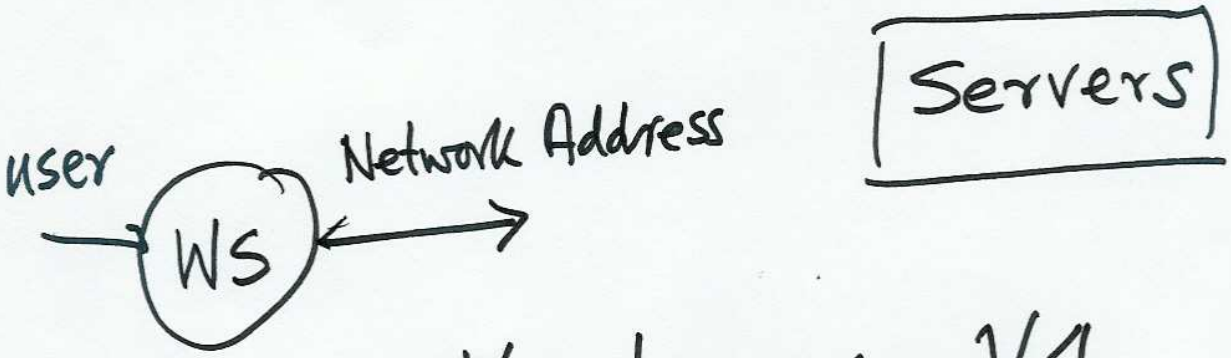
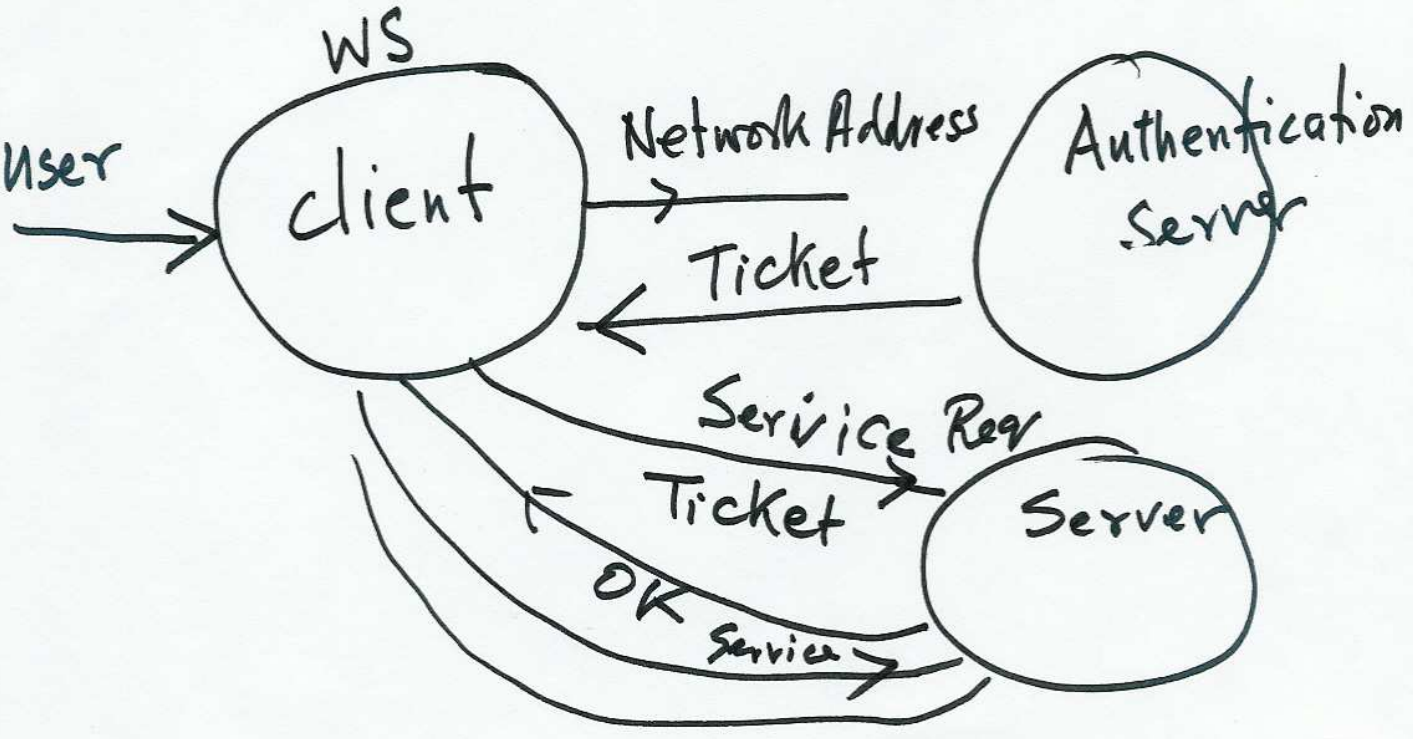
3/1/07

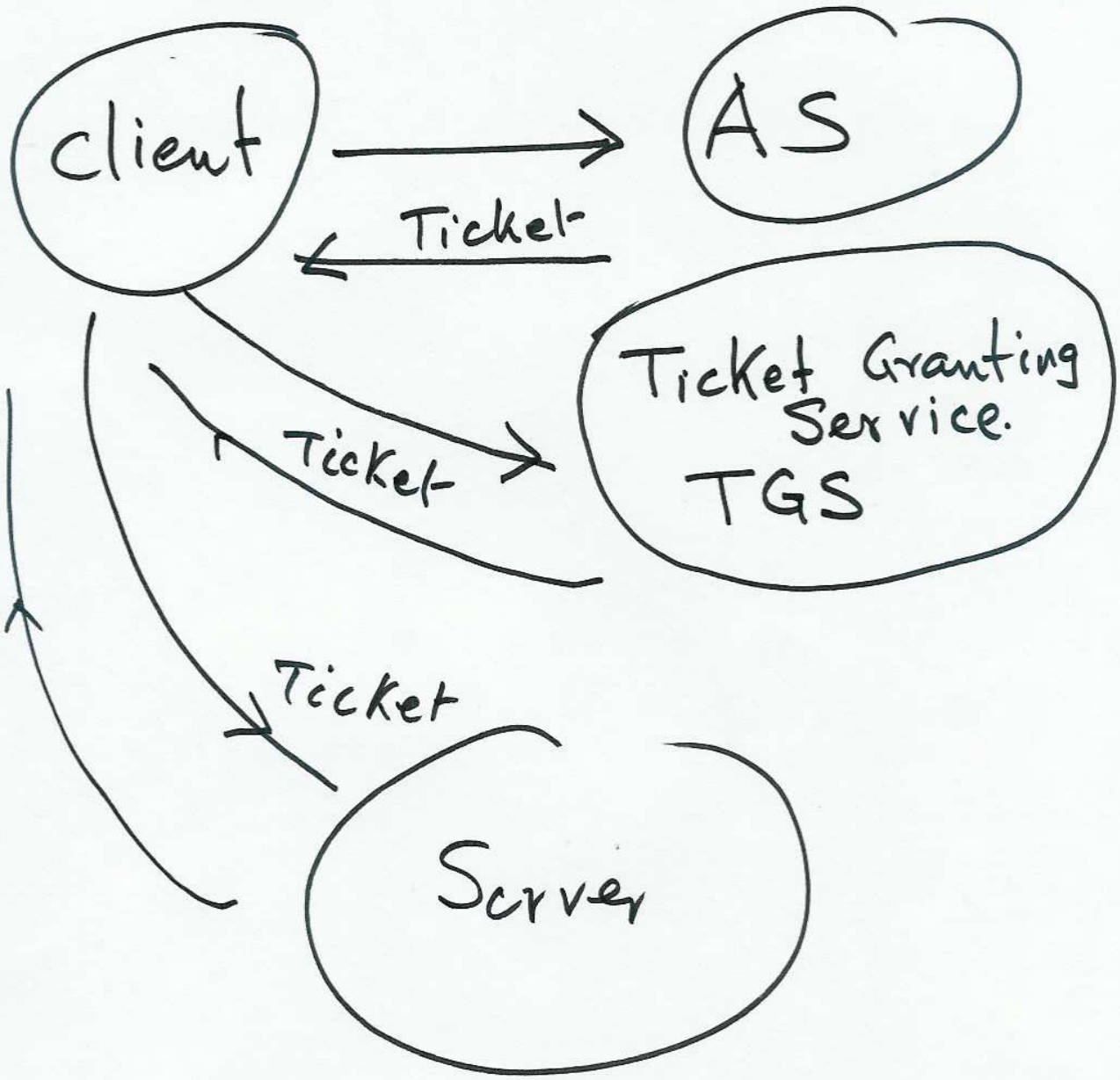
Kerberos

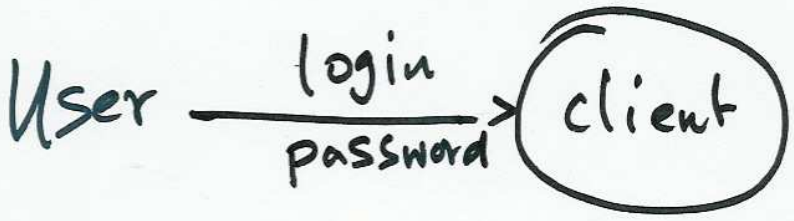
Athena



Kerberos - V4




client
AS
Ticket
Ticket Granting Service.
TGS
Ticket
Ticket
Server



$$C \longrightarrow AS: \quad ID_c \| ID_{TGS} \| TS_1$$

$$AS \rightarrow C: \quad E_{K_c}\left[K_{c,TGS} \| ID_{TGS} \| TS_2 \| Lifetime_2 \| Ticket_{TGS}\right]$$

$$Ticket_{TGS} = E_{K_{TGS}}\left[K_{CTGS} \| ID_c \| AD_c \| ID_{TGS} \| lifetime2\right]$$

$$C \rightarrow TGS: \quad ID_v \| Ticket_{TGS} \| Authenticator_c$$

$$TGS \rightarrow C: \quad E_{K_{CTGS}}\left[K_{cv} \| ID_v \| TS_4 \| Ticket_v\right]$$

$$Ticket_v = E_{K_v}\left[K_{cv} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime\ 4\right]$$

$$Authenticator_C = E_{K_{TGS}}[ID_C \| AD_C \| TS_3]$$

$C \rightarrow V_1$: $Ticket_V \| Authenticator_{C'}$

$$Authenticator_{C'} = E_{K_{CV}}[ID_C \| AD_C \| TS_5]$$

$V_1 \rightarrow C$: $E_{K_{CV}}[TS_5 + 1]$

PKI - X.509

Table 14.3 Summary of Kerberos Version 5 Message Exchanges

(1) C → AS Options$\|ID_c\|Realm_c\|ID_{tgs}\|Times\|Nonce_1$
(2) AS → C $Realm_c\|ID_C\|Ticket_{tgs}\|\mathrm{E}(K_c, [K_{c,tgs}\|Times\|Nonce_1\|Realm_{tgs}\|ID_{tgs}])$
$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags\|K_{c,tgs}\|Realm_c\|ID_C\|AD_C\|Times])$

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

(3) C → TGS Options$\|ID_v\|Times\|\|Nonce_2\|Ticket_{tgs}\|Authenticator_c$
(4) TGS → C $Realm_c\|ID_C\|Ticket_v\|\mathrm{E}(K_{c,tgs}, [K_{c,v}\|Times\|Nonce_2\|Realm_v\|ID_v])$
$Ticket_{tgs} = \mathrm{E}(K_{tgs}, [Flags\|K_{c,tgs}\|Realm_c\|ID_C\|AD_C\|Times])$
$\mathrm{Ticket}_v = \mathrm{E}(K_v, [\mathrm{Flags}\|K_{c,v}\|\mathrm{Realm}_c\|\mathrm{ID}_C\|\mathrm{AD}_C\|\mathrm{Times}])$
$Authenticator_c = \mathrm{E}(K_{c,tgs}, [ID_C\|Realm_c\|TS_1])$

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

(5) C → V Options$\|\mathrm{Ticket}_v\|\mathrm{Authenticator}_c$
(6) V → C $\mathrm{E}_{K_{c,v}}[\mathrm{TS}_2\|\mathrm{Subkey}\|\mathrm{Seq\#}]$
$\mathrm{Ticket}_v = \mathrm{E}(K_v, [\mathrm{Flags}\|K_{c,v}\|\mathrm{Realm}_c\|\mathrm{ID}_C\|\mathrm{AD}_C\|\mathrm{Times}])$
$Authenticator_c = \mathrm{E}(K_{c,v}, [ID_C\|Realm_c\|TS_2\|Subkey\|Seq\#])$

(c) Client/Server Authentication Exchange to obtain service

- **Times:** Used by the client to request the following time settings in the ticket:
  —from: the desired start time for the requested ticket
  —till: the requested expiration time for the requested ticket
  —rtime: requested renew-till time
- **Nonce:** A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information. The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options. These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the **ticket-granting service exchange** for versions 4 and 5. We see that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service. In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1). The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2), returning a ticket plus information needed by the client, the latter encrypted with the session key now shared by the client and the TGS.

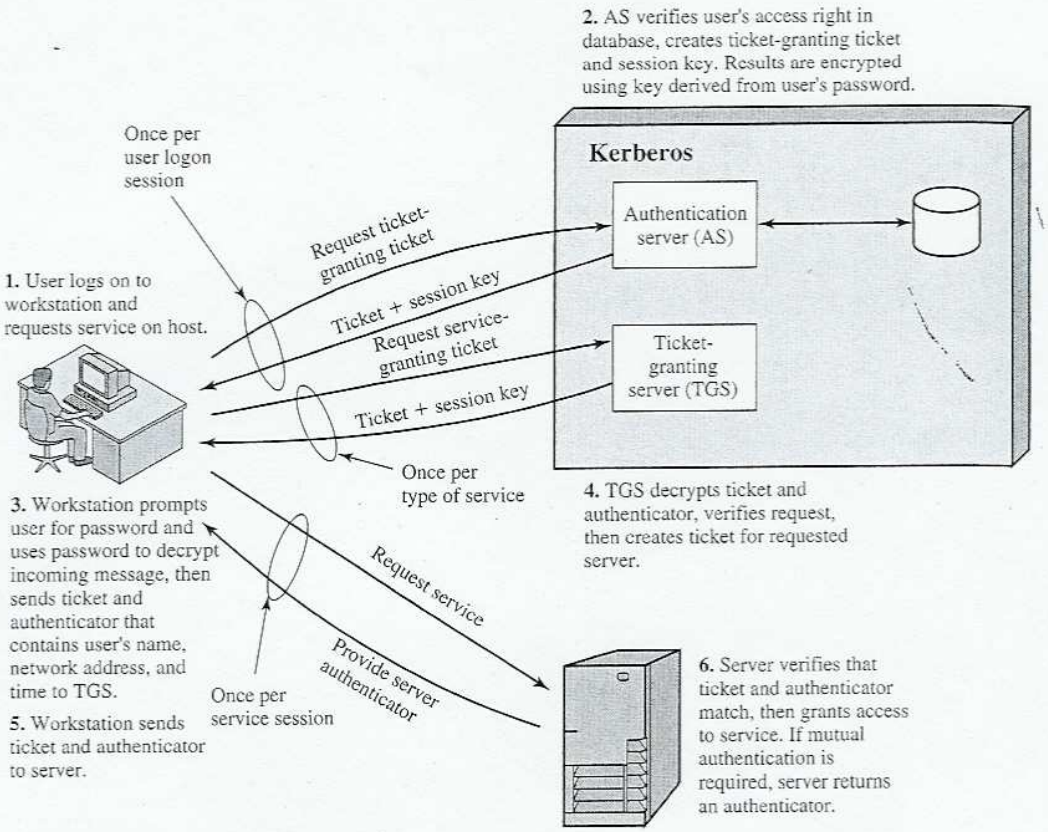


Figure 14.1 Overview of Kerberos

With these ground rules in place, we can describe the mechanism as follows (Figure 14.2): A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm). The client can then apply to the remote TGS for a service-granting ticket for the desired server in the realm of the remote TGS.

The details of the exchanges illustrated in Figure 14.2 are as follows (compare Table 14.1):

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

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

(3) C → TGS: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$

(5) C → TGS_{rem}: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(6) TGS_{rem} → C: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$

(7) C → V_{rem}: $Ticket_{vrem} \| Authenticator_c$

Table 14.1 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

inside the message encrypted with K_c, only the user's client can read it. The same session key is included in the ticket, which can be read only by the TGS. Thus, the session key has been securely delivered to both C and the TGS.

Note that several additional pieces of information have been added to this first phase of the dialogue. Message (1) includes a timestamp, so that the AS knows that the message is timely. Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the TGS and to learn its expiration time.

Armed with the ticket and the session key, C is ready to approach the TGS. As before, C sends the TGS a message that includes the ticket plus the ID of the requested service (message (3) in Table 14.1b). In addition, C transmits an authenticator, which includes the ID and address of C's user and a timestamp. Unlike the ticket, which is reusable, the authenticator is intended for use only once and has a very short lifetime. The TGS can decrypt the ticket with the key that it shares with the AS. This ticket indicates that user C has been provided with the session key $K_{c,tgs}$. In effect, the ticket says, "Anyone who uses $K_{c,tgs}$ must be C." The TGS uses the session key to decrypt the authenticator. The TGS can then check the name and address from the authenticator with that of the ticket and with the network address of the incoming message. If all match, then the TGS is assured that the sender of the ticket is indeed the ticket's real owner. In effect, the authenticator says, "At time TS_3, I hereby use $K_{c,tgs}$." Note that the ticket does not prove anyone's identity but is a way to distribute keys securely. It is the authenticator that proves the client's identity. Because the authenticator can be used only once and has a short lifetime, the threat of an opponent stealing both the ticket and the authenticator for presentation later is countered.

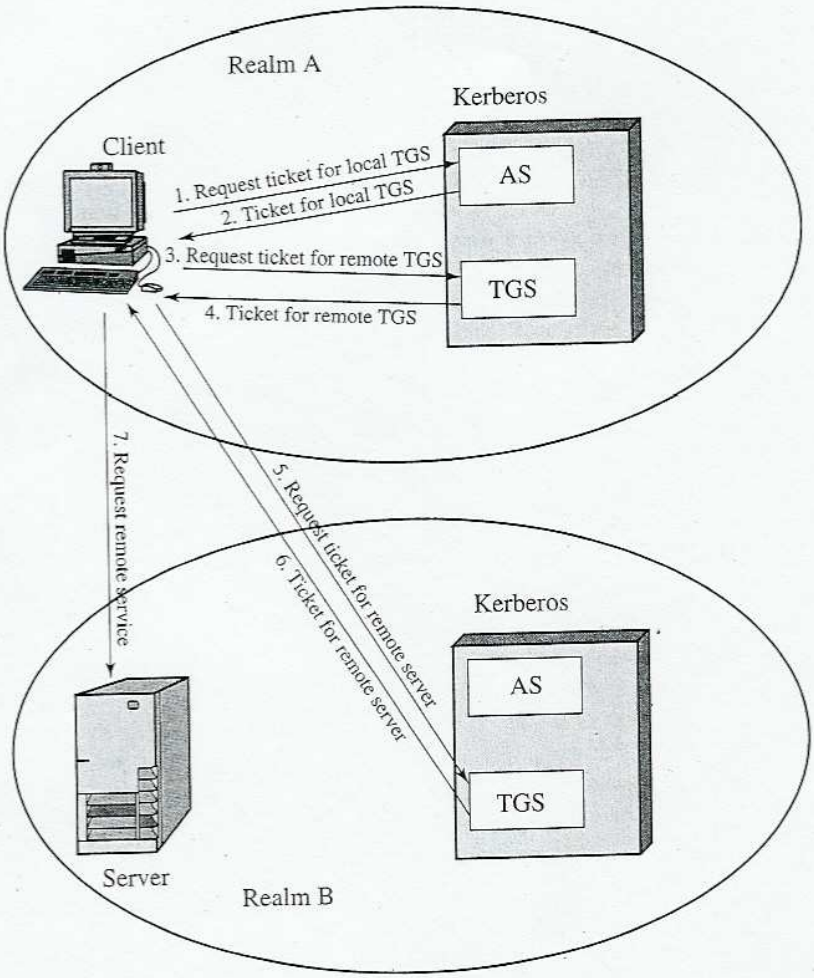


Figure 14.2 Request for Service in Another Realm

The ticket presented to the remote server (V_{rem}) indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request.

One problem presented by the foregoing approach is that it does not scale well to many realms. If there are N realms, then there must be $N(N-1)/2$ secure key exchanges so that each Kerberos realm can interoperate with all other Kerberos realms.

Kerberos Version 5

Kerberos Version 5 is specified in RFC 1510 and provides a number of improvements over version 4 [KOHL94]. To begin, we provide an overview of the changes from version 4 to version 5 and then look at the version 5 protocol.