# Kerberos Ticket Exchanges

Created by

Thomas Jerry Scott

2004

# What's In a Kerberos Packet?

- A kerberos packet contains two parts:
  - A cleartext portion



| Domain Name |
| --- |
| Principal Name |
| Timestamp |

  - An encrypted portion



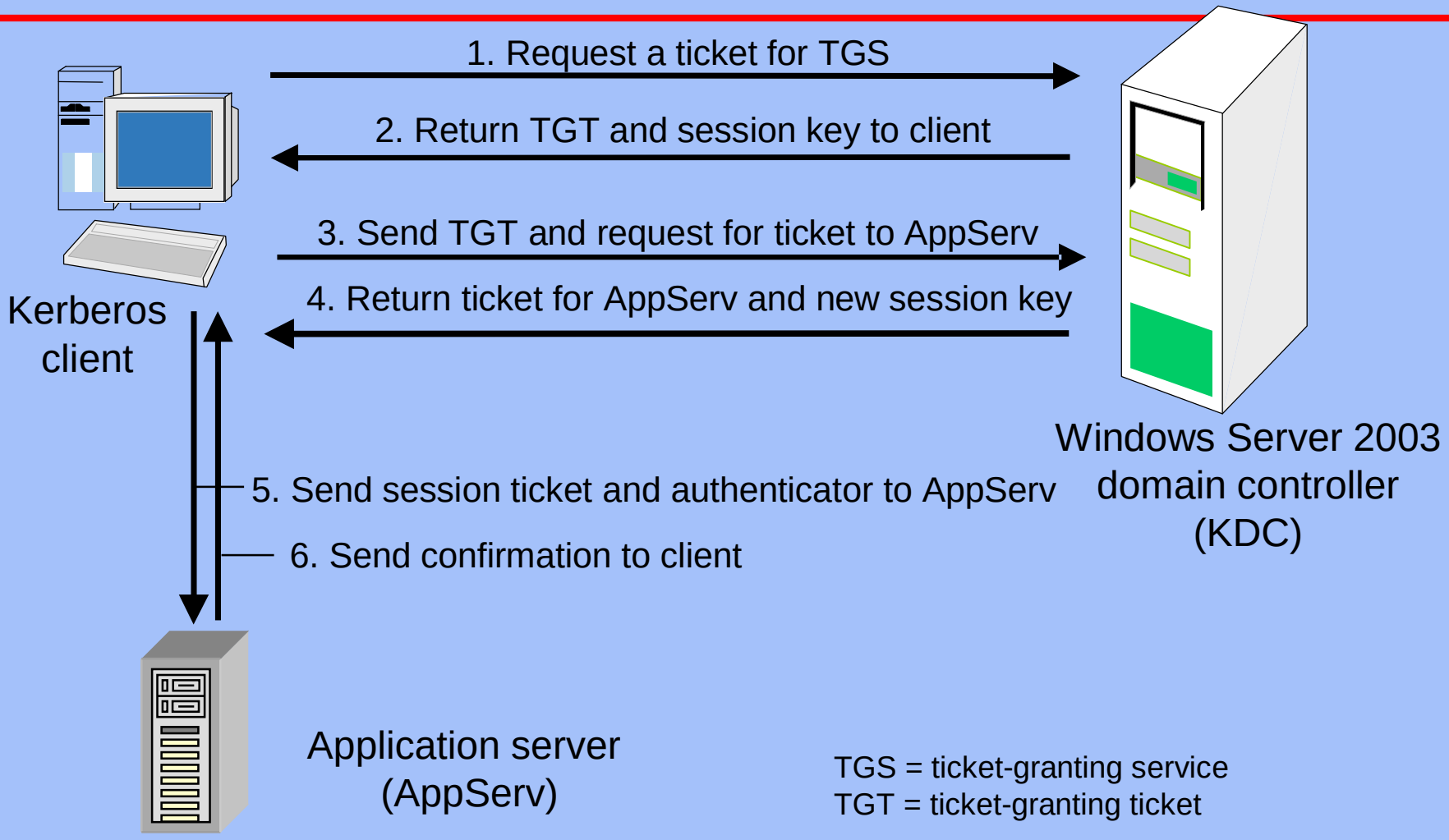
| Ticket Flags |
| --- |
| Encryption Key |
| Timestamp |
| Host Addresses |
| Authorization Data |

# Notation for a Kerberos Packet

To help understand Kerberos packet flow, we use the following syntax

| Symbol | Meaning |
|---|---|
| { authN} | Authenticator number N |
| $K_C$ | Kerberos Key for client C |
| $K_K$ | KDC's Kerberos Key |
| $K_S$ | Server S Kerberos Key |
| {TGT} $K_K$ | Ticket Granting Ticket, encrypted with KDC's Key |
| $\{T_S\}$ $K_S$ | Session Ticket for Server S, encrypted with S's Key |
| $K_{C\,K}$ | Kerberos session key for client C and the KDC |
| $K_{C,S}$ | Kerberos session key for client C and the KDC |

# Steps to Kerberos Authentication




TGS = ticket-granting service
TGT = ticket-granting ticket

# Kerberos First Ticket Details

- Windows KDC's use "pre-authentication"
  - First request from client is encrypted with $K_C$, the hash of the client's password
    - The key $K_C$ is known to the KDC and the Client
- The Client request is [Clear1]{auth1} $K_C$
  - Clear1 contains the UserName, domain name, and a timestamp
  - Auth1 contains this and other information encrypted with the Client's password hash

# KDC's Actions to First Request

- The KDC gets [Clear1]{auth1} $K_C$ and
  - Reads [Clear1], and uses the Client's password hash, $K_C$ , to decrypt {auth1}
  - The decrypted contents of auth1 are compared to those in Clear1
  - If these match, the KDC authenticates the Client and provides a Ticket Granting Ticket, {TGT} $K_K$ back to the client
    - The client cannot read the {TGT} - no $K_K$

# Step 2: Sending the TGT To Client

- The KDC sends the following to the client

  [Clear2] {TGT} $K_K$ {Auth2} $K_C$

- If the timestamps in [Clear1] and [Client2] match, the client is satisfied that it's request reached a KDC
- Client uses $K_C$ to decrypt {Auth2} $K_C$
- Auth2 has the Client's token info and the session key $K_{C,K}$ for future use with KDC

# Client Now Has TGT and Token

- Once the client has received the TGT and verified its timestamp
  - It now has a token to attach to its current process and future processes it creates
  - It has two keys in its key credentials cache
    - $K_{C,K}$ to communicate with KDC in the future
    - $K_C$ will not be used again in this login session
  - It will use the TGT in the future when it wants to connect to a member server, S.

# Step 3: Client to Member Server

- When the client wishes to visit a member server, it sends the following to KDC
  - [Clear3] {TGT} $K_K$ {auth3} $K_{C,K}$
- The TGT contains
  - The session key $K_{C,K}$
  - The SIDS and token info for this client.
  - The KDC does not need to refigure this info!

# Step 3: The {auth3} Info

- The client sends

  [Clear3] and {Auth3} to the KDC

  – Clear3 contains another timestamp and the client name and domain name

  – Auth3 contains the timestamp and the info that Client C wants to visit Server S

  – The KDC uses $K_K$ to decrypt the TGT and gets the token info and  from it.  Using $K_{C,K}$ , it decrypts Auth3 and verifies the client request.

# Step 4: Send Session Ticket

- The KDC verifies the client request, and sends the Client

$$[\text{Clear4}]\ \{T_S\}\ K_S\ \{\text{Auth4}\}\ K_{C,K}$$

- The client uses $K_{C,K}$ to decrypt {Auth4}.
- Auth4 contains the following
  - A session key $K_{C,S}$ to use with Server S
  - An encrypted timestamp

# Client C's Cache After Step 4

- After the Client gets the second packet from the KDC, its credentials cache is
  - $\{TGT\}$ $\{T_S\}$ $K_C$ $K_{C,K}$ and $K_{C,S}$
- The client cannot read the TGT and the special session ticket $T_S$ .

# Step 5: Client Sends Ticket to S

- Now the client has what it needs to connect to server S.
- To connect to Server S, it sends the following to server
  - [Clear5] $\{T_S\}K_S$ {Auth5} $K_{C,S}$

# Step 5: What does Server S do?

- Once Server S gets the information from Client C, it
  - Uses its machine Key, $K_S$ to decrypt the session ticket $\{T_S\}K_S$ T
  - From this ticket, Server S gets the session key, $K_{C,S}$ , that client C used to encrypt Auth5
  - Server S then decrypts Auth5, and when its TimeStamp agrees with the one in Clear5, Server S has authenticated Client C.

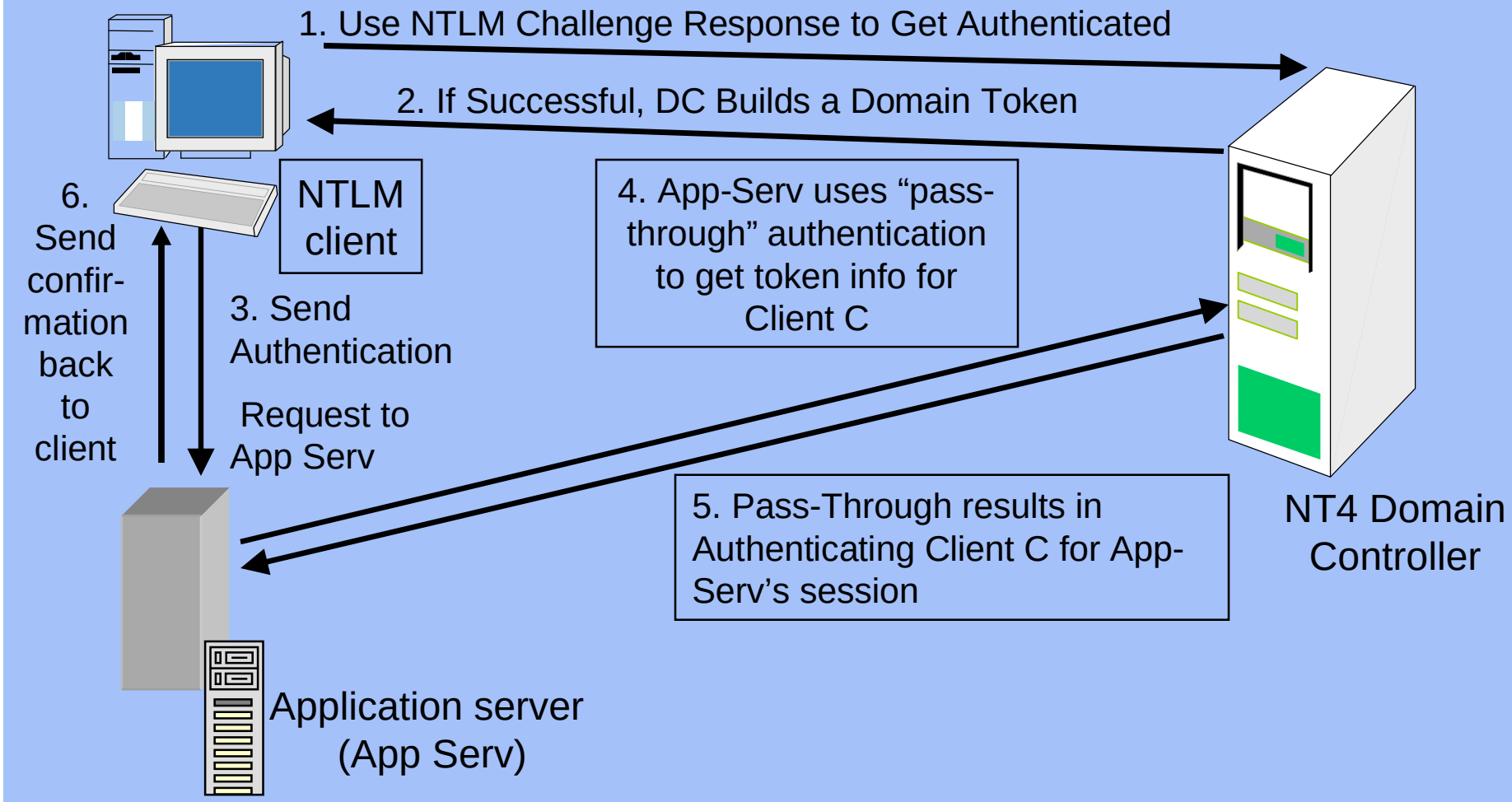
# Step 5:Server S's Activity

- Once Server S has authenticated Client C
  - It uses the Token information in $\{T_S\}$ to build a token for Client C
  - Server S does not have to go back to the KDC to authenticate Client C
  - This allows Kerberos systems to scale authentication more efficiently than NT member servers in NT domains did.

# Step 6:  Server S back to Client C

- In the last step, Server S can send information back to Client C

# NTLM Authentication



# Questions and Answers

- Review
  - What are the encryption keys?
  - What are session tickets?
  - What are session keys?
  - What is in each of the ticket pairs?
  - How does Kerberos authentication scale better than NTLM?