Software Security and Dependability

ENGR5560G

Lecture 09

User Authentication

Dr. Khalid A. Hafeez
Spring, 2025

User-Authentication

- The process of determining whether some user or some application who or what it declares itself to be.
- Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server
- User authentication is distinct from message authentication

Three Key Principles (NIST SP 800-63):

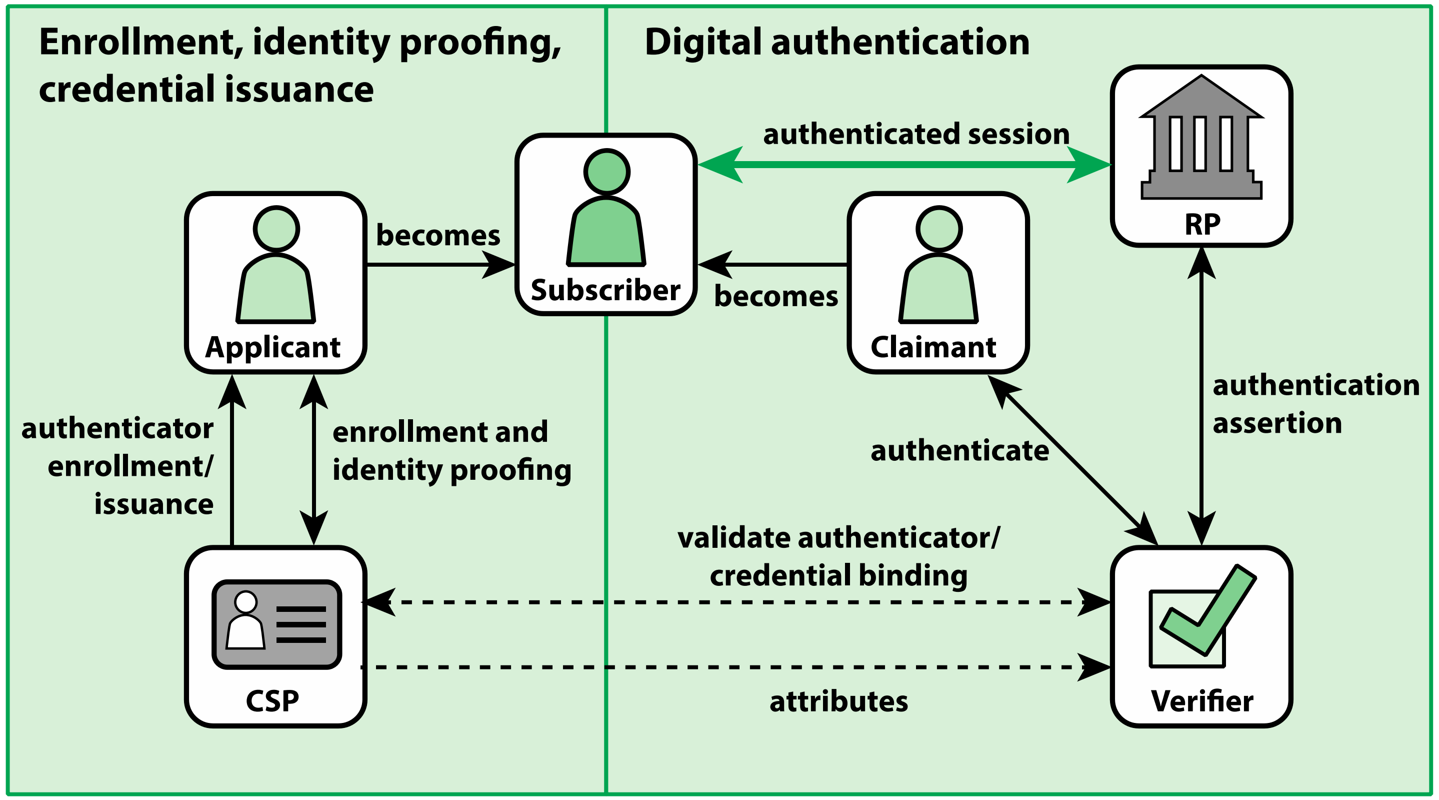
- **Digital identity:**
  - Unique representation of a subject engaged in an online transaction
  - Representation consists of an attribute or set of attributes that uniquely describe a subject within a given context of a digital service
- **Identity proofing:**
  - Establishes that a subject is who they claim to be
  - Collecting, validating, and verifying information about a person
- **Digital authentication:**
  - The process of determining the validity of one or more authenticators used to claim a digital identity
  - Successful authentication provides reasonable risk-based assurances that the subject accessing the service today is the same as the subject that previously accessed the service

Model for Electronic User Authentication

- **NIST SP 800-63-2**

Entities:

- Credential Service Provider (CSP)
- Verifier
- Relying Party (RP)
- Applicant
- Claimant
- Subscriber



Enrollment, identity proofing, credential issuance
Digital authentication
Applicant
becomes
Subscriber
authenticated session
RP
becomes
Claimant
authenticator enrollment/ issuance
enrollment and identity proofing
authenticate
authentication assertion
validate authenticator/ credential binding
CSP
attributes
Verifier

CSP = credential service provider
RP = relying party

Figure 16.1 The NIST 800-63 Digital Identity Model

Means of User Authentication

There are three general means, or *authentication factors*, of authenticating a user’s identity, which can be used alone or in combination:

1. **Knowledge factor:**
   - Requires the user to demonstrate knowledge of secret information.
   - Routinely used in single-layer authentication processes.
   - Passwords, passphrases, personal identification numbers (PINs), or answers to secret questions
2. **Possession factor:**
   - Physical entity possessed by the authorized user to connect to the client computer or portal.
   - This type of authenticator used to be referred to as a token or hardware token.
     - *Connected Hardware token*: connected to a computer logically or physically
       - Items such as smart cards, electronic keycards, USB tokens, physical keys.
     - *Disconnected Hardware token*: Do not directly connected to the client computer
       - An authenticator (software or hardware) to generate a code for Two-factor authentication
3. **Inherence factor:**
   - Static biometrics: fingerprint, retina, and face;
   - Dynamic biometrics: voice, handwriting, and typing rhythm

Authentication Factors

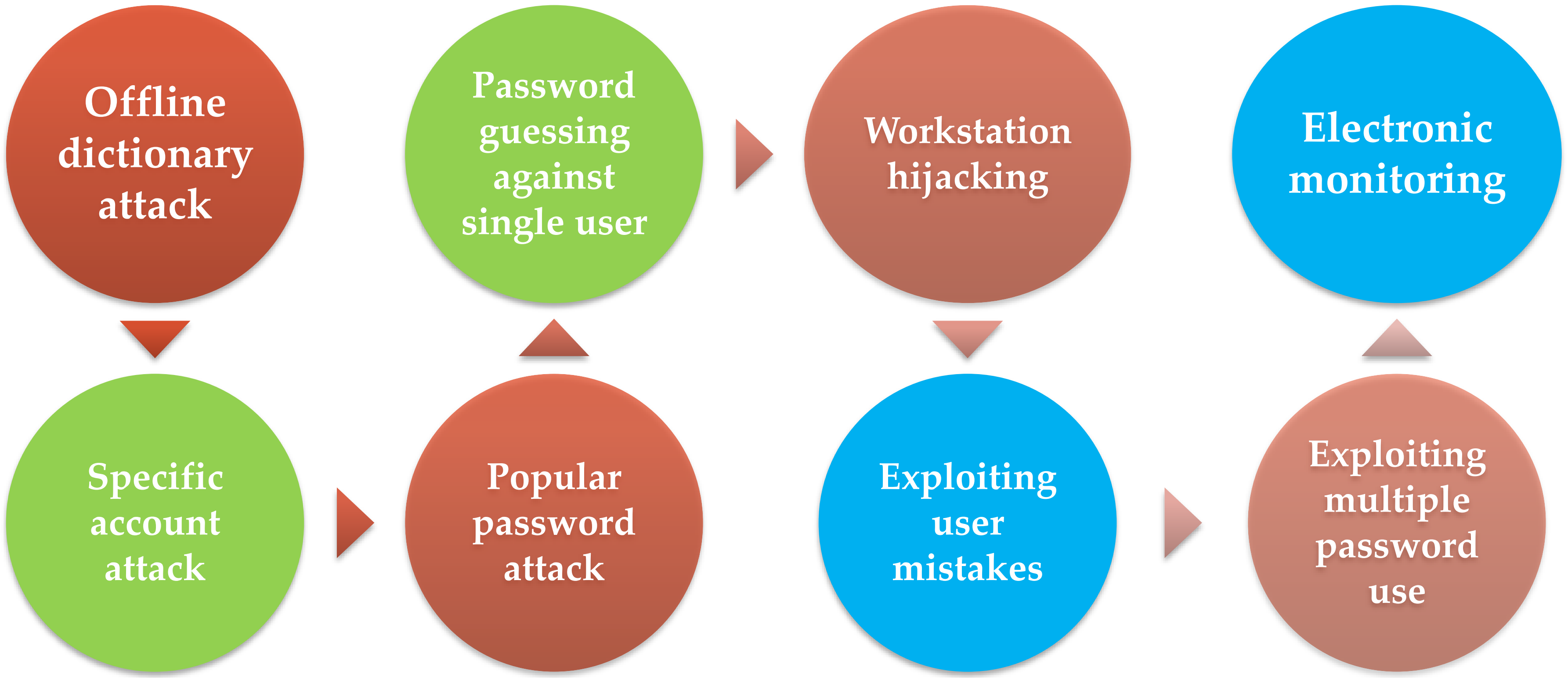
Factor	Examples	Properties
Knowledge	User ID Password PIN	Can be shared Many passwords easy to guess Can be forgotten
Possession	Smart Card Electronic Badge Electronic Key	Can be shared Can be duplicated (cloned) Can be lost or stolen
Inherence	Fingerprint Face Iris Voice print	Not possible to share False positives and false Negatives possible Forging difficult

Password-Based Authentication

- Password system is the widely used line of defense against intruders
  - User provides name/login and password
  - System compares password with the one stored for that specified login
  - The user ID:
    - Determines that the user is authorized to access the system
    - Determines the user’s privileges
    - Is used in discretionary access control
      - For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

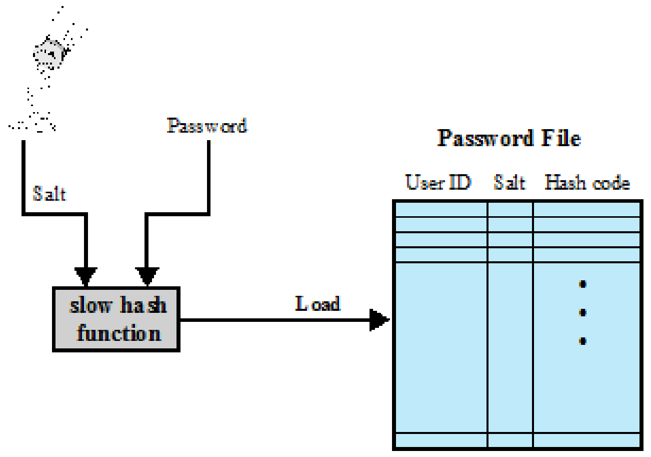
Password-Based Authentication

- Password Vulnerabilities
  - A password-based authentication system maintains a password file indexed by user ID.
    - Usually stores a hashed password.
  - **Attack strategies:**

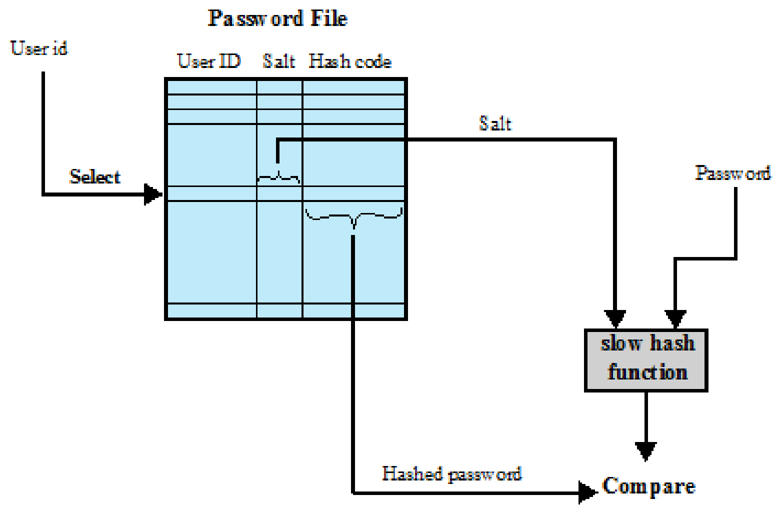


Password-Based Authentication

- The use of hashed passwords
  - The use of hashed passwords and a salt value.
    - Used in all UNIX variants as well as on several other operating systems.
  - The salt serves three purposes:
    - Prevents duplicate passwords from being visible in the password file
    - It greatly increases the difficulty of offline dictionary attacks
    - It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them



(a) Loading a new password



(b) Verifying a password

Password Selection Strategies

User education

Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords

Computer generated passwords

Users have trouble remembering them

Reactive password checking

System periodically runs its own password cracker to find guessable passwords

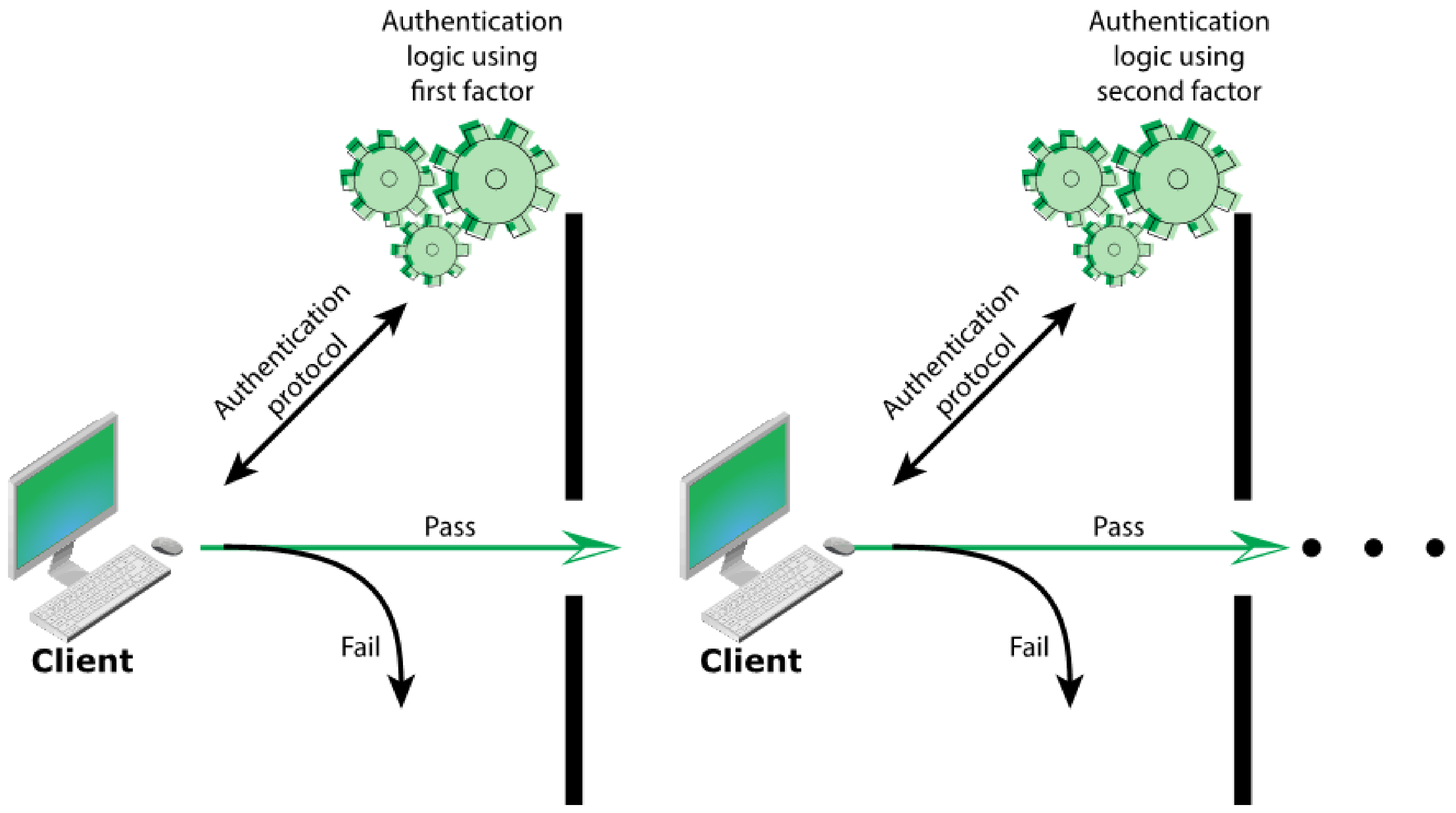
Complex password policy

User is allowed to select their own password, however the system checks to see if the password is allowable, and if not, rejects it

Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable

Multifactor Authentication

- Use of more than one of the authentication means


Authentication logic using first factor
Authentication protocol
Client
Pass
Fail
Authentication logic using second factor
Authentication protocol
Client
Pass
Fail

Mutual Authentication

- Protocols which enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys

Central to the problem of authenticated key exchange are two issues:

Timeliness

- Important because of the threat of message replays
- Such replays could allow an opponent to:
  - compromise a session key
  - successfully impersonate another party
  - disrupt operations by presenting parties with messages that appear genuine but are not

Confidentiality

- Essential identification and session-key information must be communicated in encrypted form
- This requires the prior existence of secret or public keys that can be used for this purpose

Replay Attacks

1. The simplest replay attack is one in which the opponent simply copies a message and replays it later
2. An opponent can replay a timestamped message within the valid time window, this incident can be logged
3. An opponent can replay a timestamped message within the valid time window, but in addition, the opponent suppresses the original message; thus, the repetition cannot be detected
4. Another attack involves a backward replay (sender) without modification. This attack is possible if symmetric encryption is used, and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

Approaches to Coping With Replay Attacks

- Attach a sequence number to each message used in an authentication exchange
  - A new message is accepted only if its sequence number is in the proper order
  - Difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with
  - Generally, not used for authentication and key exchange because of overhead
- Timestamps
  - Requires that clocks among the various participants be synchronized
  - Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time
- Challenge/response
  - Party A, expecting a fresh message from B, first sends B a *nonce* (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value

Example

- TCP Connection:
  - C → S : SYN(sequence Number of client(SNc))
  - S→ C : SYN (sequence Number of Server(SNs)), ACK (SNc)
  - C→S: ACK (SNs)

    Data (SNc + #) transmission

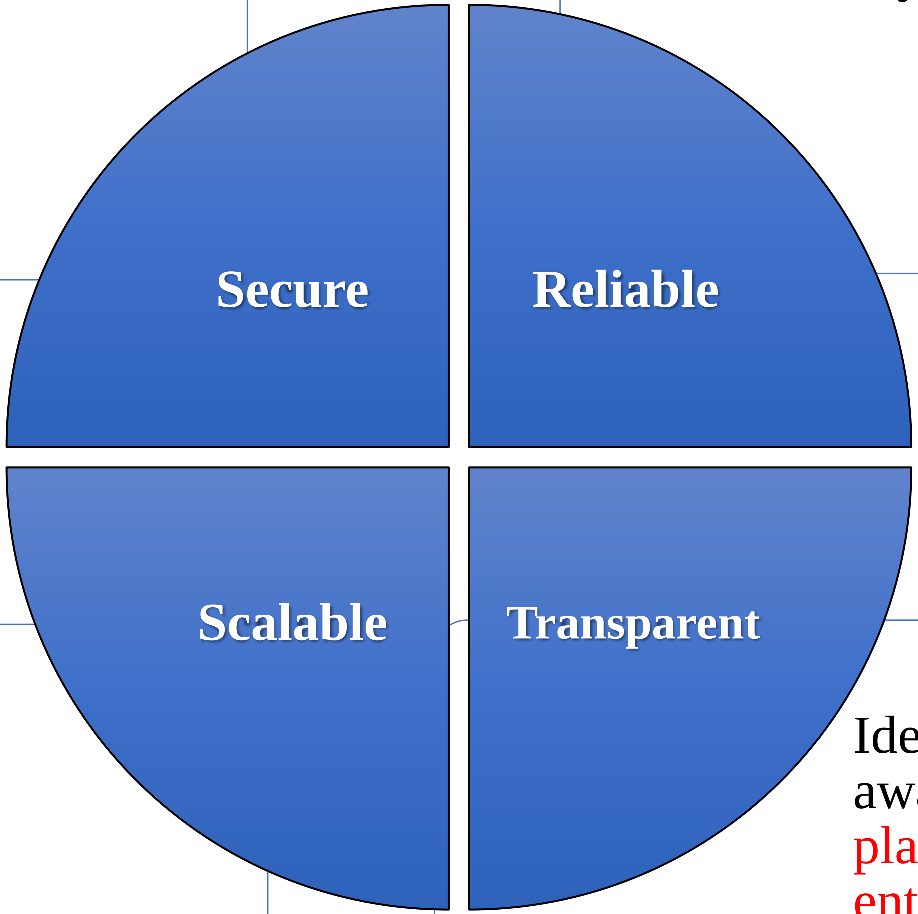
RST ← adversary can reset the existing link

Kerberos

- **Kerberos** is a key distribution and user authentication service developed at MIT
- It addresses an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- **Problem: A workstation cannot be trusted to identify its users correctly to network services:**
  - A user may gain access to a particular workstation and pretend to be another user operating from that workstation
  - A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation
  - A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations
- **Solution:**
  - Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users
    - Relies exclusively on symmetric encryption, making **no** use of public-key encryption

Kerberos Requirements

- The first published report on Kerberos listed the following requirements:



- A network eavesdropper should not be able to obtain the necessary information to impersonate a user

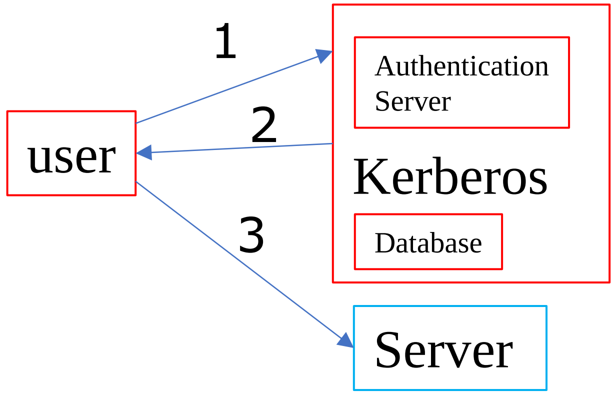
- Should be highly reliable and should employ a distributed server architecture with one system able to back up another

- The system should be capable of supporting large numbers of clients and servers

Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password

Kerberos v4

- Makes use of DES to provide the authentication service
- **Authentication server (AS)**
  - Knows the passwords of all users and stores these in a centralized database
  - Shares a unique secret key with each server
- **Ticket**
  - Created once the AS accepts the user as authentic; contains the user's ID and network address and the server's ID
  - Encrypted using the secret key shared by the AS and the server



➢ Consider the following dialogue (**scenario-1**)

Issues:

1. Need to supply password for each attempt
2. Plaintext transmission of the password [message (1)]
3. Need a new ticket and supply password for every different service

➢ **Solution:**

1. A scheme for avoiding plaintext Passwords
2. Introduce a new Ticket Granting Server (TGS)

Senario-1:

(1) C → AS: $ID_C \| P_C \| ID_V$

(2) AS → C: *Ticket*

(3) C → V: $ID_C \|$ *Ticket*

Ticket = $E(K_v, [ID_C \| AD_C \| ID_V])$

where

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

Kerberos v4

- **Ticket-granting server (TGS)**
  - Issues tickets to users who have been authenticated to AS
  - Each time the user requires access to a new service the client applies to the TGS using the ticket to authenticate itself
  - The TGS then grants a ticket for the particular service
  - The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested

Issues:

1. lifetime associated with the ticket-granting ticket and service-granting ticket.
2. Server authentication

Once per user logon session:

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

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

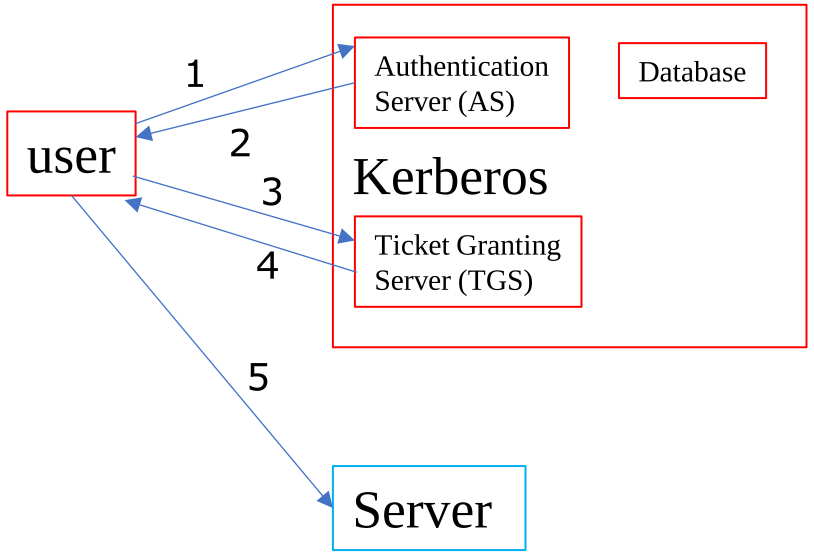
(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$



Kerberos v4

➢Authentication Dialogue

The lifetime associated with the ticket-granting ticket creates a problem:

- If the lifetime is very short (e.g., minutes), the user will be repeatedly asked for a password
- If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay

A network service (the TGS or an application service) must be able to prove that the person using a ticket is the same person to whom that ticket was issued

Servers need to authenticate themselves to users

Kerberos: V4

➢The actual Kerberos protocol.

(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** $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$
$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$
$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

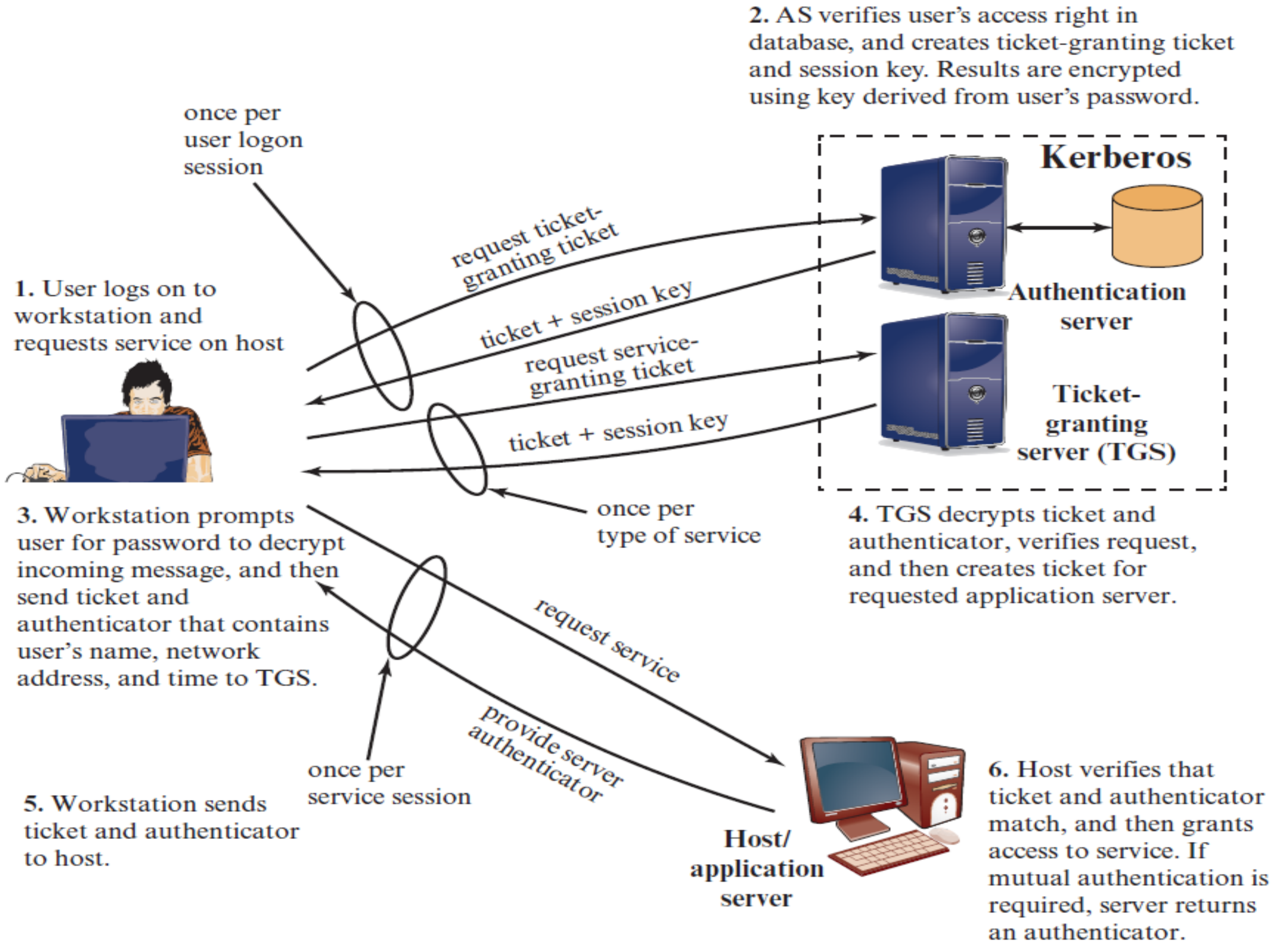
(b) Ticket-Granting Service Exchange to obtain service-granting ticket

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

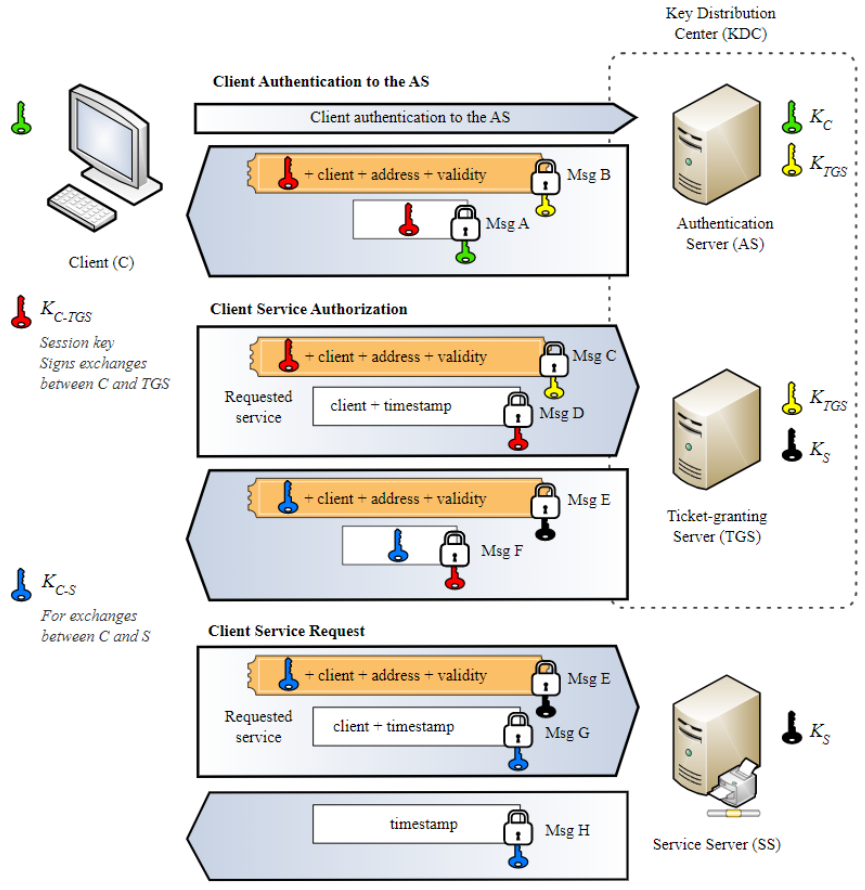
(6) **V → C** $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$
$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

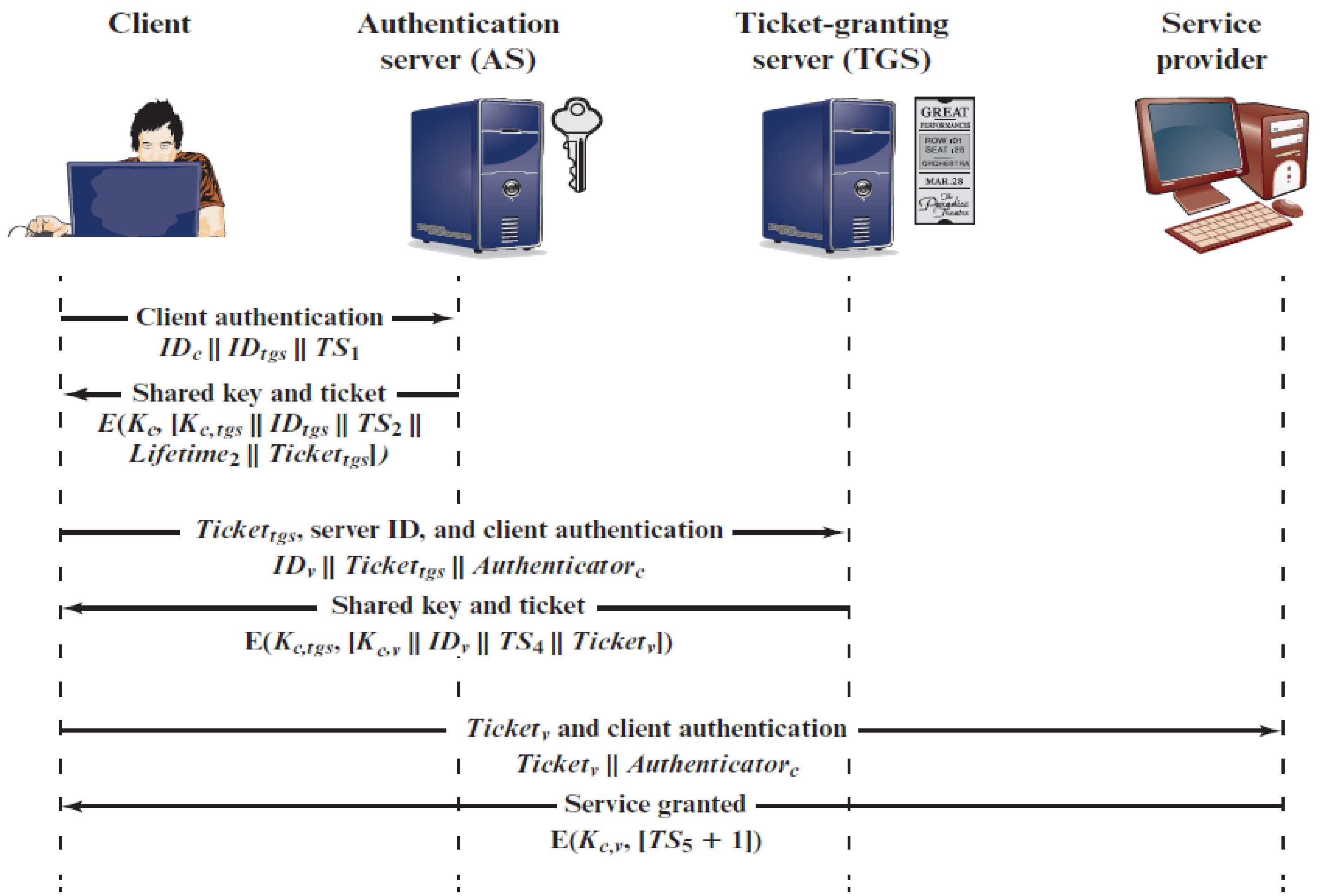
(c) Client/Server Authentication Exchange to obtain service


2. AS verifies user's access right in database, and creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication server
Ticket-granting server (TGS)
once per user logon session
1. User logs on to workstation and requests service on host
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
3. Workstation prompts user for password to decrypt incoming message, and then send ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, and then creates ticket for requested application server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
6. Host verifies that ticket and authenticator match, and then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos Exchanges among the parties



Key Distribution Center (KDC)
Client Authentication to the AS
Client authentication to the AS
+ client + address + validity
Msg B
Msg A
Authentication Server (AS)
K_C
K_{TGS}
Client (C)
$K_{C\text{-}TGS}$
Session key
Signs exchanges between C and TGS
Client Service Authorization
+ client + address + validity
Msg C
Requested service
client + timestamp
Msg D
K_{TGS}
K_S
+ client + address + validity
Msg E
Msg F
Ticket-granting Server (TGS)
$K_{C\text{-}S}$
For exchanges between C and S
Client Service Request
+ client + address + validity
Msg E
Requested service
client + timestamp
Msg G
K_S
timestamp
Msg H
Service Server (SS)



Client
Authentication server (AS)
Ticket-granting server (TGS)
Service provider
Client authentication
$ID_c \parallel ID_{tgs} \parallel TS_1$
Shared key and ticket
$E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
$Ticket_{tgs}$, server ID, and client authentication
$ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
Shared key and ticket
$E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$
$Ticket_v$ and client authentication
$Ticket_v \parallel Authenticator_c$
Service granted
$E(K_{c,v}, [TS_5 + 1])$

Rationale for the Elements of the Kerberos Version 4 Protocol

(a) Authentication Service Exchange

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password. enabling AS and client to verify password. and protecting contents of message (2).
$K_{c, tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

Rationale for the Elements of the Kerberos Version 4 Protocol

(b) Ticket-Granting Service Exchange

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.

Rationale for the Elements of the Kerberos Version 4 Protocol

(b) Ticket-Granting Service Exchange

Message (4)	TGS returns service-granting ticket.
$K_{c, tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c, v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering.
$K_{c, tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay.
$K_{c, tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

Rationale for the Elements of the Kerberos Version 4 Protocol

(c) Client/Server Authentication

Message (5)	**Client requests service.**
$Ticket_V$	Assures server that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.

Rationale for the Elements of the Kerberos Version 4 Protocol

(c) Client/Server Authentication

Message (6)	**Optional authentication of server to client.**
$K_{c,v}$	Assures C that this message is from V.
$TS_5 + 1$	Assures C that this is not a replay of an old reply.
$Ticket_v$	Reusable so that client does not need to request a new ticket from TGS for each access to the same server.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_V	Assures server that it has decrypted ticket properly.
TS_4	Informs server of time this ticket was issued.
$Lifetime_4$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_5	Informs server of time this authenticator was generated.

Kerberos Realm

- A Kerberos Realm: is a set of managed nodes that share the same Kerberos database
- The database resides on the Kerberos master computer system, which should be kept in a physically secure room
- A read-only copy of the Kerberos database might also reside on other Kerberos computer systems
- All changes to the database must be made on the master computer system
- Changing or accessing the contents of a Kerberos database requires the Kerberos master password

Kerberos Realm and Multiple Kerberi

- A full-service Kerberos environment (it is called Kerberos realm) consisting of a Kerberos server, a number of clients, and a number of application servers requires that:

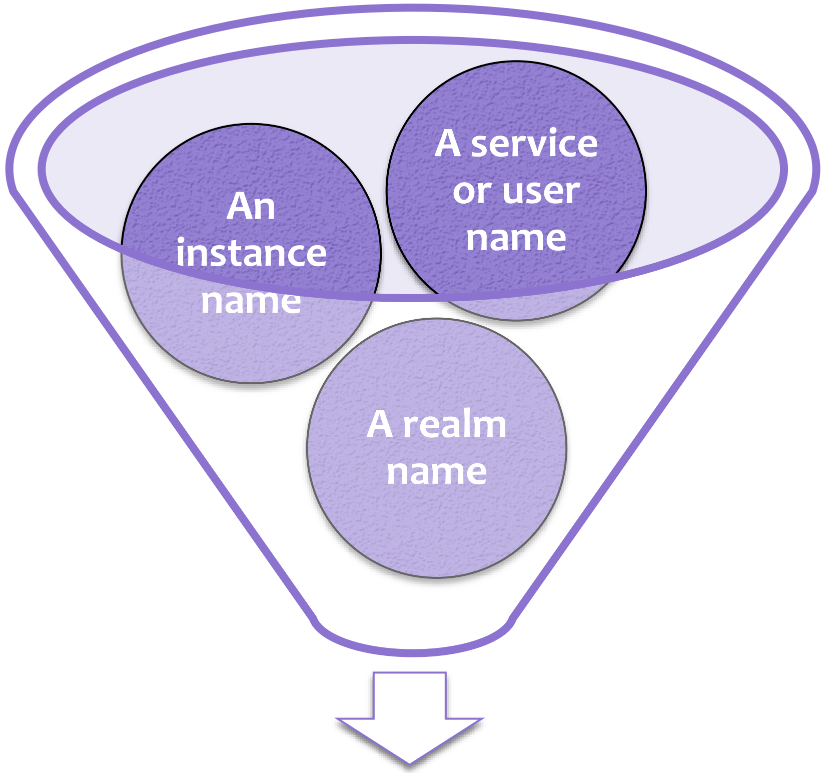
  1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database; all users are registered with the Kerberos server
  2. The Kerberos server must share a secret key with each server; all servers are registered with the Kerberos server

  For two realms to support interrealm authentication, a third requirement is added:

  3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm; the two Kerberos servers are registered with each other

Kerberos Principal

- Kerberos Principal: is a service or user that is known to the Kerberos system
  - It is identified by its principal name that has three parts:
    - A service or user name
    - An instance name
    - A realm name

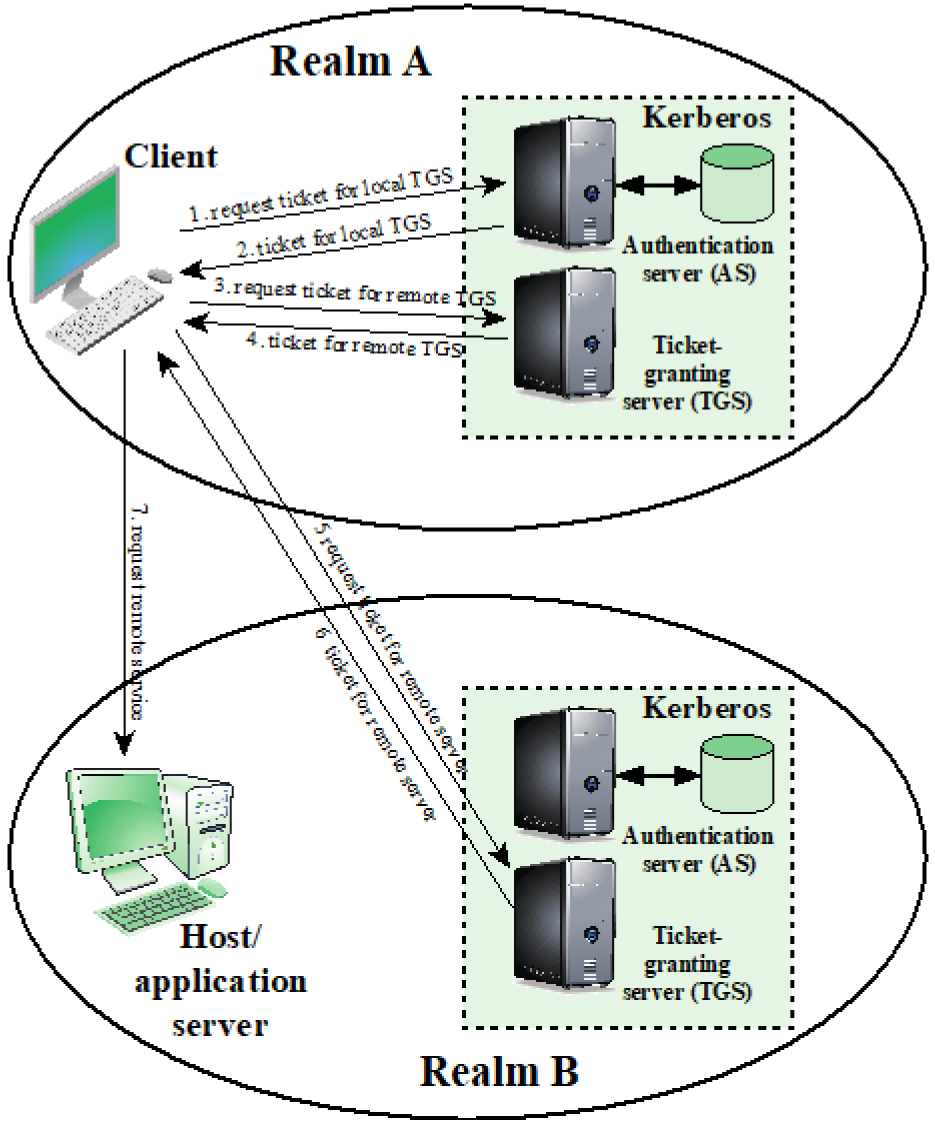


Three parts of a principal name

Kerberos Principal

- Request for Service in Another Realm
  - 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.

(1) $C \rightarrow AS$:	$ID_c \parallel ID_{tgs} \parallel TS_1$
(2) $AS \rightarrow C$:	$E(K_c, [K_{c,\,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$
(3) $C \rightarrow TGS$:	$ID_{tgsrem} \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) $TGS \rightarrow C$:	$E(K_{c,tgs}, [K_{c,\,tgsrem} \parallel ID_{tgsrem} \parallel TS_4 \parallel Ticket_{tgsrem}])$
(5) $C \rightarrow TGS_{rem}$:	$ID_{vrem} \parallel Ticket_{tgsrem} \parallel Authenticator_c$
(6) $TGS_{rem} \rightarrow C$:	$E(K_{c,tgsrem}, [K_{c,\,vrem} \parallel ID_{vrem} \parallel TS_6 \parallel Ticket_{vrem}])$
(7) $C \rightarrow V_{rem}$:	$Ticket_{vrem} \parallel Authenticator_c$


Realm A
Client
Kerberos
1. request ticket for local TGS
2. ticket for local TGS
3. request ticket for remote TGS
4. ticket for remote TGS
Authentication server (AS)
Ticket-granting server (TGS)
7. request remote service
5. request ticket for remote server
6. ticket for remote server
Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Host/ application server
Realm B

Differences Between Versions 4 and 5

- **Version 5 is intended to address the limitations of version 4 in two areas:**
  - Environmental shortcomings
    - Encryption system dependence: DES to AES
    - Internet protocol dependence: allows any type of addresses other than IP
    - Message byte ordering: uses ASN.1 format to provide an unambiguous byte ordering
    - Ticket lifetime: from 8-bit to use start and end time
    - Authentication forwarding: allow credentials issued to one client to be forwarded to some other host and used by some other client
    - Interrealm authentication: efferent in ver5
  - Technical deficiencies
    - Double encryption of the tickets: changed to one encryption
    - PCBC encryption: from Propagating Cipher Block Chaining to CBC
    - Session keys: Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket: In version 5, it is possible for a client and server to negotiate a new subsession key each time
    - Password attacks: Both versions are vulnerable to a password attack

Kerberos Version 5 Message Exchanges

•

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

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

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

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

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

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

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

$Ticket_v = \mathrm{E}(K_v, [Flags \parallel K_{c,v} \parallel Realm_c \parallel ID_C \parallel \mathrm{AD}_C \parallel Times])$

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

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

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

(6) V → C $\mathrm{E}_{K_{c,v}} [\, TS_2 \parallel Subkey \parallel Seq\# \,]$

$\mathrm{Ticket}_v = \mathrm{E}(K_v, [\mathrm{Flags} \parallel K_{c,v} \parallel Realm_c \parallel ID_C \parallel AD_C \parallel Times])$

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

(c) Client/Server Authentication Exchange to obtain service