Network Security

Asim Rasheed

Kerberos

Security Concerns

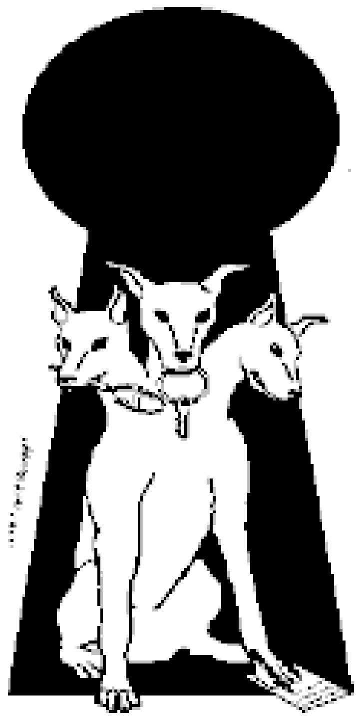
- key concerns are **confidentiality and timeliness**
- to provide confidentiality must encrypt identification and session key info
- which requires the use of previously shared private or public keys
- need timeliness to prevent **replay attacks**
- provided by using sequence numbers or timestamps or challenge/response

Simple Authentication

- To avoid impersonation, servers must be able to confirm identities
- Authentication Server (AS) can be used
  - Knows the password of all users
  - AS shares unique secret key with each server

KERBEROS

- In Greek mythology, a many headed dog, the guardian of the entrance of Hades



KERBEROS

- Users wish to access services on servers.
- Three threats exist:
  - User pretend to be another user.
  - User alter the network address of a workstation.
  - User eavesdrop on exchanges and use a replay attack.

Introduction

- Kerberos is a secret key based
- Provides authentication services
- Login session: Time between user logs in an logs out
- Kerberos consists of Key Distribution Service
  - Runs on a secure node
- User logs into the workstation by providing username and password
  - Used to obtain information from KDC that is useful to access remote resources
- Two versions: version 4 and 5

Kerberos Requirements

- Its first identified requirements as:
  - Secure
  - Reliable
  - Transparent
  - Scalable
- Implemented using an authentication protocol based on Needham-Schroeder

Kerberos v4 Overview

- A basic third-party authentication scheme
- Have an Authentication Server (AS)
  - Users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- Have a Ticket Granting server (TGS)
  - Users subsequently request access to other services from TGS on basis of users TGT

Tickets and Ticket-Granting Tickets

- Kerberos Server shares a secret key with each user
  - Known as Master key
- Kerberos Server invents a session key KAB
  - When a user A informs Kerberos Server it wants to talk to user B
  - KAB is encrypted using user A's master key
  - KAB is also encrypted with user B's master key and returns to user A
- Message consisting of KAB and some other information, encrypted with user B's master key is known as **Ticket**

Tickets and Ticket-Granting Tickets

- User B can decrypt the KAB and user A's name
- User B knows that anyone else who has KAB is acting on behalf of user A
- Master key is derived from user's password
- Session key SA is used by user A for a single session
  - Used to ask for tickets to resources
  - Only valid for a small time

Session Key

- Workstation on behalf of user A asks the Authentication Server (AS) for a session key SA
- SA is transmitted encrypted with user A’s master key
- AS also sends Ticket-Granting Ticket encrypted with Kerberos Server ’s master key
  - SA, user A’s name and TGT expiration time
- Workstation decrypts SA and only remember SA and TGT
- TGT is sent to KDC to acquire the session key for communicating between two parties

Session Key

- Ticket-Granting Ticket server is used for TGT
- TGT server and AS are collocated
  - Since both need to use the same information

Configuration

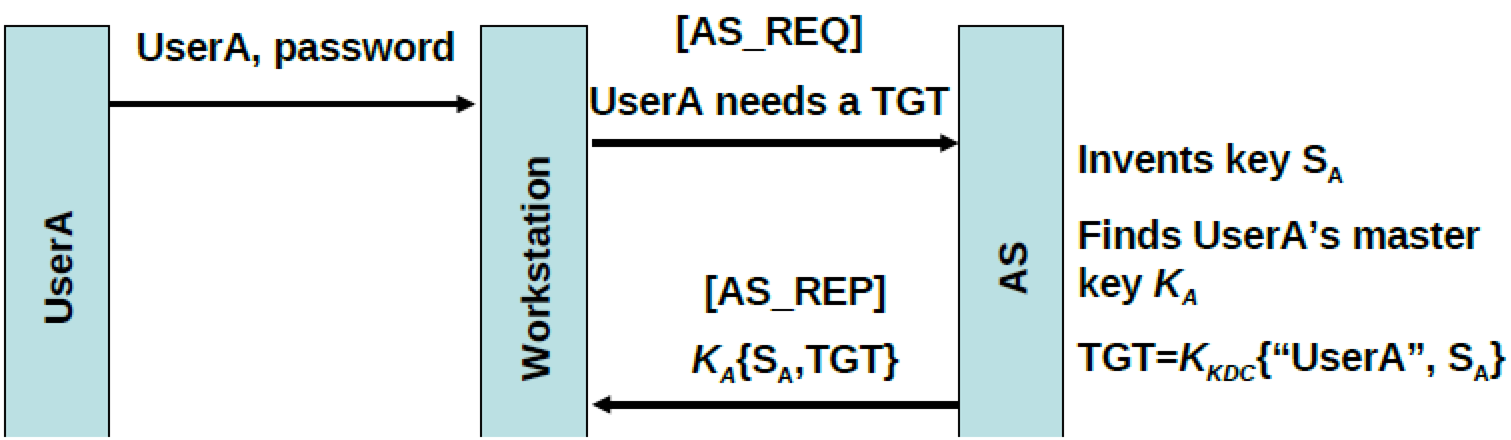
- Each principal has its own secret key called the master key
- Kerberos server
  - Authentication Server (AS)
  - Ticket Granting Server (TGS)
- AS keeps the master keys encrypted using its master key
- Kerberos uses DES

Obtaining a Session Key and TGT

- User gives the account name
- AS returns credentials:
  - Session key
  - Ticket Granting Ticket, which contains session key, user's name and an expiration time
- Information is doubly encrypted
- Workstation converts password into DES key and decrypts the information
- Once getting the key the master key is discarded
- Only retains the TGT and session key

Obtaining TGT

- Prompts for password after the reception of credentials



What is TGT for?

- When User A needs to access a remote resource
  - TGT is sent by the workstation to TGS
  - Workstation requests for Ticket
- TGS operates without having any volatile data
  - Has static database
- For each request just sends the response and forgets it

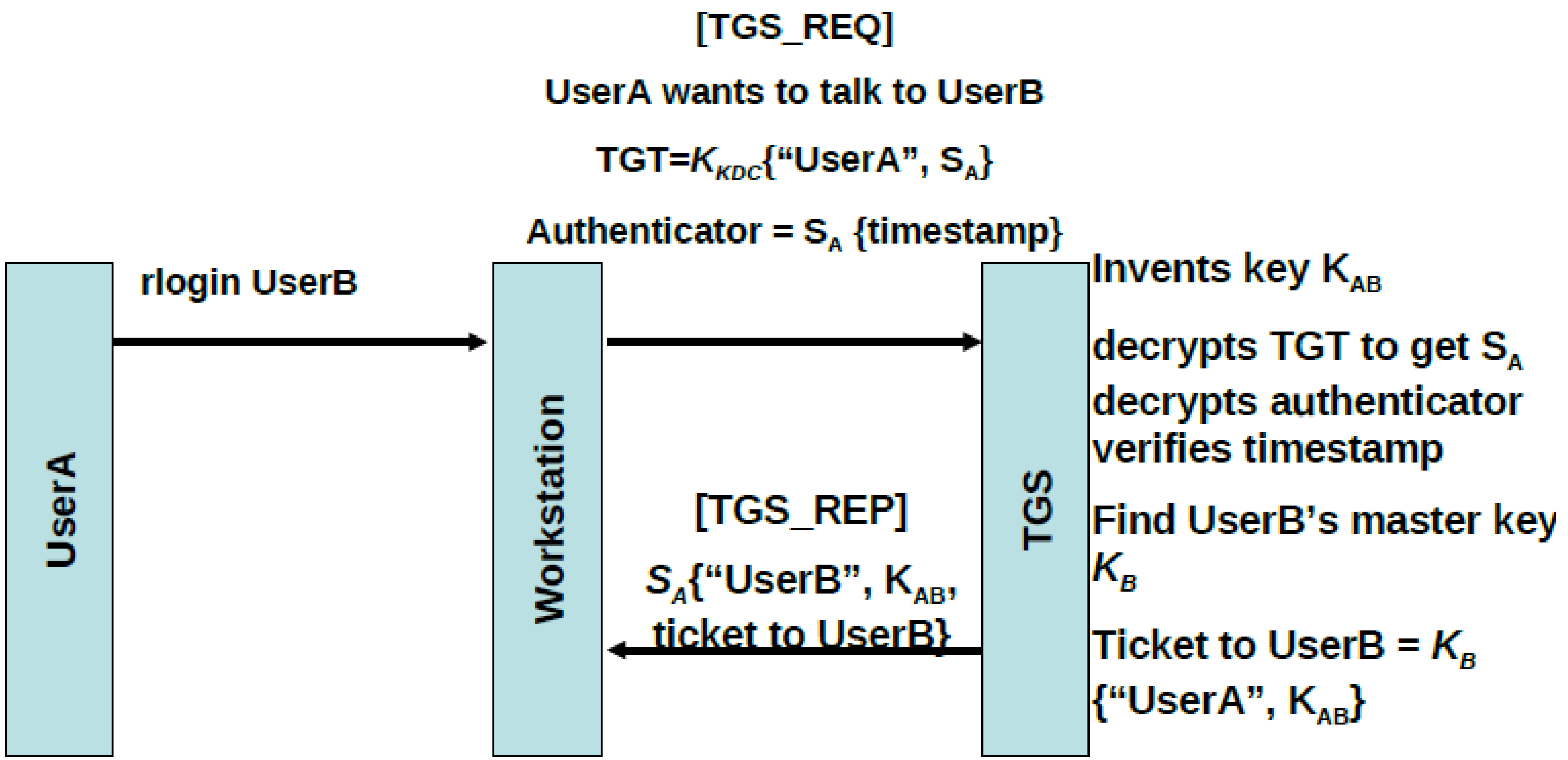
Talking to Remote Node

- Workstation sends request to TGS
  - TGT
  - Name of the remote resource
  - Authenticator: contains SA and time of day encrypted
- Reply contains
  - Ticket to remote resource
  - KAB: session key to be shared, encrypted with SA
- Because of authenticator, resources need to be synchronized

Acquiring the shared key

- TGS decrypts the TGT
- Checks expiration time in TGT if valid generates KAB
- Ticket is created which contains KAB, name of User A and expiration time, encrypted with UserB’s master key KB
- TGS returns the ticket, along with name and KAB
- All this encrypted using SA

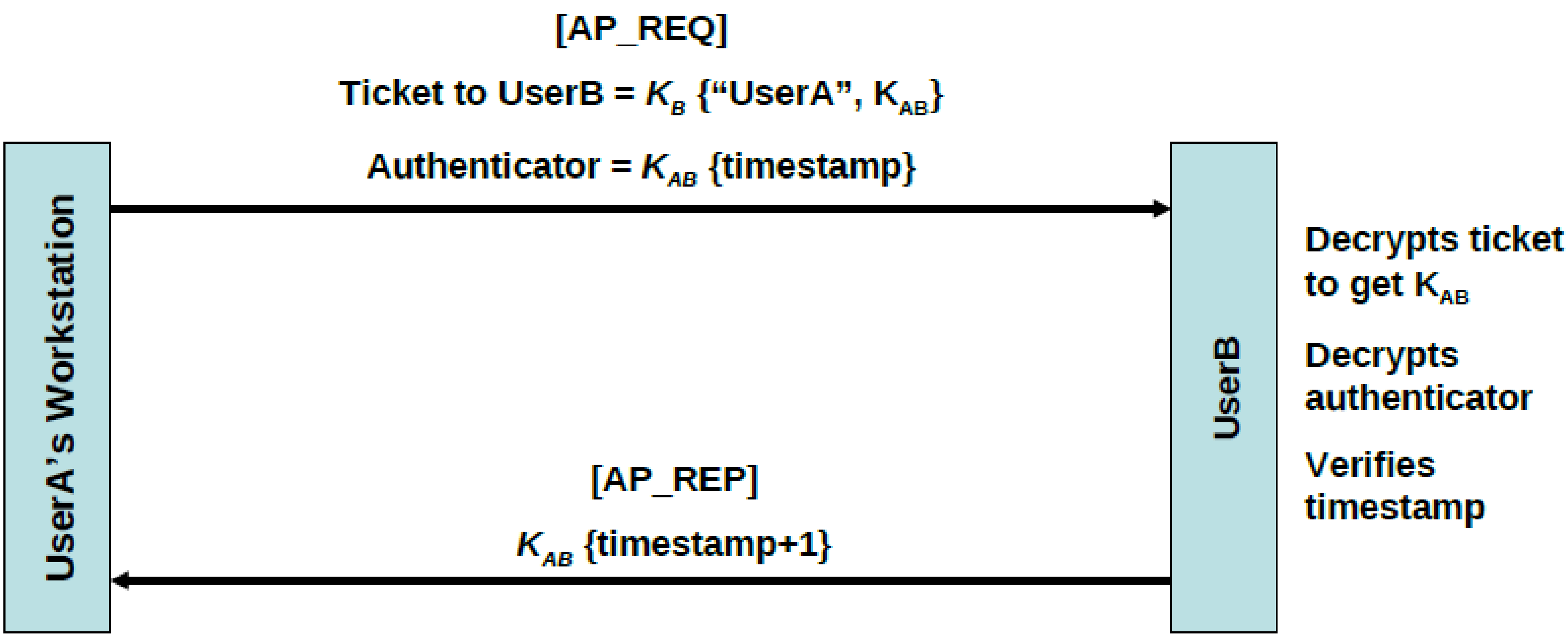
Getting a Ticket



Communicating With Remote Resource

- User A sends request to User B
  - Ticket
  - Authenticator: time is encrypted with KAB
- User B decrypts the ticket and gets the KAB and name User A
- User B assumes anyone who knows KAB is acting on behalf of User A

Logging into User B



- User B keeps track of the recent timestamps
  - Avoids replay attacks

Kerberos Version 4

- Terms:
  - C = Client
  - AS = authentication server
  - V = server
  - IDc = identifier of user on C
  - IDv = identifier of V
  - Pc = password of user on C
  - ADc = network address of C
  - Kv = secret encryption key shared by AS an V
  - TS = timestamp
  - || = concatenation

Simple Authentication Scenario

- C -> AS: *IDC* || *PC* || *IDV*
- AS -> C: Ticket
- C -> V: IDC || Ticket

Ticket = EKV [IDC || ADC || IDV]

- With the Ticket, C can apply to server for service
- Ticket is valid only for the originator of the request due to the presence of Address of C
- **The threat is that an opponent will steal the ticket and use it before it expires**

Secure Authentication Dialogue

- Problem:
  - Lifetime associated with the ticket-granting ticket
- If too short -> repeatedly asked for password
- If too long -> greater opportunity to replay
- To overcome major problems:
  - Minimize password entry
  - Plaintext transmission of password
- Introduce Ticket Granting Server (TGS)

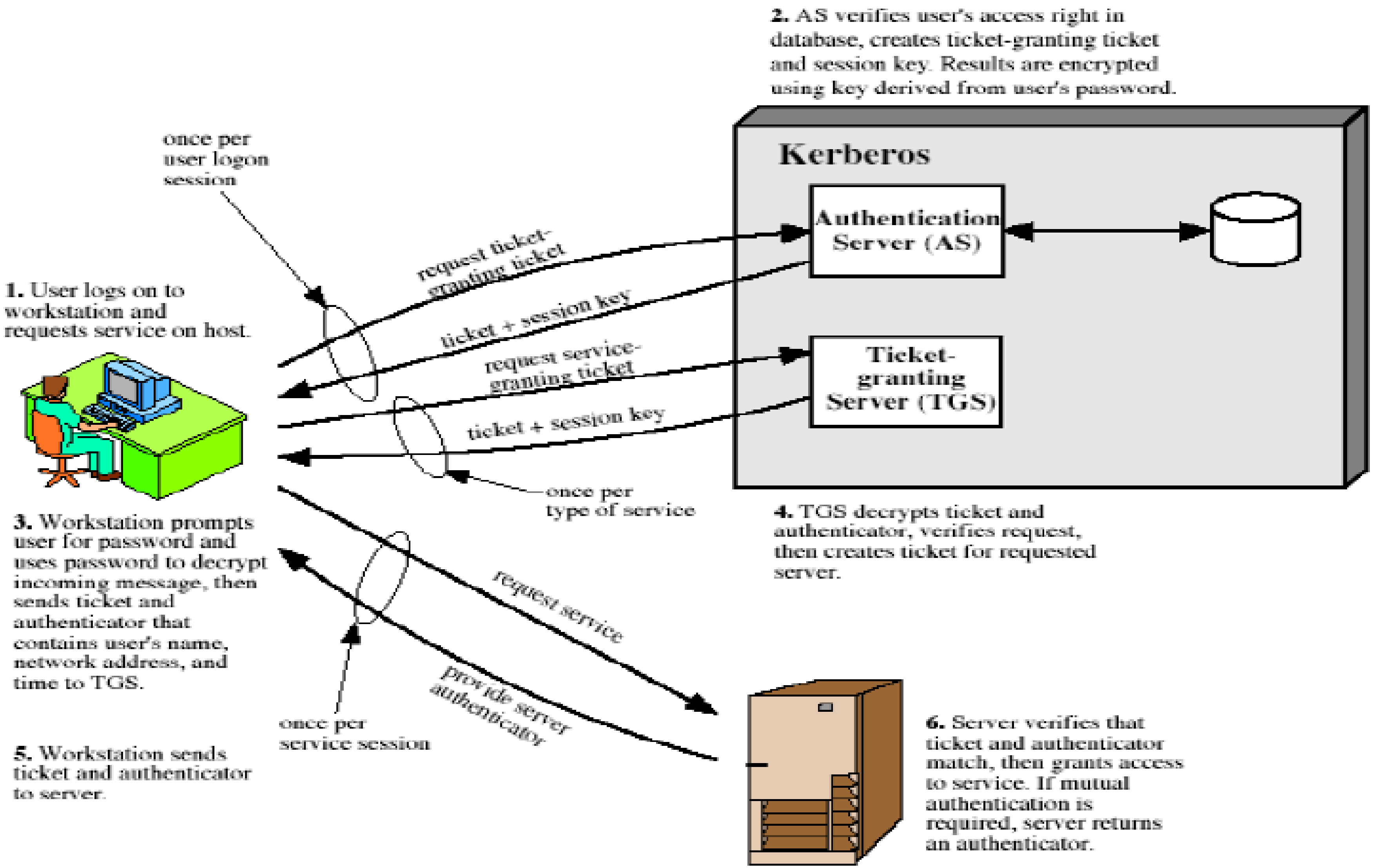
Secure Authentication Dialogue

- Once per user logon session:
  - C-> AS : IDC || ID_{tgs}
  - AS-> C: EKC [$Ticket_{tgs}$]
- Once per type of service
  - C->TGS: IDC||IDV||$Ticket_{tgs}$
  - TGS->C: $Ticket_{v}$
- Once per service session
  - C->V: IDC||Ticketv
- $Ticket_{tgs} = E_{ktgs}$ [IDC||ADC||IDtgs||TS1||Lifetime1]
- $Ticket_{V}$ = EKV [IDC||ADC||IDV||TS2||Lifetime2]

Secure Authentication

- TGS issues tickets to users who have been authenticated to AS
- Only the correct user with the password can acquire ticket

Kerberos 4 Overview



Replicated Kerberos Servers

- Single point of failure in case of single Kerberos Server
- Need to have multiple Kerberos Servers
- Share same Master Key and identical databases
- One Kerberos Server maintains the master copy
  - Every update must be made in it
- Other sites download the database periodically

Replicated Kerberos Servers

- Read only operations are used in authentication,
  - Hence can work even if master Kerberos Server is down
- Replication solves the problem of causing bottle neck at one server
- Principal’s master keys are stored in encrypted form
  - No threat of keys going to an intruder
- Threat of changing the data is there
  - Could be removed by transmitting a hash of it

Kerberos Realms

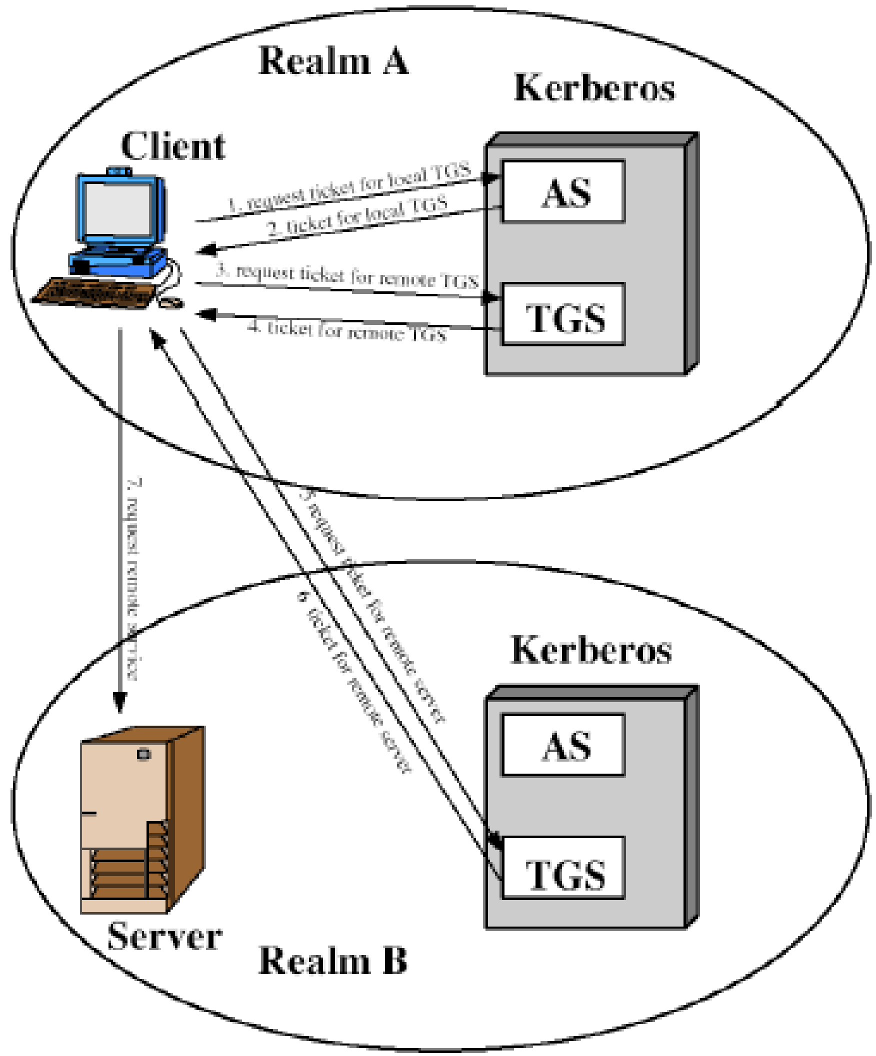
- A Kerberos environment consists of:
  - A Kerberos server
  - A number of clients, all registered with server
- This is termed as realm
  - Typically a single administrative domain
- If have multiple realms, their Kerberos servers must share keys and trust

Realms

- Every realm has its own Kerberos Server
- Two Kerberos Servers in different realms have
  - Different master keys
  - Different databases

Inter Realm Communication

- Client requests ticket of remote TGS from local TGS
- Then requests for a ticket for the service in the remote realm
- After getting ticket gets the service



Key Version Numbers

- User can be easily able to change his password
- Changing the password would change the master key
  - needs to be updated
- Changing the master key can create problems
- TGT with a session key that was obtained using the older master key would not stay valid

Key Version Numbers

- Each key is given a version number
- Keys are stored with the version number
- Different versions of the key are remembered by the resources
- For sometime password change may not be able to propagate completely

Kerberos V4 Message Exchange Authentication Service Exchange

- **C -> AS:** $ID_c \,||\, ID_{tgs} \,||\, TS_1$
- **AS -> C:** $E_{Kc}\,[K_{c,tgs} \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2 \,||\, Ticket_{tgs}]$

$$Ticket_{tgs} = E_{Ktgs}\; K_{c,tgs} \,||\, ID_C \,||\, AD_C \,||\, ID_{tgs} \,||\, TS_2 \,||\, Lifetime_2\;]$$

Ticket Granting Service Exchange

- **C -> TGS:** $ID_v \,||\, Ticket_{tgs} \,||\, Authenticator_c$
- **TGS -> C:** $E_{Kc,tgs}\,[K_{c,v} \,||\, ID_v \,||\, TS_4 \,||\, Ticket_v\,]$

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

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

$$Authenticator_c = E_{Kc,tgs}\,[ID_C \,||\, AD_C \,||\, TS_3]$$

Client Server Authentication Exchange

- **C -> V:** $Ticket_v \,||\, Authenticator_c$
- **V -> C:** $E_{Kc,v}[TS_5 + 1]$ (for mutual authentication)

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

$$Authenticator_c = E_{Kc,v}\,[ID_C \,||\, AD_C \,||\, TS_5]$$

Kerberos Version 5

- Developed in mid 1990's
- Provides improvements over v4
  - Addresses environmental shortcomings
    - Encryption algo, network protocol, byte order, ticket lifetime, authentication forwarding, inter-realm auth
  - And technical deficiencies
    - Double encryption, non-std mode of use, session keys, password attacks

Difference Between Version 4 & 5

- Encryption system dépendance (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Inter-realm authentication

Kerberos V5

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) **C → AS:** Options $\|$ $ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$
(2) **AS → C:** $Realm_c \| ID_C \| Ticket_{tgs} \| E_{K_c}\left[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) **C → TGS:** Options $\|$ $ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$
(4) **TGS → C:** $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}\left[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| Realm_c \| TS_1\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) **C → V:** Options $\|$ $Ticket_v \| Authenticator_c$
(6) **V → C:** $E_{K_{c,v}}$ [$TS_2 \|$ Subkey $\|$ Seq#]
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,v}}\left[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#\right]$

V5 Ticket Flags

- Initial: This ticket was issued using the AS protocol and not issued based on a ticket-granting ticket.
- Pre-authent: During initial authentication, the client was authenticated by the KDC before a ticket was issued.
- Hw-authent: The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.
- Renewable: Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date

Ticket Flags

- May-postdate: Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.
  - Postdated: Indicates that this ticket has been postdated; the end server can check the auth time field to see when the original authentication occurred.
  - Invalid: This ticket is invalid and must be validated by the KDC before use.
  - Proxy-able: Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket

Ticket Flags

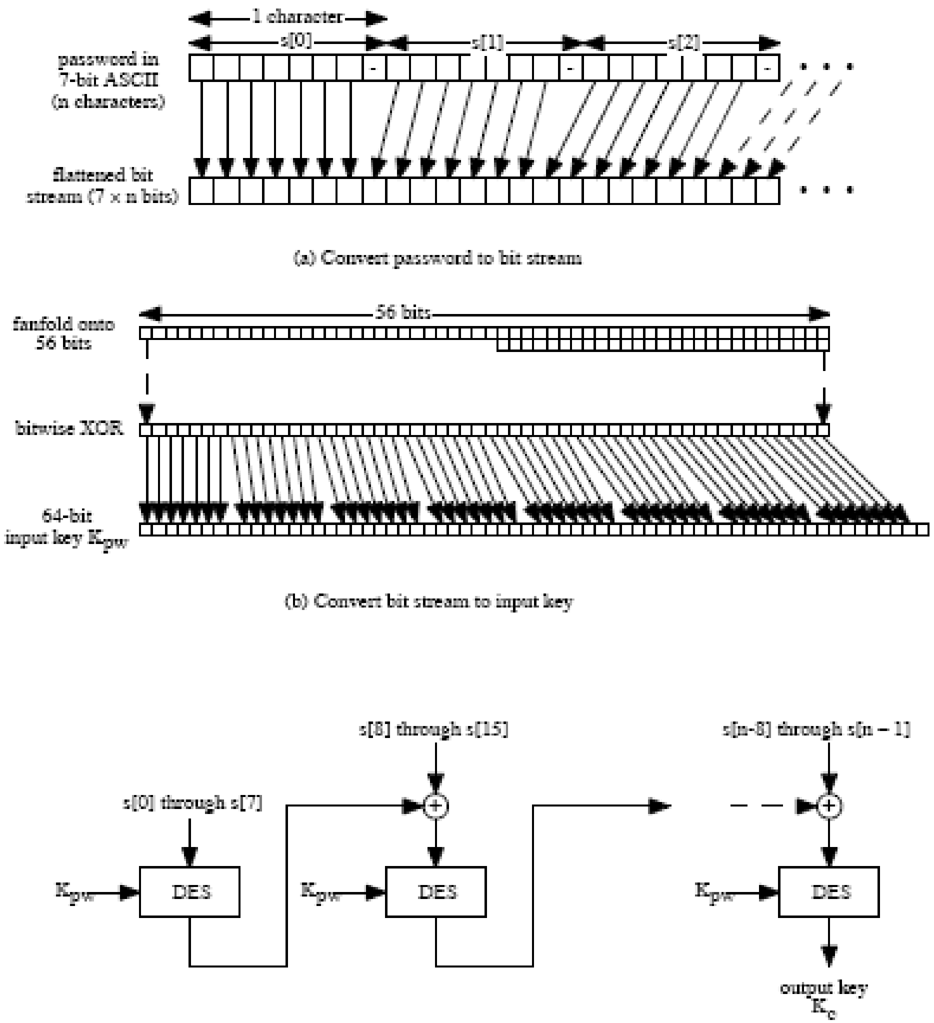
- Proxy: Indicates that this ticket is a proxy.
- Forwardable: Tells TGS that a new ticket-granting ticket with a different network address may be issued based on this ticket-granting ticket.
- Forwarded: Indicates that this ticket has either been forwarded or was issued based on authentication involving a forwarded ticket-granting ticket

Kerberos Encryption Techniques

- Password-to-Key Transformation
  - In kerberos, password are limited to the use of the characters that can be represented in a 7-bit ASCII format.
  - In the first step, character string, s, is packed in bit String, b, such that the first character is stored in the first 7 bits, the second in the second 7 bits, and so on.

$$
\begin{aligned}
b[0] &= \text{bit } 0 \text{ of } s[0] \\
&\cdots \\
b[6] &= \text{bit } 6 \text{ of } s[0] \\
b[7] &= \text{bit } 0 \text{ of } s[1] \\
&\cdots \\
b[7i + m] &= \text{bit } m \text{ of } s[i] \qquad 0 \le m \le 6
\end{aligned}
$$

Generation of Encryption Key from Password



(a) Convert password to bit stream

(b) Convert bit stream to input key

(c) Generate DES CBC checksum of password

Password-to-Key Transformation

- In the next step, string is compacted to 56 bits by aligning the bits in "fanfold" fashion and performing a bitwise XOR.
- For example if bit string length is 59

$$b[55] = b[55] \oplus b[56]$$

$$b[54] = b[54] \oplus b[57]$$

$$b[53] = b[53] \oplus b[58]$$

- Creates a 56-bit DES key.
- 7-bits are mapped onto 8-bits to form an input key K_{pw}
- Original password is encrypted using CBC mode of DES with Key Kpw

Any question ?