Kerberos
User Authentication

COEP TECHNOLOGICAL UNIVERSITY

Jibi Abraham **Shivajinagar, Pune-411 005**

(A Unitary Technological University of Govt. of Maharashtra)

Authentication

- Authentication verifies the identity of a user or service
- Two Types
  - Data (message) authentication
    - Dealt with Hash, MAC, Digital Signature
  - User Authentication
    - Like Kerberos, Single Sign-On (SSO), Identity Management

Need of User Authentication

- User at local machine wishes to access services running on servers in a distributed network
- Local machine cannot be trusted to identify users correctly
- Threats:
  - Impersonation: User may gain access to local machine and pretend to be another user
  - User may alter IP address of local machine so that requests sent from altered machine appear to come from impersonated machine
  - User may eavesdrop on exchanges and use replay attack to gain access to the server or disrupt operations

User Authentication

- Fundamental security building block
  - basis of access control and user accountability
- Is the process of verifying an identity claimed by or for a system entity
- Has two steps:
  - identification - specify identifier
  - verification - bind entity (person) and identifier
- Different from message authentication

User Identification

- Describes a method by which a User claims to have a specific identity or to validate a user's claims of who it claims it to be
- It is the critical first step in applying access control
- Can be provided by the use of username or account number etc
- Authentication involves two step process; entering the public information (identification) and then entering the private information
- It establishes trust between the user and the system for the allocation of privileges

Methods of User Authentication

- There are four means of authenticating a user's identity, which can be used alone or in combination:
  - **Something the individual knows:** password, PIN, or answers to a prearranged set of questions
  - **Something the individual possesses (*token)*:** electronic keycards, smart cards, and physical keys.
  - **Something the individual is (static biometrics):** fingerprint, retina, and face.
  - **Something the individual does (dynamic biometrics):** recognition by voice pattern, handwriting characteristics, and typing rhythm
- All have issues

Authentication Protocols

- Authentication Protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys
- May be used one-way or mutual
- Key issues are
  - Confidentiality – to protect session keys (to prevent masquerading and to prevent compromise of session keys)
  - Timeliness – to prevent replay attacks

Replay Attacks

- Where a valid signed message is copied and later resent
- Allow an opponent to compromise a session key or successfully impersonate another party
  - Simple replay
  - Repetition that can be logged
  - Repetition that cannot be detected
  - Backward replay without modification
- Countermeasures include
  - Use of sequence numbers (generally impractical)
  - Timestamps (needs synchronized clocks)
  - Challenge/response (using unique nonce)

One-Way Authentication

- Required when sender and receiver are not in communications at same time (eg. email)
- Needs the email header in clear so that the mail can be delivered by email system
- May want contents of body protected using encryption
- May want the sender to be authenticated

One-Way Authentication

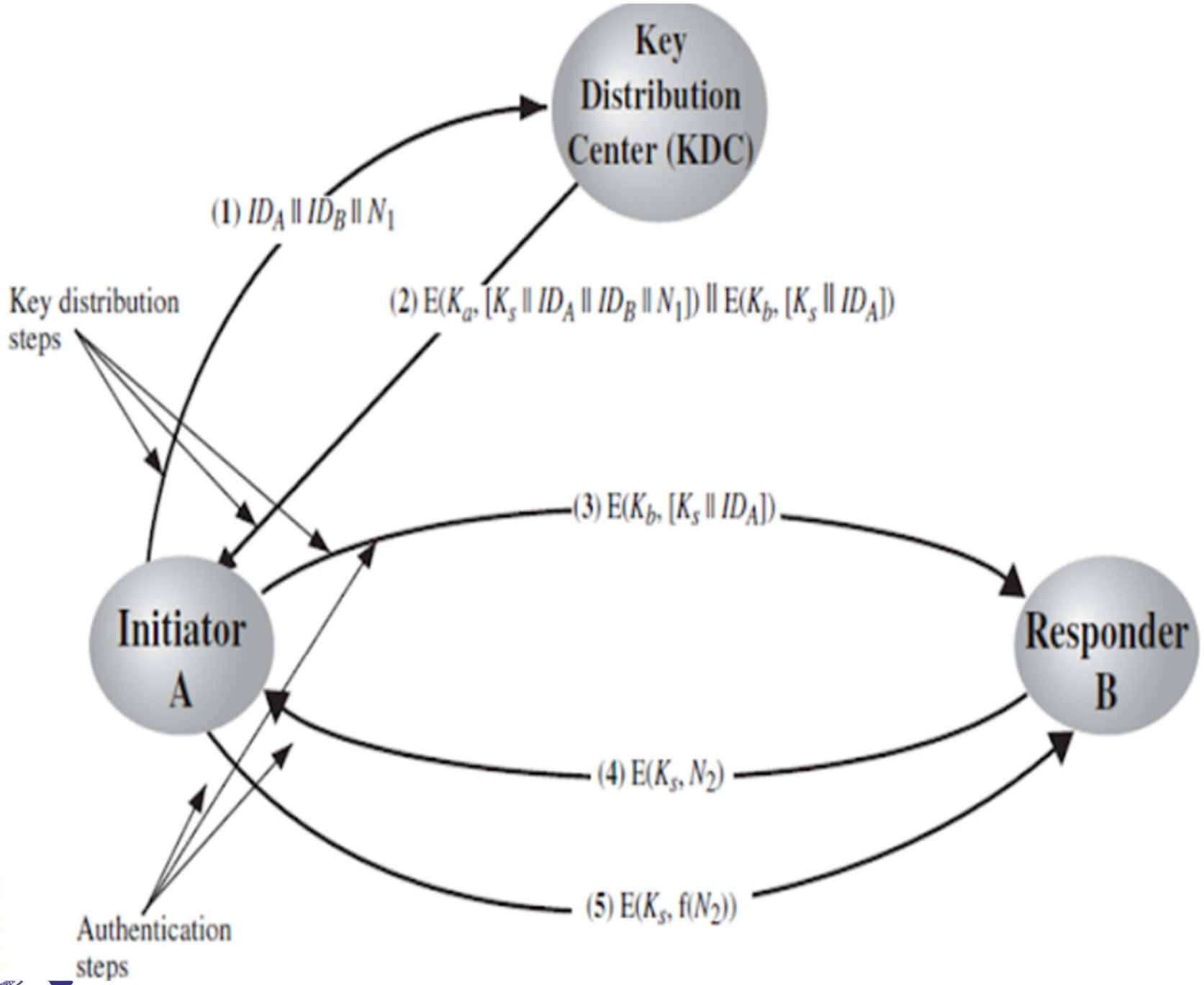
- Have public-key approaches for email
  - Encryption of messages for confidentiality, authentication, or both
  - Must know public keys of users
  - Using costly public-key algorithm on long message
- For confidentiality, encrypt the message with one-time secret key, and the secret key is public key encrypted
- For authentication use a digital signature
  - May need to protect by encrypting signature
- Use digital certificate to supply public key

Needham-Schroeder Protocol

- Use Symmetric Encryption
- Usually with a trusted Key Distribution Center (KDC)
  - Each party shares its own master key with KDC
  - KDC generates session key K_s used between parties
  - Master keys are used to distribute these to them

1. A->KDC: $ID_A \,||\, ID_B \,||\, N_1$
2. KDC -> A: $E(K_a,[K_s||ID_B||N_1||\ E(K_b,[K_s||ID_A])])$
3. A -> B: $E(K_b, [K_s||ID_A])$
4. B -> A: $E(K_s,\ [N_2])$
5. A -> B: $E(K_s, [f(N_2)])$

Needham-Schroeder Protocol



Needham-Schroeder Protocol

1. A->KDC: $ID_A \,||\, ID_B \,||\, N_1$
2. KDC -> A: $E(K_a,[K_s \,||\, ID_B \,||\, N_1 \,||\, E(K_b,[K_s \,||\, ID_A])])$
3. A -> B: $E(K_b, [K_s \,||\, ID_A])$
4. B -> A: $E(K_s, [N_2])$
5. A -> B: $E(K_s, [f(N_2)])$

- Is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent, convincing B that itis communicating with A

Denning 81- Authentication Protocol

- Timestamps in steps 2 & 3 ensure that the key distribution is a fresh exchange

$$A \rightarrow KDC: ID_A||ID_B$$

$$KDC \rightarrow A: E\,(K_a, [K_s||ID_B||T||E\,(K_b, [K_s||ID_A||T])])$$

$$A \rightarrow B: E\,(K_b, [K_s||ID_A||T])$$

$$B \rightarrow A: E\,(K_s, N_1)$$

$$A \rightarrow B: E\,(K_s, f(N_1))$$

Neuman 93 – Authentication Protocol

- using an extra nonce

$$A \rightarrow B: ID_A \,||\, N_a$$

$$B \rightarrow KDC: ID_B \,||\, N_b \,||\, E(K_b, [ID_A \,||\, N_a \,||\, T_b])$$

$$KDC \rightarrow A: E(K_a, [ID_B \,||\, N_a \,||\, K_s \,||\, T_b]) \,||\, E(K_b, [ID_A \,||\, K_s \,||\, T_b]) || N_b)$$

$$A \rightarrow B: E(K_b, [ID_A \,||\, K_s \,||\, T_b]) \,||\, E(K_s, N_b)$$

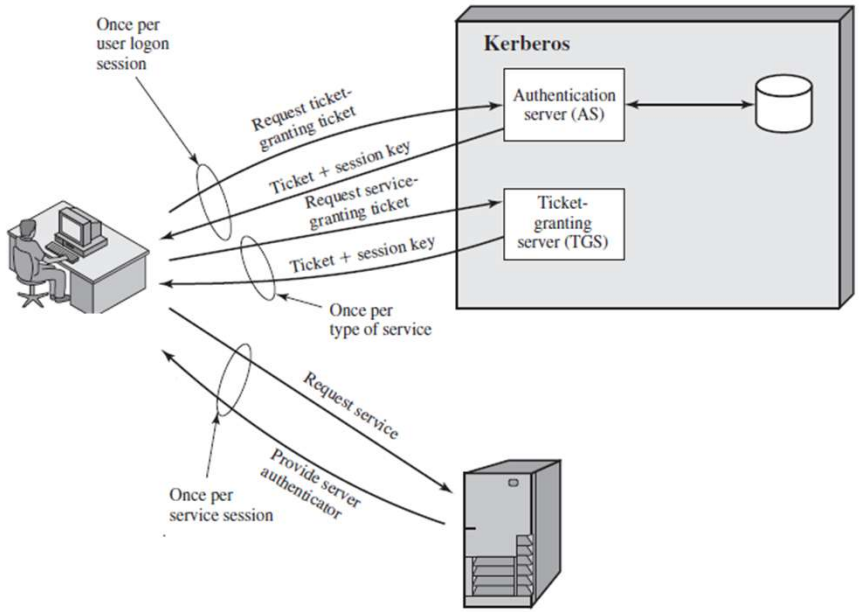
Kerberos -Authentication

- Trusted key server system from MIT
- Implemented using an authentication protocol based on Needham-Schroeder
- Kerberos is a de-facto authentication standard for heterogeneous networks and used in distributed environments
- It works on a client/server model using symmetric key algorithm
  - allows users access to services distributed throughout the network
  - without needing to trust all workstations
  - rather all trust a central authentication server

Kerberos Requirements

- Two versions in use: 4 and 5
- It provides end-to-end security
- First published report identified its requirements as:
  - Security
  - Reliability
  - Transparency (not to delay the process)
  - Scalability
- Most Kerberos authentications work with shared secret keys and eliminates the need to share the passwords over the network

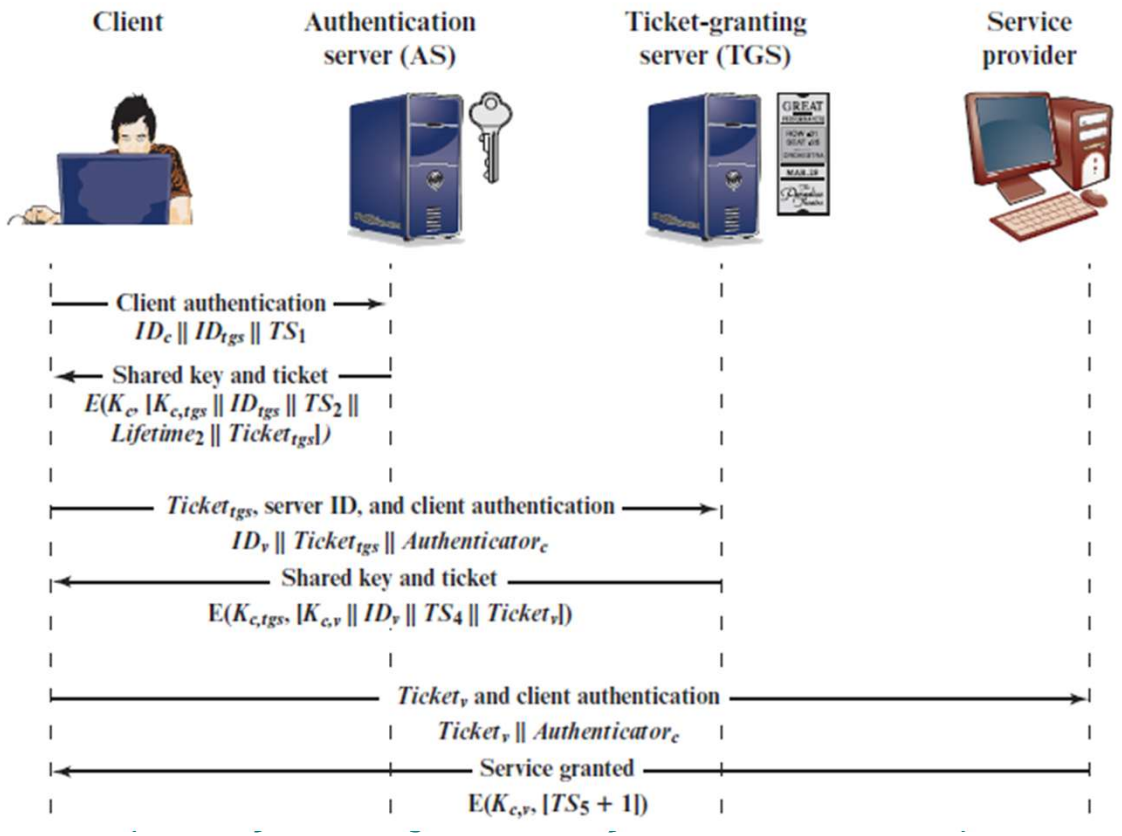
Kerberos Realms



- Realm consists of:
  - a Kerberos server
  - several clients, all registered with the server
    - Kerberse server have user ID hashed password of all users
  - application servers, sharing keys with all server
- A *realm* is typically a single administrative domain

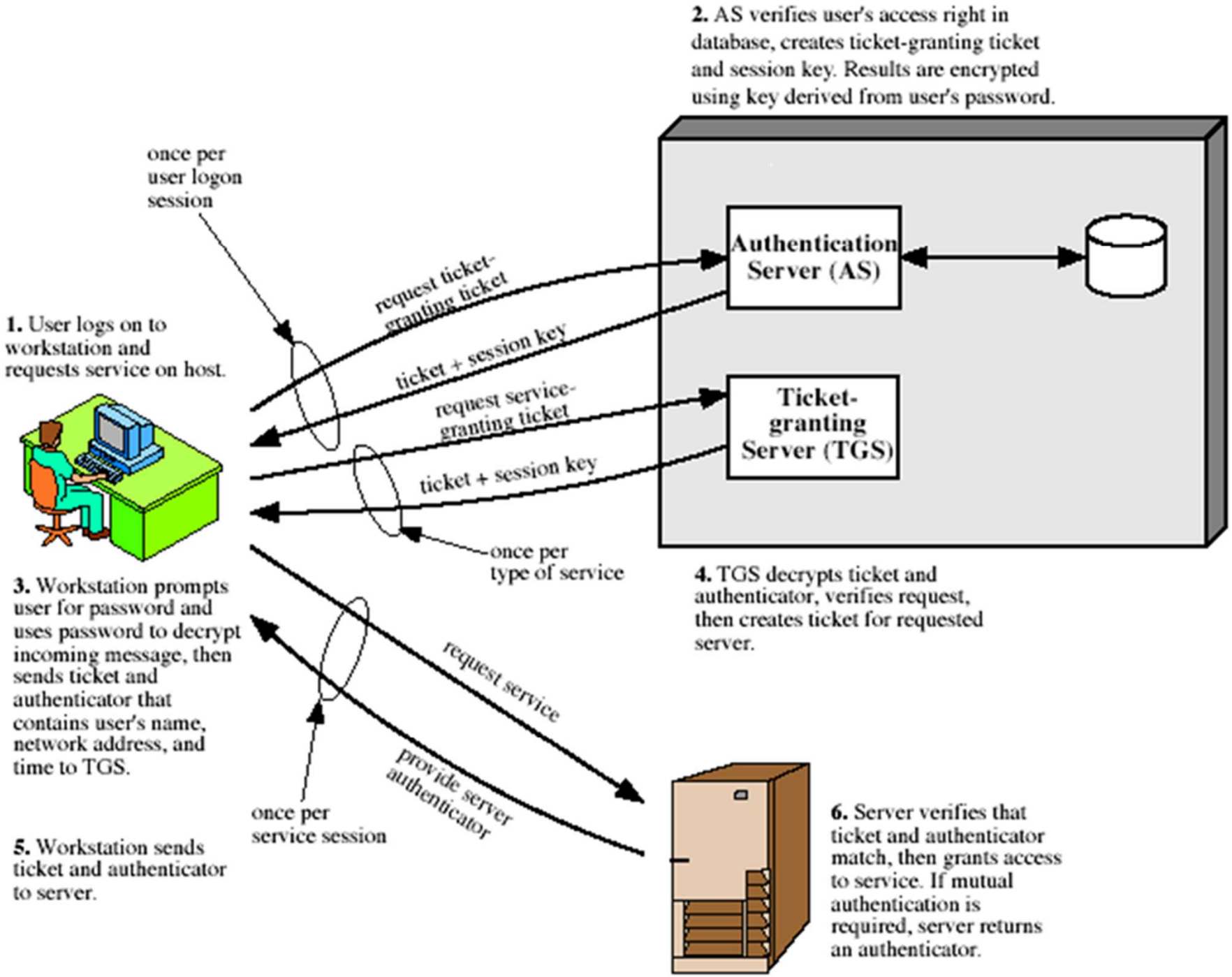
Kerberos Overview

- *Authentication Server (AS)*
  - Knows the passwords of all users
  - user initially negotiates with AS to identify themselves
  - AS provides a *ticket granting ticket TGT*
- *Ticket Granting server (TGS)*
  - user subsequently request access to other services from TGS on basis of user's TGT



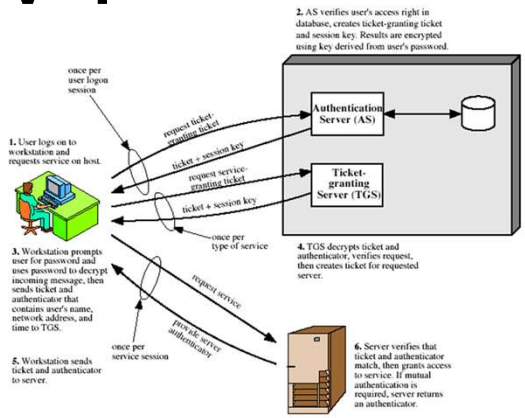
COEP Tech

User Authentication Architecture



Authentication Dialogue V4

- Introduced *Ticket Granting server (TGS)*
- Once per user logon session



Authentication Service Exhange: To obtain Ticket-Granting Ticket

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

(2) AS → C: $E_{Kc}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_1 \| Ticket_{tgs}]$

– $Ticket_{tgs} = E_{Ktgs}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_1]$

– ID_{TGS} = Identifier of the TGS

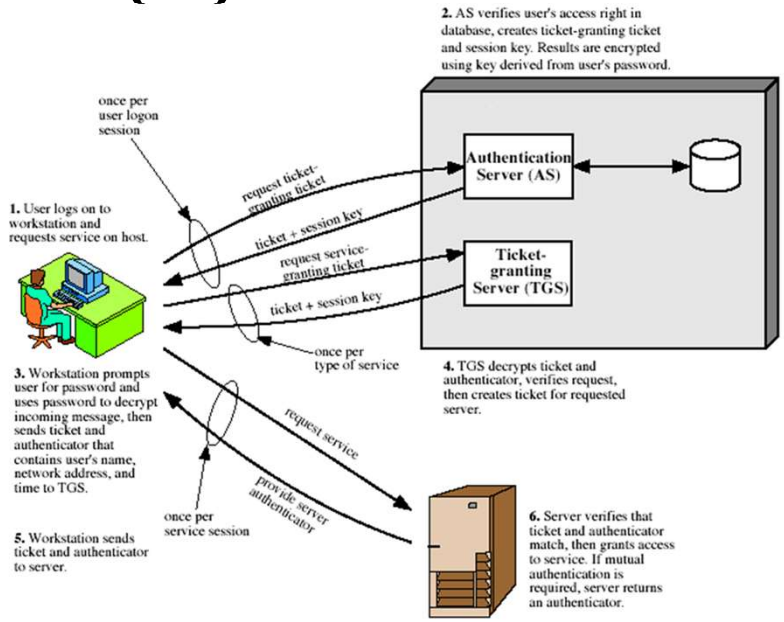
– $Ticket_{TGS}$ = *Ticket-granting ticket or TGT*

– TS_1 = timestamp

– $Lifetime_1$ = lifetime of the ticket

– K_C = key derived from user's password

Messages (3) and (4)



- Once per type of service

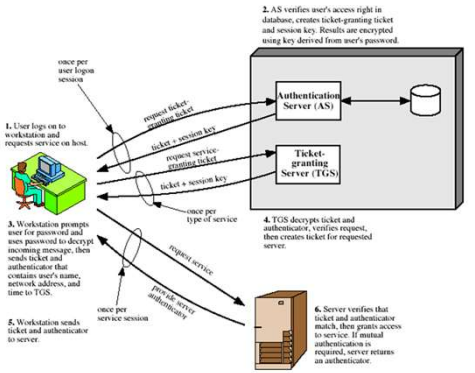
Ticket-Granting Service Echange: To obtain Service-Granting Ticket

(3) C → TGS: $ID_V \,||\, Ticket_{tgs} \,||\, Authenticator_c$

(4) TGS → C: $E_{Kc}[K_{c,v} || ID_V || TS_4 || Ticket_v]$

- $Ticket_v = E_{Kv}[K_{c,v} \,||\, ID_C \,||\, AD_C \,||\, ID_V \,||\, TS_4 \,||\, Lifetime_4]$
- $Authenticator_c = K_{c,tgs}[ID_C \,||\, AD_C \,||\, TS_3]$

Message 5



- Once per service session
- *C says to V "I am ID_C and have a ticket frc TGS". Let me in!*

Client/Server Authentication Exhange: To Obtain Service

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

(6) V → C: $EKc,v[TS5 + 1]$

- $Authenticator_c = K_{c,v}[IDC \| AD_C \| TS_5]$

Multiple Realms

- Network of clients and servers under different administrative organizations constitute multiple realms
- If have multiple realms, their Kerberos servers must share keys and trust
- If there are n realms, n(n-1)/2 secure key exchanges needed

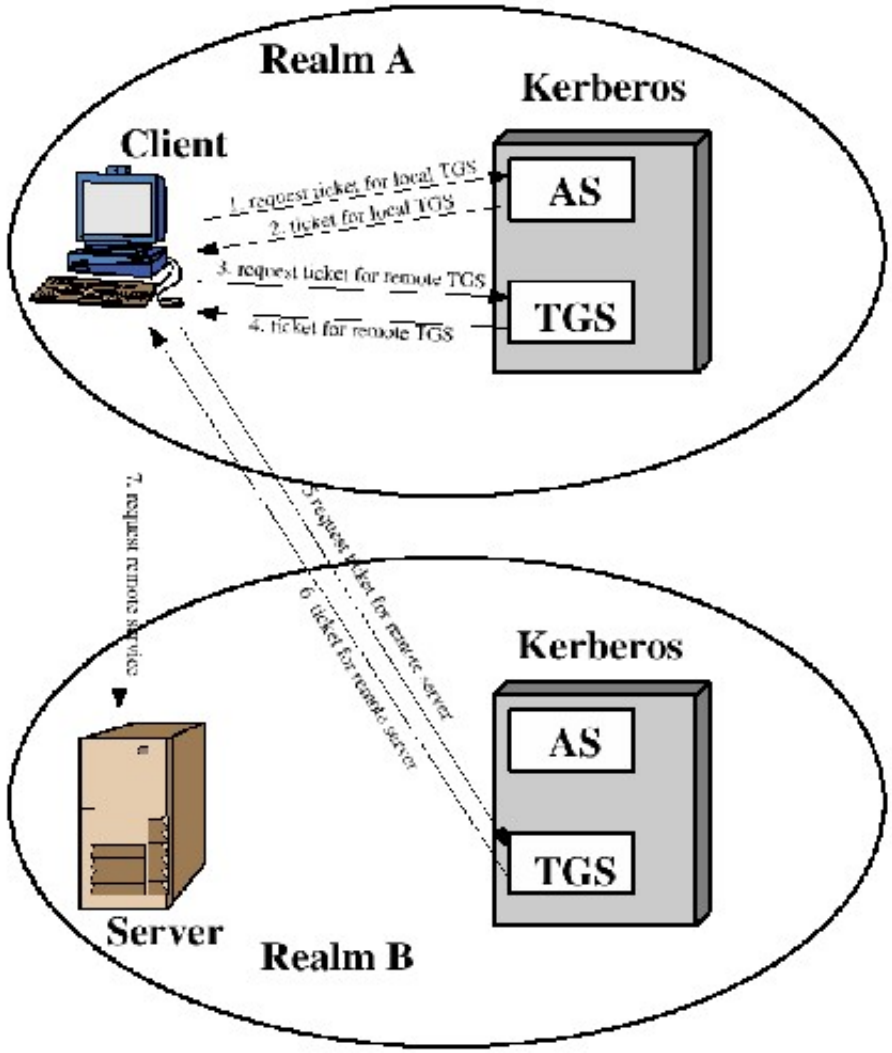


Figure 4.2 Request for Service in Another Realm

Kerberos Version 5

- Developed in mid 1990's
- Provides improvements over v4
  - addresses environmental shortcomings
    - encryption algorithm (only DES), network protocol (only IP), byte order (ASN not possible), ticket lifetime (max 1280 min, not sufficient), authentication forwarding (not allowed), inter-realm authentication
  - and technical deficiencies
    - double encryption, non-standard mode of use (PCBC only), session keys (possibility of replay in same key usage), password attacks
- V5 is specified as Internet standard RFC 1510

Kerberos V5

- Options: used to request that certain flags are set in the returned ticket
- Times: Used by client to request the ticket time settings such as
  - From: start time
  - Till: Expiation time
  - Rtime: requested renew till time
  - Nonce: a random value to be repeated to avoid replay attacks
  - Subkey: a session specific encryption key
  - Sequence no: used for a session to avoid replay attack
  - Flags – expanded functionality

Kerberos V5 Interaction

Kerberos V4 Environmental Shortcomings

- Which is the encryption algorithm used?
  - DES
- Does it support IPv6?
  - No. Only IPv4
- Does it support for cross-realm authentication?
  - No
- Does it use Abstract Syntax Notation One (ASN.1)?
  - No. It uses a simpler, proprietary "receiver-makes-right" encoding system
- What is the ticket lifetime used?
  - max 1280 min, not sufficient
- Does authentication forwarding allowed?
  - No

Kerberos V4 Technical Deficiencies

Authentication Service Exhange: To obtain Ticket-Granting Ticket

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

(2) AS → C: $E_{Kc}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_1 \| Ticket_{tgs}]$

– $Ticket_{tgs} = E_{Ktgs}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_1]$

- Does the double-encrypted $Ticket_{tgs}$ provide double Security strength?
  - No
- Does it use cyber mode operation
  - No. Uses non-standard chaining mode, Propagating Cipher Block Chaining (PCBC)
- , despite its use of timestamps for prevention
- session keys (possibility of replay in same key usage), password attacks

Kerberos V4 Technical Deficiencies

- Is there a known vulnerability to session key replay attacks?
  - **Attacker captures credentials:** Attacker uses a network sniffer to intercept a client's service ticket and authenticator

Client/Server Authentication Exhange: To Obtain Service

(5) C → V: $Ticket_v \| Authenticator_c$ $Authenticator_c = K_{c,v}[IDC \| AD_C \| TS_5]$

(6) V → C: $EKc,v[TS5 + 1]$

  - **Attacker initiates a DoS attack:** To prevent the legitimate client from accessing the service, attacker floods the network or perform a targeted DoS attack against the client

Kerberos V4 Session Key Replay Attack

- **Attacker spoofs the client's IP:** With the client offline, the attacker spoofs the client's IP address and replays the captured service ticket and authenticator to the server
- **Server accepts the replayed request:** Since Kerberos v4 does not enforce a robust check to ensure that the authenticator is fresh and hasn't been used before, the server accepts the replayed, but still-valid, credential. The server decrypts the authenticator using the session key from the service ticket and, finding the timestamp to be within the accepted time window, grants access to the attacker

Kerberose V5

- Which Internet standard RFC specify Kerberose V5?
  - 1510
- List 2 applications that use Kerberose V5
  - Microsoft's Windows Server
  - Active Directory implementations

Kerberos v5 Dialogue

(1) C → AS Options || ID_c || $Realm_c$ || ID_{tgs} || $Times$ || $Nonce_1$

(2) AS → C $Realm_c$ || ID_C || $Ticket_{tgs}$ || $E(K_c, [K_{c,tgs} || Times || Nonce_1 || Realm_{tgs} || ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags || K_{c,tgs} || Realm_c || ID_C || AD_C || Times])$

(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 || $Ticket_v$ || $E(K_{c,tgs}, [K_{c,v} || Times || Nonce_2 || Realm_v || ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags || K_{c,tgs} || Realm_c || ID_C || AD_C || Times])$

$Ticket_v = E(K_v, [Flags || K_{c,v} || Realm_c || ID_C || AD_C || Times])$

$Authenticator_c = E(K_{c,tgs}, [ID_C || Realm_c || TS_1])$

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

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

(6) V → C $E_{K_{c,v}}$ [TS_2 || Subkey || Seq#]

$Ticket_v = E(K_v, [Flags || K_{c,v} || Realm_c || ID_C || AD_C || Times])$

$Authenticator_c = E(K_{c,v}, [ID_C || Realm_c || TS_2 || Subkey || Seq\#])$

(c) Client/Server Authentication Exchange to obtain service

Common Kerberos v5 Options

- Broadly categorized into flags for managing tickets, options for specific client applications, and configuration settings that control the behavior of the protocol
- To control authentication behavior, manage access, and enhance security
- Ticket flags: define their properties and usage

Ticket Optionsx

- **Ticket Lifetimes**: explicit start, expiration, and renewal times to manage ticket validity
- **Renewal Time**: Allows to renew a ticket until a specified time without needing to re-authenticate with a password

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.
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 authtime field to see when the original authentication occurred.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
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.

Encryption and Authentication Options

- **Encryption Types**: supports DES, AES, and RC4
- **Pre-Authentication**: A mechanism to protect the password from attacks during the initial authentication exchange
- **Public Key and Anonymous Support**: advanced authentication methods like Public Key Cryptography for Initial Authentication (pkinit) and anonymity support

Common Kerberos v5 Options

- Network Configuration Options
  - **DNS Support**: can use DNS to find KDC hosts for a realm or to map domain names to realms
  - **Cross-Realm Authentication** for connecting Kerberos realms
  - **IP Address Support**: Supports both IPv4 and IPv6

Kerberos V4 vs V5

V4

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

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

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

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

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

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

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

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

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

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

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

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

V5

(1) C → AS Options $\| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS → C $Realm_c \| ID_C \| Ticket_{tgs} \| E(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

(3) C → TGS Options $\| ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $Realm_c \| ID_C \| Ticket_v \| E(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$

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

(6) V → C $E_{Kc,V}$ [$TS_2 \| Subkey \| Seq\#$]

$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Authenticator_c = E(K_{c,v}, [ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#])$

Kerberos v5 Improvements for Replay Protection

- **Stronger and authenticated encryption:** uses stronger encryption algorithms (like AES) and adds a sequence no, preventing attackers from easily forging or manipulating tickets.
- **Pre-authentication for password guessing:** step 2 requires to encrypt the current time with their secret key and send it to the KDC. This prevents an attacker from simply requesting a ticket for another user and then performing an offline password guessing attack on the encrypted session key.

Key Features of Kerberos V5

- **Authentication and Encryption**: provides mutual authentication between clients and servers and supports many encryption algorithms
- **Support for Timestamps and sequence number**: in tickets and authenticators, making it resistant to replay attacks
- **Cross-Realm Authentication**: supports authentication across different administrative domains (realms), enabling interoperability between organizations
- **Extensibility**: designed to be flexible and extensible, supporting multiple encryption algorithms and additional features like renewable and forwardable tickets.

Advantages of Kerberos V5

- **Scalability**: The use of multiple Ticket-Granting Servers (TGS) improves scalability and reduces the load on individual servers
- **Interoperability**: Kerberos V5 adheres to Internet Engineering Task Force (IETF) standards, ensuring compatibility with other systems

Kerberos V5 Practical Applications

- **Single Sign-On (SSO)**: enables users to authenticate once and access multiple services without re-entering credentials
- **Delegated Authentication**: allows services to act on behalf of users when accessing other services, useful in distributed applications
- **Secure Communication**: By encrypting data and verifying identities, it ensures secure communication in client-server architectures

Limitations and Considerations

- **Complex Configuration**: Setting up and managing Kerberos can be challenging, especially in large or heterogeneous environments
- **Time Synchronization**: Kerberos relies on synchronized clocks across the network to prevent replay attacks, requiring additional infrastructure
- **Migration Challenges**: Transitioning from older versions like Kerberos V4 to V5 can be complex in legacy systems