IT 422 Network Security

Authentication - Kerberos

Yasser F. O. Mohammad

REMINDER 1: Public vs. Shared Key

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key.	**Needed to Work:** 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	**Needed for Security:** 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

REMINDER 2: How to Break RSA?

1. Factorize n = Find p and q.
2. Find $\Phi(n)=(p-1)*(q-1)$
3. Find $d=e^{-1} \bmod \Phi(n)$

Now you have the private key!!!!

The only problem is that it is mathematically very difficult to factorize n.

REMINDER 3: Diffie-Hellman

- The point is that users A and B will be able to calculate the secret key using only:
  1. His private key
  2. Other's public key

- Eve needs to do a discrete logarithm because she does not have any of the private keys.

Global Public Elements

q	prime number
α	$\alpha < q$ *and* α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

REMINDER 4: Man-in-the-Middle Attack

$$A \rightarrow E : Y_A$$

$$E \rightarrow B : Y_{D1}$$

$$\begin{cases} E : K_2 = {Y_A}^{X_{D2}} \bmod q \\ B : K_1 = {Y_{D1}}^{X_B} \bmod q \end{cases}$$

$$B \rightarrow E : Y_B$$

$$E \rightarrow A : Y_{D2}$$

$$\begin{cases} E : K_1 = {Y_B}^{X_{D1}} \bmod q \\ A : K_2 = {Y_{D2}}^{X_A} \bmod q \end{cases}$$

Now E has K_1 shared with B and
K_2 shared with A

A and B think that they share the key with each other

$$A \rightarrow E : E(K_2; M)$$

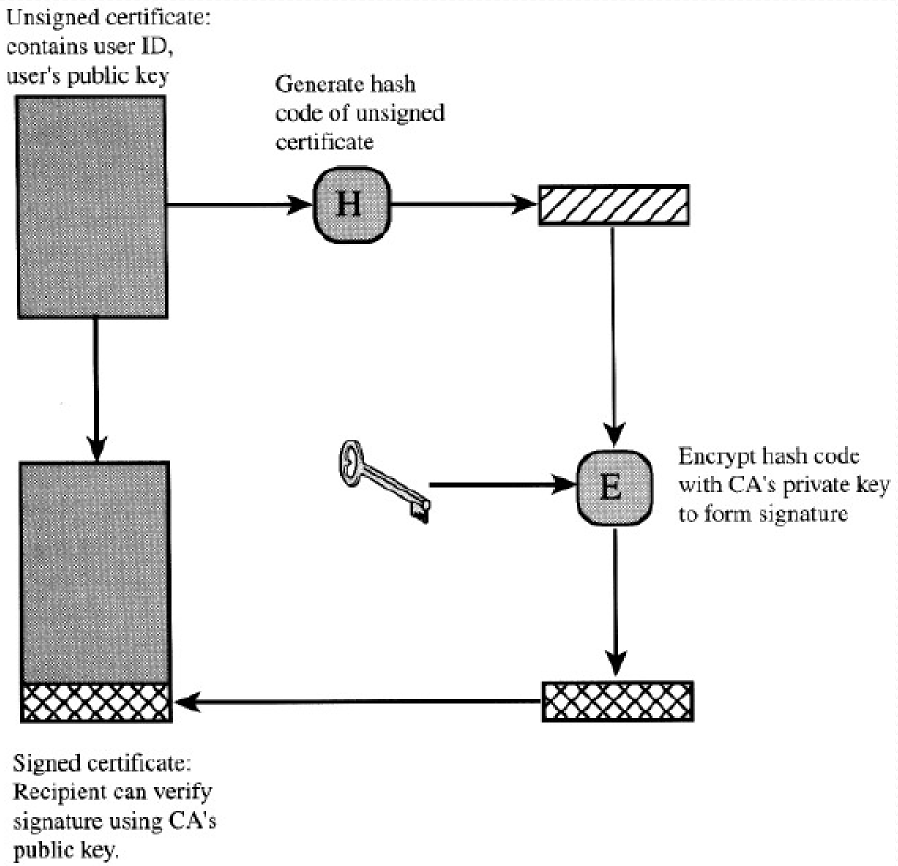
$$E \rightarrow B : E(K_1; M)$$

$$or$$

$$E \rightarrow B : E(K_1; M')$$

REMINDER 5: Distribution of Public Keys

- Public Key Certificates
- CA=Certification Authority
- CA's sign public keys of users with its private key
- X.509 standard
- Used in SSL, Secure Electronic Transaction (SET), S/MIME



User Authentication

- Standalone PC
  - pass File

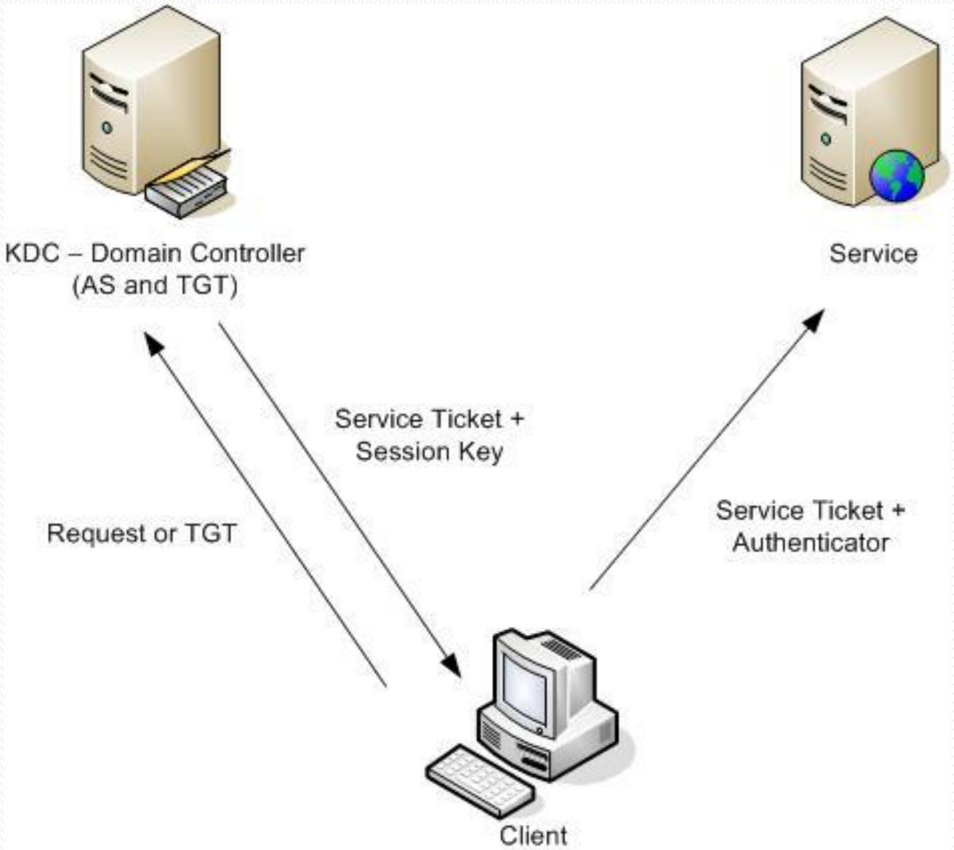
- Over a network
  - Kerberos

How to Authenticate

- What you know.
  - Password/passphrase
- What you have.
  - Smart Cards
- What you are: Static Biometrics.
  - Fingerprint/Face recognition
- What you do: Dynamic Biometrics.
  - Handwriting characteristics

Kerberos Goal

- To allow two-way authentication between human users and network services
- Uses only shared key cryptography
- Uses only passwords for authentication



Why only passwords

- Cheap
- Easy
- Fast (small bandwidth)

- Challenging!!!
  - For designers

Why Trusted Third Party

- Reduces Key management overhead:
  - Number of shared keys = N.clients + N.services
  - Without TTP
    - Number of shared Keys= N.clients * N.services

- Provides name resolution services

First Trial

(1) C $\longrightarrow$ AS: $ID_C || P_C || ID_V$

(2) AS $\longrightarrow$ C: $Ticket$

(3) C $\longrightarrow$ V: $ID_C || Ticket$

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

C	Client
AS	Authentication Server
V	Service/Server
ID_c	User's Identifier on C
ID_v	Service Identifier
P_c	Password of C
AD_c	Network Address of C
K_*	$K_{AS\text{-}*}$

- What is the problem?
    1. Password is transmitted in PLAIN

       Solution: Encrypt it
    2. 1 password entry for every service

       Solution: Use one extra service to authenticate to all

Second Trial

Once per user logon session:

(1) C ⟶ AS: $ID_C || ID_{tgs}$

(2) AS ⟶ C: $E(K_c, Ticket_{tgs})$

Once per type of service:

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

(4) TGS ⟶ C: $Ticket_v$

Once per service session:

(5) C ⟶ V: $ID_C || Ticket_v$

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

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

C	Client
AS	Authentication Server
V	Service/Server
ID_c	User's Identifier on C
ID_v	Service Identifier
P_c	Password of C
AD_c	Network Address of C
K_*	K_{AS-*}
TGS	Ticket Generation Service
TGT	Ticket Granting Ticket

- Why using *TS (Time stamp)* and *lifetime* in tickets?
  - Prevents easy replay attack (wait and gain the workstation)
- What is the problem?
  1. Lifetime balance (too short→ cumbersome, too long → not secure)
  2. Replay during lifetime
  3. How to authenticate the service to the client!!

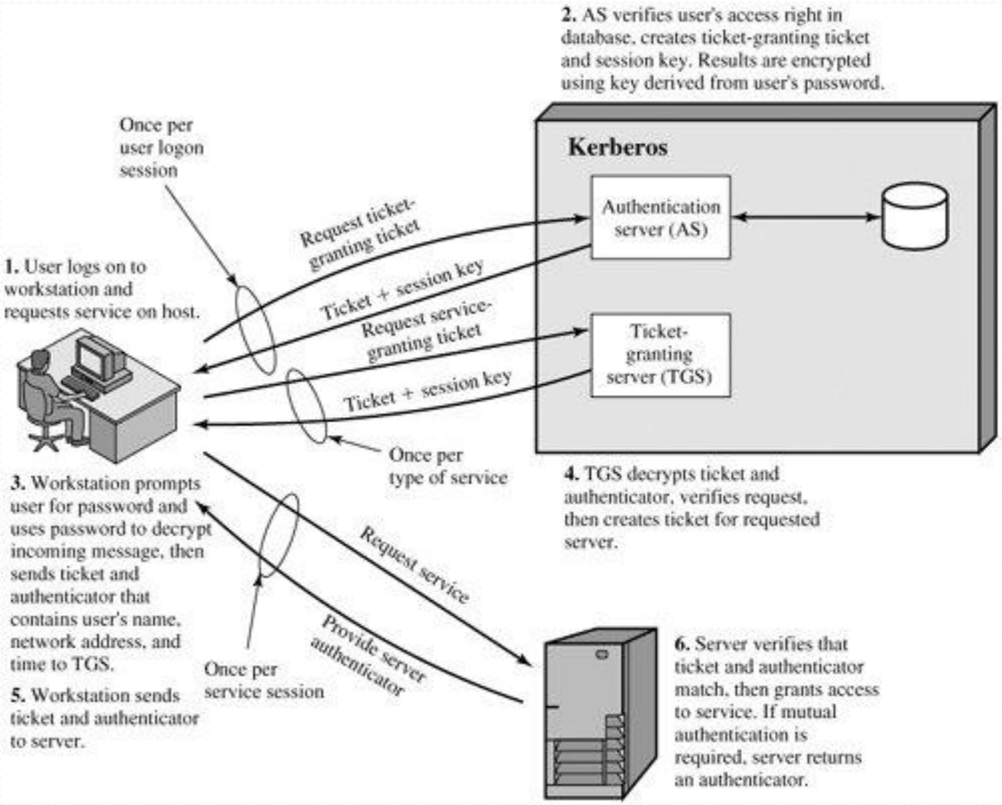
Requirements until now

- Enter the password once
  - Use TGT
- Do not depend much on network address
  - Use Authenticators
- Resist replying the ticket (Allow the service to know that the one using the ticket now is the one for whom it was issued)
  - Use Authenticators
- Two-way authentication
  - Secret information in the ticket

Added assumptions

- The network is loosely synchronized (to use timestamps)
- AS and TGS are secure and their databases are unreadable for anyone
- *This is on of the weakest point about Kerberos (Almost!!). The other one is password attack against message 1.*

Overview of Kerberos 4



Kerberos 4 Full Exchange

(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

Rationale for C-AS 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

(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

Rationale for C-TGS 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
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

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

(4) TGS $\longrightarrow$ 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

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 tamperig
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 C-V exchange

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
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

(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

$Authenticator_c$	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

- AS knows all users (clients)
- AS knows all services/servers

- *Usually AS and TGS are together and called the Kerberos Server*

Kerberos Principal

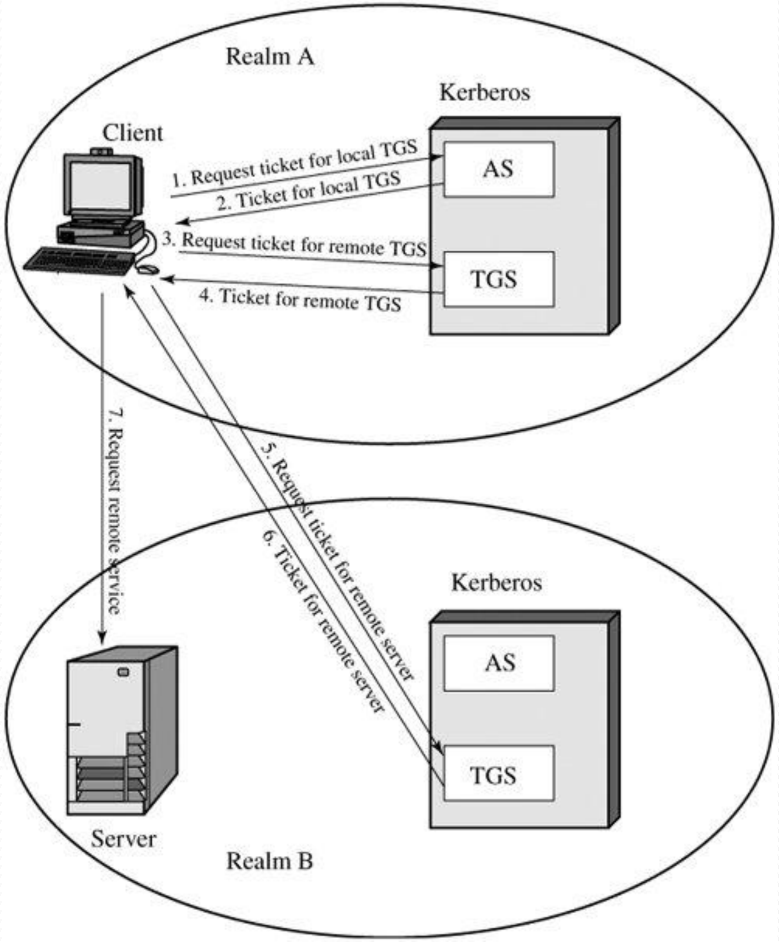
- A person or service known to the Kerberos Server

Multiple Realm Authentication

- Each Kerberos server much share a key with each other Kerberos server (in version 4)

- In summary:

  Get A TGT from your local TGS for the TGS of the other realm, then use this ticket to request tickets in services in the other realm



Multiple Realm Authentication

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

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

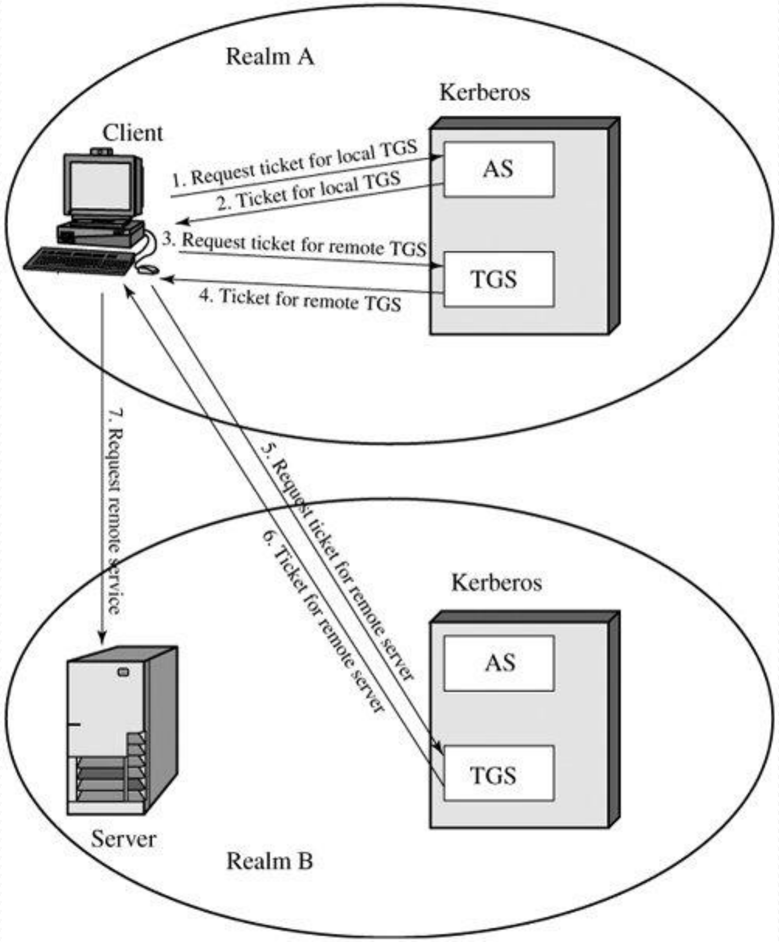
(3) C $\longrightarrow$ TGS: $ID_{tgsrem} || Ticket_{tgs} || Authenticator_c$

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

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

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

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



Shortcomings of Kerberos 4
1. Environmental Limitations

- Encryption System Dependence (DES)
  - Solution in 5: Encrypted text is tagged with Alg. name
- Internet Protocol Dependence (IP)
  - Solution in 5: messages are tagged with protocol type
- Message Byte Ordering Independence
  - Solution in 5: All messages use a predefined ordering
- Ticket Lifetime (21 hours max.)
  - Solution in 5: explicit start and end times
- Authentication Forwarding
  - Solution in 5: V_1 can use C's credentials to access V_2.
- Interrealm Authentication (n*[n-1] keys)
  - Solution in 5: Reduced number of keys

Shortcomings of Kerberos 4
2. Protocol Limitations

- Double Encryption
  - Messages 2, 4 are encrypted twice for no reason
- Propagating Cipher Block Chaining (PCBP)
  - Vulnerable to an attack since 1989
- Session Keys
  - Protects tickets and exchanges
- Password Attack
  - Capture message 1 and try to find the password (the key is as difficult as the password)

Kerberos 5 Exchange

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

(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 $\text{Options} \| ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

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

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

$Ticket_v = E(K_v, [\text{Flags} \| K_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_c \| \text{Times}])$

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

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

(5) C → V $\text{Options} \| \text{Ticket}_v \| \text{Authenticator}_c$

(6) V → C $E_{K_{c,v}}[\text{TS}_2 \| \text{Subkey} \| \text{Seq\#}]$

$\text{Ticket}_v = E(K_v, [\text{Flags} \| K_{c,v} \| \text{Realm}_c \| \text{ID}_C \| \text{AD}_C \| \text{Times}])$

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

(c) Client/Server Authentication Exchange to obtain service

---

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

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

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| \text{ID}_c \| \text{AD}_c \| \text{ID}_{tgs} \| \text{TS}_2 \| \text{Lifetime}_2])$

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

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

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

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| \text{ID}_C \| \text{AD}_C \| \text{ID}_{tgs} \| \text{TS}_2 \| \text{Lifetime}_2])$

$Ticket_v = E(K_v, [K_{c,v} \| \text{ID}_C \| \text{AD}_C \| \text{ID}_v \| \text{TS}_4 \| \text{Lifetime}_4])$

$Authenticator_c = E(K_{c,tgs}, [\text{ID}_C \| \text{AD}_C \| \text{TS}_3])$

(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)

$Ticket_v = E(K_v, [K_{c,v} \| \text{ID}_c \| \text{AD}_c \| \text{ID}_v \| \text{TS}_4 \| \text{Lifetime}_4])$

$Authenticator_c = E(K_{c,v}, [\text{ID}_c \| \text{AD}_C \| \text{TS}_5])$

(c) Client/Server Authentication Exchange to obtain service

Kerberos version 4

Major Differences between 4 and 5

- Use of nonces.
- Ability to exchange subkeys
- Inclusion of flags and options
- Supporting renewal and forwarding
- Supporting pre-authentication to support more secure transmission of the password (e.g. public key/biometrics).

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.
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.