

$c_3 = E((ID_c, T_i), K_1)$

$m_4 = c_5 = E((N_2, K_2, c_4), K_1)$

$c_4 = E((ID_c, ID_s, T_{s2}, T_{e2}, K_2), K_s)$

$m_5 = (c_4, c_6)$

$c_6 = E((ID_c, T_2), K_2)$

$m_6 = c_1 = E(T_3, K_2)$,

$T_3 = T_{2+1}$

ID_g = ticket granting server identifier

ID_c = client identifier

ID_s = app^n server identifier

N_i = Nounce

K_c = client's secret key

K_s = app^n server secret key

K_g = ticket granting server's secret key

K_1 = ticket granting ticket session key

K_2 = Service-granting ticket session key

T_{si} = starting time of validity of ticket

T_{ei} = ending time of validity of Tckt

T_i = Time stampt.

1) Kerberos Server :⇒

- It is a key component of kerberos system.
- It acts as key distribution system.
- each kerberos server has an authentication database, authentication server & ticket granting server.
- The authentication db has ID & password of all users of the system.
- kerberos server shares a unique secret key with each server of the system. Therefore authentication database also has server ID & secret key for all servers in the sys.
- The password & the secret keys are distributed physically or in some other manner as a part of kerberos installation.
- kerberos uses DES algo to generate keys & encrypt msg, but this is implemented as a separate module that can be easily replaced by any other suitable algo.
- The authentication server performs the task of verifying user's identity at the time of login without requiring password to travel over the network.

- kerberos has a single sign on facility
- thats way a user has to enter his/her password only once at the time of login no matter how diff. resources are accessed by the user after that.

Ticket granting server

- It performs the task of supplying tkts for permitting access to other servers in the system.
- These tkts are used to establish secure logical comm^n channel betn client & saved by performing mutual authentication

→ since kerberos server has valuable information in its authentication db that must be kept secret, it is extremely imp that it be installed on 'carefully protected & physically secure machine.

2) Client :

→ 2nd component of kerberos system is comprised of client processes that usually run on workstations, located in public places where their cousins are available to whatever user happens to be physically in front of

- Therefore they are entrusted
- users (client process) must get their identification varified by Kereberos server before attempting to access any other server in the system.
- Once the identity is varified, each client process must obtain ~~tiket~~ tkt from the tkt granting server for communication with a server that it wants to access

3) application server

- A server provides a specific type of service to client upon request only after verifying the authenticity of the client
- The server runs on a machine that is kept in a secure room.
- Therefore kerberos ensures that a compromise of one server does not compromise another server.

Kerberos Authentication Protocol

A – authentication server
G – ticket granting server
C – client → Kc
S – application server

Ka
Kg → secret ke
Ks

Kc - secret key of client
generated from user's password
by using one way function.

1. When user logs on to a workstation
typing login name,
the login program sends a request t
the authentication server for what
known as ticket-granting tkt in m
m1.

m1 contains user ID (ID_C) & a
nounce n1, that is used to c
validity of the reply.

- This msg is sent in plaintext form.

$m1 = ID_c, N1$

2. on receiving m1, the authentication server extracts the password of this user from the authentication database.

- It then generates a random number for use as a session key K_1.
- After this it creates a ticket granting ticket that contains user's ID (ID_c), the ticket granting ID (ID_g), starting time for validity of ticket (T_{s1}) ending time for the ticket (T_{e1}) & a copy of a session key (K_1).
- Now it encrypts this ticket by using ticket granting server's secret key (K_g) to generate a Ciphor text C1.

$$C1 = E((ID_c, ID_g, T_{s1}, T_{e1}, K_1), K_g)$$

- This encryption ensures that no one can temper with ticket granting tkt & only kerberos server can decode it.

- Next it uses the client's secret key (K_c) to generate C_2

$$C_2 = E((N_1, K_1, C_1), K_c)$$

- & it then returns C_2 to the login program in msg $m2$.

$$m2 = C_2 = E((N_1, K_1, C_1), K_c)$$

$$C_1 = E((ID_c, ID_g, T_{s1}, T_{e1}, K_1), K_g)$$

3) On receiving $m2$, the login program prompts the user for the password.

- The entered password is run through one way function that generates client's secret key K_c

- The password is then removed from the computer's memory to minimize the chance of password discloses.

- The login prog. then decrypts $c2$ by using K_c.

- If user supplied the correct password C_2 is successfully decrypted ~~Bo the~~

and login prog obtains the nounce, session key & the encrypted ticket from the msg.

- It checks the nounce for validity & if the reply & stores the session key & encrypted tkt for use when communicating with tkt-granting serv.

- When this has done, the client's secret key can also be erazed from the memory, since now tkt is used to authenticate the user.

- A login session is then started.

- user authentication is done without requiring password to travel over the n/w.

if an intruder intercepts the reply msg, it will be unable to decrypt it & thus unable to obtain session key & the tkt inside it.

4. When client process wants to access appn server,

- it requests tkt-granting server for tkt-granting tkt that can be used to communicate with ~~authentication~~ app server.
- for this client creates an authenticator by using session key Kt. that contains client ID (ID_c) & Timestamp (T_1)
- It encrypts this authenticator by using session key (K1) to obtain.

$C3 = E((ID_c, T_1), K_1)$

- This authenticator is intended for one time-stamp only & has a very short span (few minutes)
- The client sends encrypted C3 authenticator, encrypted tkt granting tkt C1, ID of appn server (ID_s) & nonce N2 to tkt granting server in msg m3.

$m3 = (ID_s, N2, C_1, C_3)$ $C_3 = E((ID_c, T_1), K_1)$

5. on receiving C3 m3,
tkt granting server decrypts C1 by using
secret key (Kg) & make sure that
it has not expired by comparing Te,
with current time.
- it extracts session key K1 &
uses it to decypt C3 to obtain
IDc & T1.
- The obtained IDc is compared with the
value of IDc in the tkt-granting tikt
to authenticate the source of req.
- If all varifications pass successfully,
the tkt-granting serva gets assured
that the sender of the tkt is indeed
the tkt's real owner.

→ here authenticator C3 proves the identity
not tkt granting tkt,
- The use of tkt granting tkt is a way to
distribut keys securely

- after successful authentication of client the tkt-granting server generates a new random session key (K_2)

- then creates a reusable service-granting tkt for access to the requested server.

- This tkt contains client ID (ID_c)
  app^n server ID (ID_s)
  starting time of validity of tkt (T_{s2})
  ending time of validity of tkt (T_{e2})
  copy of new session key (K_2)

- it then encrypts service granting tkt with secret key of app^n server (K_s)

to obtain $C_4 = E((ID_c, ID_s, T_{s2}, T_{e2}, K_2), K_s)$ —

$C_4 = E(ID_c, ID_s, T_{s2}, T_{e2}, K_2), K_s)$.

- only app^n server or kerberos server can decode it

- next it uses old session key K_1 to generate $C_5 = E((N_2, K_2, C_4), K_1)$
  it then returns C_5 to client msg m_4
  $m_4 = C_5 = E((N_2, K_2, C_4), K_1)$

- Next it uses the client's secret key (K_c) to generate C_2

$$C_2 = E((N_1, K_1, C_1), K_c)$$

- It then returns C_2 to the login program in msg m_2.

$$m_2 = C_2 = E((N_1, K_1, C_1), K_c)$$

$$C_1 = E((ID_c, ID_g, T_{s1}, T_{e1}, K_1), K_g)$$

3) On receiving m_2, the login program prompts the user for the password.

- The entered password is run through one way function that generates client's secret key K_c

- The password is then removed from the computer's memory to minimize the chance of password discloses.

- The login prog. then decrypts c_2 by using K_c.

- If user supplied the correct password C_2 is successfully decrypted ~~Both~~

and login prog obtains the nounce, session key & the encrypted ticket from the msg.

- It checks the nounce for validity & if the reply R stores the session key & encrypted tkt for use when communicating with tkt-granting serva.

- When this has done, the client's secret key can also be erazed from the memory, since now tkt is used to authenticate the user.

- A login session is then started.

- user authentication is done without requiring password to travel over the n/w.

if an intruder intercepts the reply msg, it will be unable to decrypt it & thus unable to obtain session key & the tkt inside it.

4. When client process wants to access appn server,

- it requests tkt-granting server for tkt-granting tkt that can be used to communicate with ~~authentication~~ app server.
- for this client creates an authenticator by using session key Kt. that contains client ID (ID_c) & Timestamp (T_1)
- It encrypts this authentication by using session key (K_1) to obtain.

$$C3 = E((ID_c, T_1), K_1)$$

- This authenticator is intended for one time-stamp only & has a very short span (few minutes)
- The client sends ~~C3~~ encrypted C3 authenticator, encrypted tkt granting tkt C1, ID of appn server (ID_s) & nonce N_2 to tkt granting server in msg m3.

$$m3 = (ID_s, N_2, C_1, C_3) \qquad C_3 = E((ID_c, T_1), K_1)$$