6.915 Computer and Network Security	Fall Term, 1995

Lecture 25 : December 7, 1995

Lecturer: Ron Rivest *Scribe: Ellis Chi*

Professor Rivest suggested us talk about their end of term projects in next lecture. We were encouraged to come to the lecture on next Tuesday, December 11, and prepare a 3-minute presentation introducing the topic of the paper and summarizes the approach to the problems and, if any, interesting problems for research in the future.

Today's lecture is about Kerberos, the authentication and authorization protocol used by MIT Project Athena. There is one handout for this lecture. The first paper in the handout, *Authentication and authenticated key exchange* from William Stallings' *Network and Internetwork Security*, gives a general idea about Kerberos and is quite easy to understand.

In the lecture, we talked about the difficulty of having an authentication system in a distributed network. Then we talked about the development of Kerberos, and the idea of implementing Kerberos in large computer network.

1 Kerberos

1.1 Authentication in distributed system

The difficulty of implementing an authentication system is that authentication is hard to deal with in distributed systems. Say there is a client who wants to get a service from a server, e.g. getting a file ¿from a server, there is a question on how the server knows if the user is authorized to access the file.

1.2 Version 1: A Simple Authentication Protocol

In version 1 of an authentication protocol, it has client, server, and an authenticated server (AS). AS contains every authorized clients' passwords. To authenticate, the client sends his/her password to AS in plaintext. Then AS sends the client back a credential, represented in the form of a ticket, to tell the server that the client is authorized. The protocol is as follows:

1. The client requests for a ticket by sending his/her ID and password in plaintext.
2. If ID exists, AS checks the password received against its database. If matched, the server sends the client a ticket.
3. The client uses the ticket to obtain services from the server.

There is one major weaknesses in this protocol. the client's password and the ticket are both sent in plaintext.

1.3 Version 2: Encrypted Ticket

Version 2 is the same as the simple authentication protocol presented on p.317 of *Authentication and authenticated key exchange*. In this protocol, the password is still sent in clear text. However, the ticket is encrypted with the secret key shared by AS and the server. This prevents the content of the ticket from being altered, and another client with a different ID cannot use the ticket since the ID_c is embedded in the encrypted ticket.

1. C $\rightarrow$ AS: ID_C, P_C, ID_V
2. AS $\rightarrow$ C : Ticket
3. C $\rightarrow$ V : ID_C, Ticket

- C = client
- AS = authentication server
- V = server
- ID_C = identifier of user on C
- ID_V = identifier of V
- P_C= password of user on C
- AD_C = network address of C

- K_V = secret encryption key shared by AS and V

The major weakness of this protocol is that the client needs to request a new ticket when he/she needs to talk to another server. This requires either the client enter the password every time he/she talks to a new server, or the workstation buffer the client's password, and both methods are undesirable.

1.4 Version 3: Ticket Granting Service

Version 3 is the same as the improved authentication protocol presented on p.319 of *Authentication and authenticated key exchange*. This protocol has a few major improvement. First, this protocol introduces a new component called ticket granting server (TGS). TGS grants a ticket for a requested service to a user who proves his/her authenticity by showing the ticket obtained from AS. AS and TGS can be physically in the same server, but they are logically different. AS only deals with user authentication, while TGS gives out tickets for services requested by authenticated users.

Separating the work between TGS and AS provides an abstraction between authentication and authorization. A client can be an authenticated user (having a valid ID and account), but he/she might not be authorized to use certain services in the authenticated domain. For example, a student user is authenticated Athena user but cannot access the lockers of the classes that he/she is not registered for. Moreover, with a well-specified interface, each service administrator can set up his/her own TGS which can then be connected to the AS through a "central TGS", and the design becomes more flexible.

Another improvement is that the password is not being sent in clear text any more; instead, AS sends a *challenge*, a ticket decrypted only by the client's password.

Besides the initial greeting given by the user, each message transmitted is associated with a *timestamp*, and the *lifetime* of the message. The timestamp is used to prevent a message replay, and the lifetime is issued to a session to prevent an adversary from being able to use the session forever.

When a user logs in:

1. C $\rightarrow$ AS: ID_C, ID_{tgs}
2. AS $\rightarrow$ C: $E_{Kc}[Ticket_{tgs}]$

When a user acquires a ticket for a new service:

1. C $\rightarrow$ TGS: ID_C, ID_v, $Ticket_{tgs}$
2. TGS $\rightarrow$ C: $Ticket_v$

When a user starts a service session:

1. C $\rightarrow$ V: ID_C, $Ticket_v$

- $Ticket_{tgs} = E_{Ktgs}[ID_C,\ AD_C,\ ID_{tgs},$
- $TS_1,\ Lifetime_1]$
- $Ticket_v = E_{Kv}[ID_C,\ AD_C,\ ID_v,$
- $TS_2,\ Lifetime_2]$

1.5 Kerberos Version 4

Version 4 is presented on p.322 of *Authentication and authenticated key exchange.*

When a user logs in:

1. C $\rightarrow$ AS: ID_C, ID_{tgs}, TS_1
2. AS $\rightarrow$ C: $E_{Kc}[K_{C,tgs},\ ID_{tgs},\ TS_2,\ Lifetime_2,\ Ticket_{tgs}]$

- $Ticket_{tgs} = E_{Ktgs}[K_{C,tgs},\ ID_C,\ AD_C,\ ID_{tgs}, TS_2,\ Lifetime_2]$

When a user acquires a ticket for a new service:

1. C $\rightarrow$ TGS: ID_V, $Ticket_{tgs}$, $Authenticator_C$
2. TGS $\rightarrow$ C: $E_{Kc,tgs}[K_{C,V},\ ID_V,\ TS_4,\ Ticket_v]$

- $Ticket_V = E_{Kv}[K_{C,V},\ ID_C,\ AD_C,\ ID_V,\ TS_4,\ Lifetime_4]$
- $Authenticator_C = E_{Kc,tgs}[ID_C,\ AD_C,\ TS_3]$

When a user starts a service session:

1. C $\rightarrow$ V: $Authenticator_C$, $Ticket_v$

2. V $\rightarrow$ C: $E_{Kc,v}[TS_5 + 1]$

2 Expanding Kerberos domain

Kerberos can be implemented in a larger computer network, e.g. a company trying to connect the Kerberos in different sites. This can be done by setting up local Kerberos networks and granting tickets to obtain services through other TGS's. When a user tries to authenticate him/herself for a service in a remote site, the local TGS authenticates the user first, then grants a remote service through the remote TGS. The TGS's can be connected in different ways. For example, they can be all connected to one another (a complete graph), or hierarchy (trees). The tree structure is supported in Kerberos Version 5.

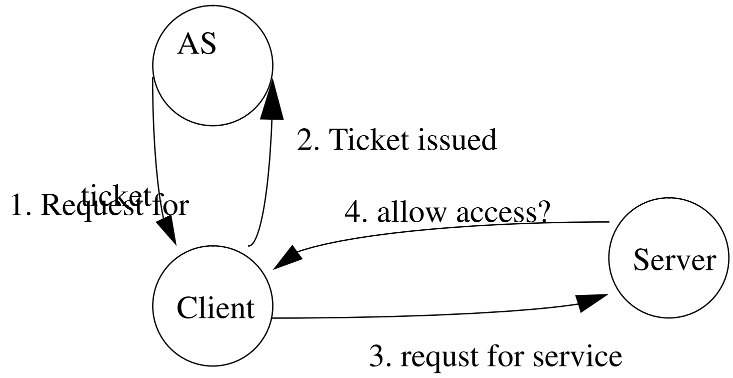

Figure 1: Version 1 authentication protocol. Here password is sent in plaintext.

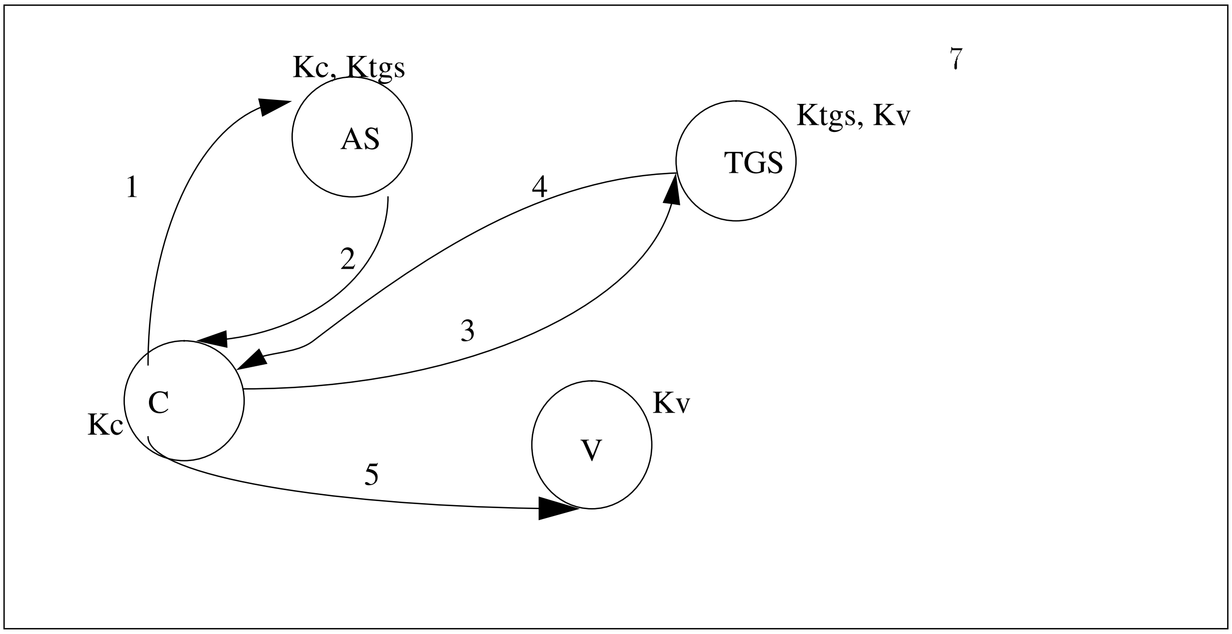

Figure 2: Version 3 authentication protocol with ticket encrypted