Lab 6: Implementation and Application of Kerberos

Kerberos is a key distribution and user authentication service. It provides a centralized server to authenticate users to servers and servers to users. There are different versions of Kerberos, and in this lab, we are going to implement Version 4, which makes use of DES.

The detailed steps of message exchanges of Kerberos 4

This is from *Page 19* of the slide *Lect04-small.pdf* of this course, which is under *Key Distribution and User Authentication [Sept 16]*. You can refer to that slide or the end of this instruction if you need other information.

Summary of Kerberos Version 4 Message Exchanges

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V $Ticket_v \| Authenticator_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$(for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

Task:

It is an **individual** project. Firstly, you need to create a chat program (socket) to establish the communication between client and servers. Basically, you will need a client **C**, and two servers, one of them serves as **V**, the other serves as both **AS** and **TGS**. After establishing the communication, you can send messages step by step following the order shown in the figure above. In step 3 and 5, you should check if the received tickets are still valid (do not expire), which will be explained below. Note that "||" means concatenation.

Some parameter settings:

1. Use these fixed ID in your codes:
   ID_c = "CIS3319USERID"
   ID_v = "CIS3319SERVERID"
   ID_{tgs} = "CIS3319TGSID"
2. Network address of client: AD_c = "127.0.0.1:{port}"
   {port} here is the port number you use.
3. TS: Use Unix time (aka. Epoch time) in seconds as timestamp.
4. Lifetime2 = 60 and Lifetime4 = 86400 (in seconds)
5. Keys: K_c, K_{tgs}, K_v are pre-shared keys between C and AS, AS and TGS, TGS and V, respectively. Similar to lab 1 and 2, generate the keys in advance and load them in the proper process. The other keys are generated and transmitted on the fly.

Check the validity of tickets:

In step 3 and 5, **TGS** and **V** need to check if the $Ticket_{tgs}$ and $Ticket_v$ are expired. For example, if $(current\ Unix\ time - TS_2) < Lifetime_2$, then $Ticket_{tgs}$ is still valid, otherwise, it's expired.

Printout:

Your codes are supposed to show the following information on screen for each step when executed.
step (1): print out the received message on **AS** side.
step (2): print out the plaintext of the received ciphertext, as well as $Ticket_{tgs}$ on **C** side.
step (3): print out the received message and validity (valid or not) of $Ticket_{tgs}$ on **TGS** side.
step (4): print out the plaintext of the received ciphertext, as well as $Ticket_v$ on **C** side.
step (5): print out the received message and validity (valid or not) of $Ticket_v$ on **V** side.
step (6): print out the plaintext of the received ciphertext, which should be TS_5+1 on **C** side.

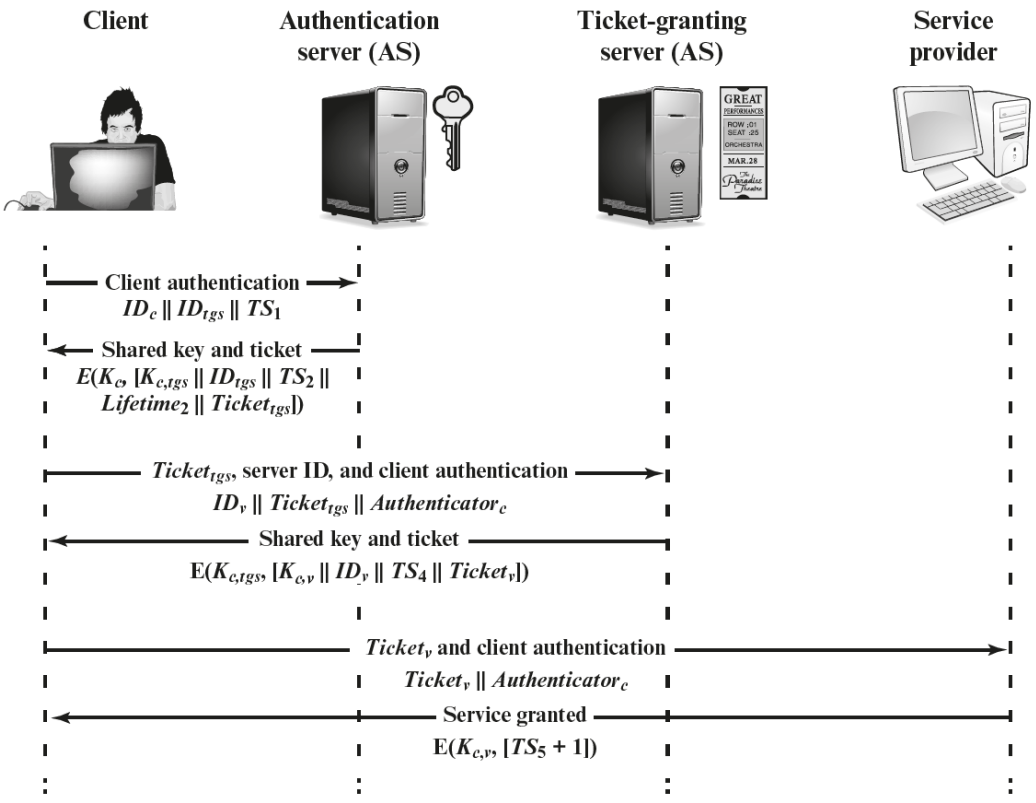
Submission

Submit the following files in a **zip** onto Canvas.

1. All of your source code files for this lab. Code comments will be helpful for the TA to understand your code.
2. A README file including information:
   a) Which language and external libraries you use.
   b) Which IDE you use.
   c) Detailed steps about how to run your code to finally obtain the required outputs. TA will check and run your code based on this instruction. If your code cannot be successfully executed, points will be deducted. **This is the most important part used to evaluate your work, so please be as specific as you can.**
3. A lab report including screenshots of each step of the entire code execution procedure, **as clear and specific as you can**. Alternatively, you can choose to use screen recording to record the entire code execution procedure and submit the recorded file.

Due: 11am, March 30, 2021.

Kerberos Exchanges



Notations

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

(a) Authentication Service Exchange

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (4)	TGS returns service-granting ticket.
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering.
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay.

$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

(b) Ticket-Granting Service Exchange

Message (5)	Client requests service.
$Ticket_V$	Assures server that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (6)	Optional authentication of server to client.
$K_{c,v}$	Assures C that this message is from V.
$TS_5 + 1$	Assures C that this is not a replay of an old reply.
$Ticket_v$	Reusable so that client does not need to request a new ticket from TGS for each access to the same server.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_V	Assures server that it has decrypted ticket properly.
TS_4	Informs server of time this ticket was issued.
$Lifetime_4$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_c	Must match address in ticket to authenticate ticket.
TS_5	Informs server of time this authenticator was generated.

(c) Client/Server Authentication Exchange