An Introduction to Kerberos

CS60002: Distributed Systems

Bhaskar Pal
Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur
Private Key Cryptography

- Same key is used for encryption and decryption
- There is a trusted authority called the authentication server (AS)
  - Keeps the secrets
- Every user shares its private secret key with AS
  - User X doesn’t know the private key of user Y
- Kay Distribution: When X wants to communicate with Y, they need to use a secret key between them
  - AS is responsible for distributing this session key (conversation key) between X and Y
- Everybody has to trust AS
How it works?

• AS knows the private keys of Alice and Bob
• Alice and Bob requires a session key
  – Alice doesn’t know the private key of Bob
• How the session key is transmitted to Alice and Bob
A simple overview

- Alice → AS : Alice, Bob
- AS recovers $K_A$ and $K_B$ and creates a session key $K_{A,B}$
- It makes two copies of $K_{A,B}$
  - One is for Alice and is encrypted using Alice’s private key
  - One is for Bob, encrypted using Bob’s private key (ticket)
    - Also included the identity of Alice
- The ticket conveys the following to Bob
  - I am AS (only AS knows the private key of Bob)
  - Alice wants to communicate with you
  - Only you and Alice (except me) have the knowledge of $K_{A,B}$
  - If some one proves that she has the knowledge of $K_{A,B}$ -- it is Alice
Overview Contd..

- $\text{AS} \rightarrow \text{Alice} : E_{K(A)} \{\text{Bob, } T_{A, B}, K_{A, B}, \text{timestamp, } ..\}$
- $T_{A, B} = E_{K(B)} \{ \text{Alice, Bob, } K_{A, B}, ... \}$

- Alice sends the Ticket to the Bob
  - Adding an authenticator to prove its authenticity
    - Ticket can be replayed by some intruder

- Authenticator
  - $E_{K(A, B)} \{\text{Alice, timestamp}\}$

- The session key recovered from the ticket is used to decrypt the authenticator

- Timestamp checks for replay of Authenticator

- Mutual Authen. : $\text{Bob} \rightarrow \text{Alice} : E_{K(A, B)} \{\text{timestamp + 1}\}$
Kerberos Basics

• Kerberos is an authentication protocol implemented on Project Athena at MIT

• Athena provides an open network computing environment

• Each user has complete control of its workstation

• The workstations can not be trusted completely to identify its users to the network services

• Kerberos acted as a third party authenticator
  – Helps the user to prove its identity to the various services and vice versa
Kerberos Basics

- It is based on symmetrical cryptographic algorithms (private key cryptosystems)
  - Same key is used for encryption as well as decryption
  - Uses DES

- Every user U has a private key that can be obtained by
  - $K_U = f(password)$

- Every users private key is also known to Kerberos
  - Kerberos maintains a database of its users and their private keys

- Kerberos uses this private key for communicating any message to the user
  - User is convinced about Kerberos’s authenticity

- If an user U gets a message encrypted using its private key
  - The message must be from Kerberos
  - In case of replays? 
Kerberos Basics

- Kerberos requires the workstations to be synchronized.
- A *timestamp* which is the current time of the sender is added in the message to check for any replays.
- The receiver checks for the timeliness by comparing its own clock value with that of the *timestamp*.
  - Timely if *timestamp* is equal to the local clock value.
The Basic notion

- To request a service from a server, the client goes through three phases of authentication

- Phase 1
  - The client requests a ticket from the Kerberos
  - Kerberos grants a ticket and a session key
  - The ticket is used for requesting other tickets for various services
  - Ticket conveys the identity of the client to the server
  - The session key is used for conversation between the client and the server
Basic notions

- Phase - 2
  - The client uses the ticket of the first phase to request a ticket from the ticket granting server (TGS) for a specific service

- Phase 3
  - The client presents the key to the server for the service
Protocols

• The three phases of authentication is achieved via two authentication protocols

• The user-authentication protocol (1st Phase)
  – Verifies the authenticity of the user and grants the initial ticket and the session key

• Client - Server authentication protocol (2nd & 3rd phases)
  – Mutual authentication of a client and a server

• Hierarchy
  – Medium-term session key (TGT) – get once and use for requesting other sessions
  – Short-term key – used for a particular service
Phase - 1 (Getting the Initial Key)

- $U \rightarrow C : U$
- $C \rightarrow K : U, tgs$

\[ (1) \]

- Kerberos finds out $K_U$ and $K_{tgs}$
- It creates the session key $K_{U, tgs}$
- It creates the ticket
  \[ T_{U, tgs} = E_{K(tgs)}(U, tgs, K_U, tgs, timestamp, life) \]

- $K \rightarrow C : E_K(U) \{ T_{U, tgs}, K_U, tgs, tgs, timestamp, life \}$

\[ (2) \]
Phase - 2 (Getting Server Tickets)

- C → TGS : S, T_U, tgs, A_U
- A_U is the authenticator : E_{K(U, tgs)} \{C, timestamp\}
- Intruder can replay S, T_U, tgs
  - Session key is used to verify first level of authenticity
  - Session key may be the same in a session
  - timestamp is used for second level of authenticity
- Ticket: T_C, S = E_{K(S)} (C, S, K_C, S, timestamp, life)
- TGS → C : E_{K(U, tgs)} \{T_C, S, K_C, S, timestamp, life\}
Phase - 3 (Requesting the Service)

- $C \rightarrow S : T_{C,S}, A_C$  \hspace{1cm} (5)

- $A_C$ is sent to prevent foul play by the intruder
  - $E_{K(C,S)}(C, \text{timestamp})$

- $S \rightarrow C : E_{K(C,S)}\{\text{timestamp + 1}\}$ \hspace{1cm} (6)
Why two servers?

• **Note that**
  – First phase is used for user-authentication (using the id and password)
  – Second and third phase may continue several times with the same TGT granted by the first phase

• **In absence of this additional phase**
  – For each service, the user needs to authenticate itself using its **password**
  – Once the intruder gets the first session key, it can continue doing malicious works throughout the session
  – That’s why **life** and **timestamp** are mentioned