# Lab 5: 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 the slide *CIS_3319_Chapter_3.5_Kerberos.pdf* of this course, and you can refer to this slide if you need other information, like what each notation stands for.

### Kerberos Version 4 Message Exchanges

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

**(1) C → AS:** $ID_c \| ID_{tgs} \| TS_1$

**(2) AS → C:** $E_{K_c}\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

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

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

**(4) TGS → C:** $E_{K_{c,tgs}}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| AD_C \| TS_3\right.$$

**(c) Client/Server Authentication Exchange: to obtain service**

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

**(6) V → C:** $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{c,v}}\left[ID_C \| AD_C \| TS_5\right]$$

## Task:

This time, it's 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. $AD_c$ = "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.
2. A README file including information:
   a) which language and external libraries you use.
   b) which IDE you use.
   c) other details about how to run your code step by step (if any). TAs will look at

and run your code, and check the printouts.

**No demo or report needed.**

**Due:** 5pm on April 10.