School of Electrical Engineering and Computing

SENG2250/6250
SYSTEM AND NETWORK SECURITY
(S2, 2020)

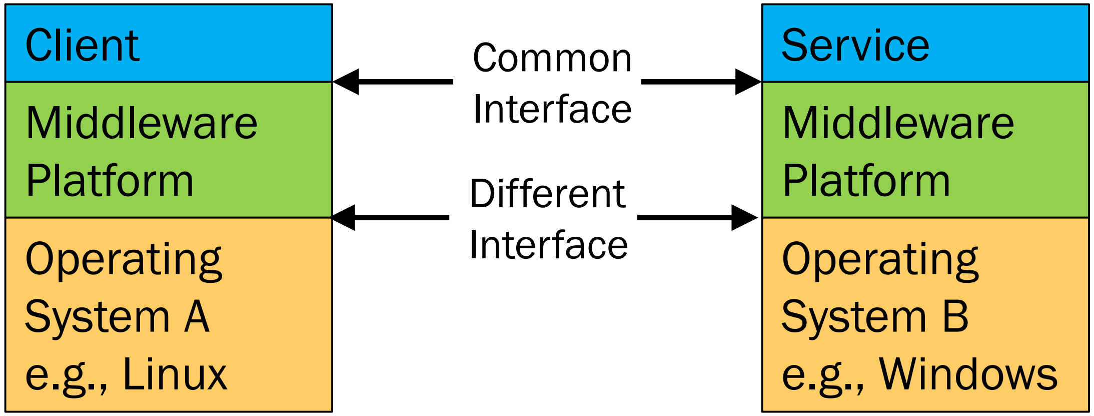
Distributed System Security
(Part 1 - Kerberos)

THE UNIVERSITY OF NEWCASTLE
AUSTRALIA

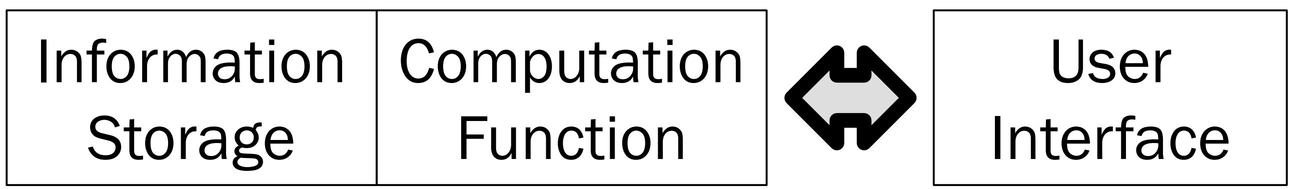
Outline

- Distributed Computing
  - *Client/Server Model*
  - *Remote Procedure Call*
- Kerberos
  - *Motivations*
  - *Basic Ideas*
  - *Technical details*
  - *Limitations*

Middleware Platform



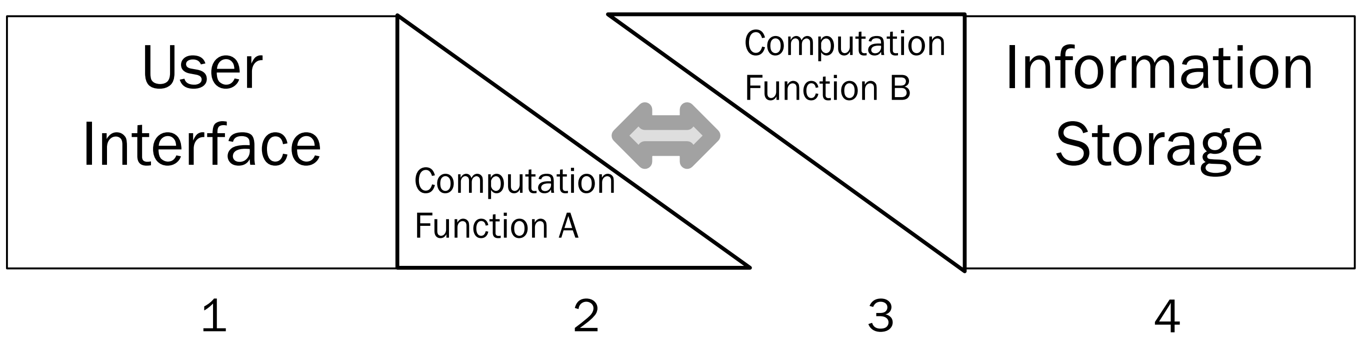
Classic Computing Model



Classic Computing Model

Information Storage (Retrieve data points of curve)	Computation Function (Create bit-map of curve)	⇔	User Interface (Display bitmap)
User Interface (Display bitmap)	Computation Function (Create bit-map of curve)	⇔	Information Storage (Retrieve data points of curve)

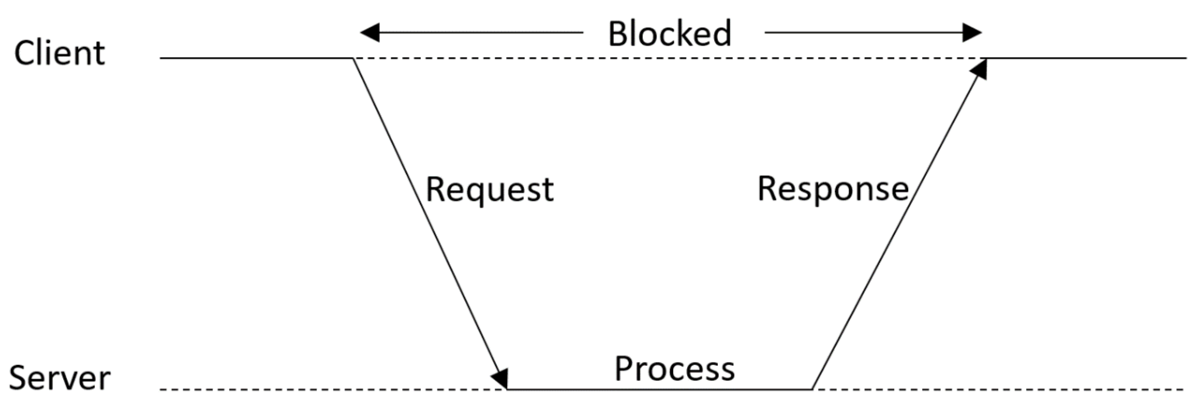
Distributed Computing



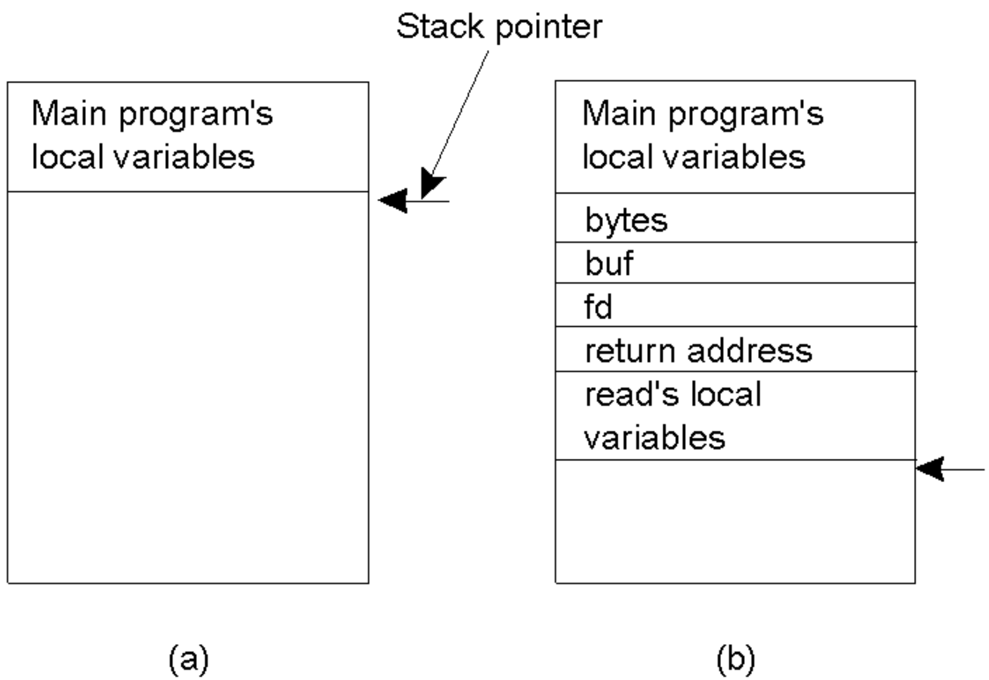
1. Display bitmap
2. Uncompress representation of curve and create bitmap
3. Calculate curve from data points and compress representation
4. Retrieve data points of curve

Basic Client/Server Model

- Client and server are to be active simultaneously.
- Client issues request and wait for reply.
- Server usually to be waiting for requests and process them.
- Client is blocked while waiting for reply.
- Failures have to be handled immediately.
- Not suitable to many scenarios.



Local Procedure Call



- (a) The stack before local procedure call.
- (b) The stack during local procedure call.

Remote Procedure Call

- Client invokes procedures and execution takes place at the remote server

- Aims
  - *Avoid explicit message to be exchanged between processes.*
  - *Make the procedure call on remote machine to look like as local call.*

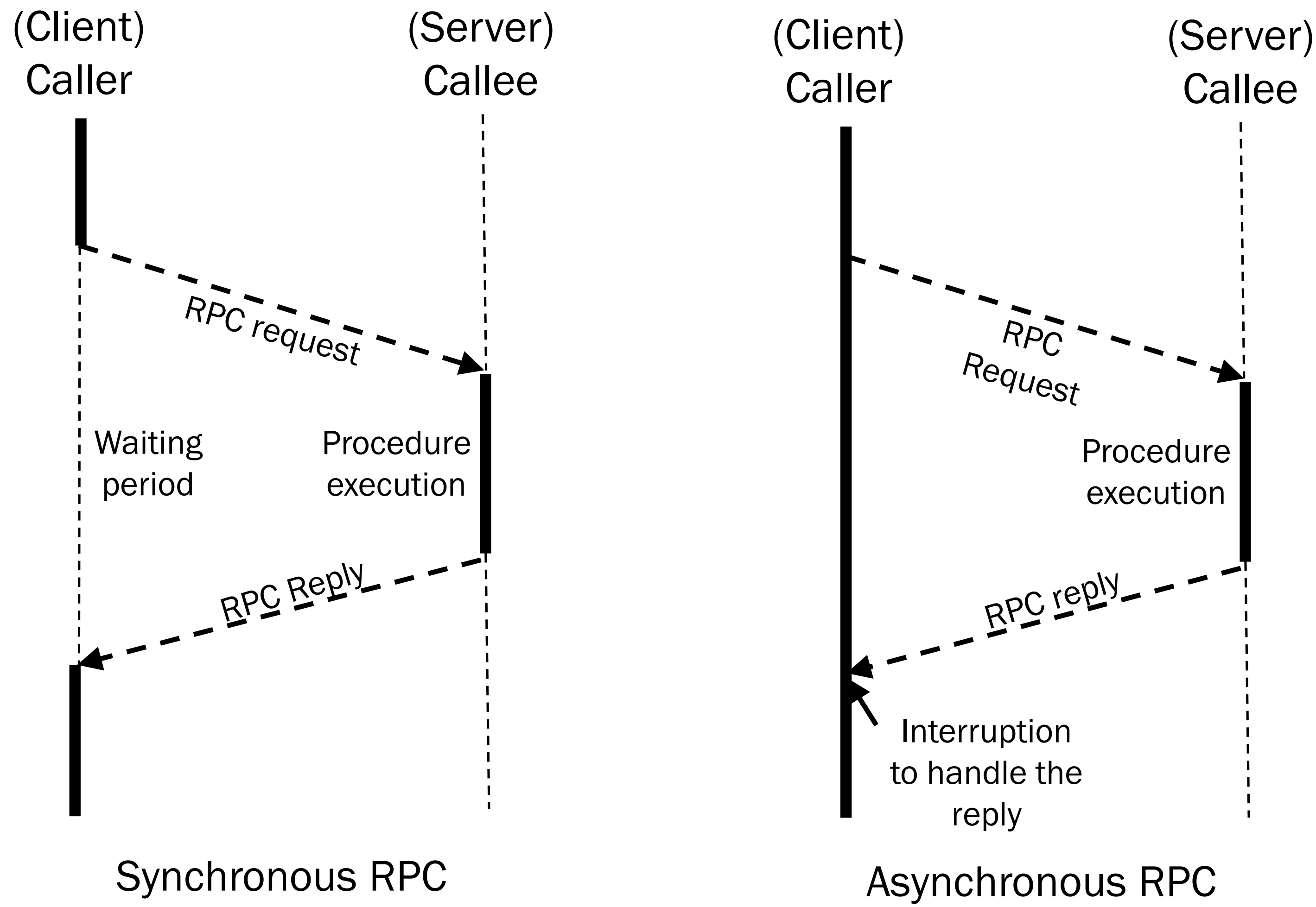
Remote Procedure Call

- Client and Server agree on procedures
  - *Names, arguments, etc.*
- Client needs to know the interface of the procedure
- Server exports an interface, client imports the interface
- Manage communication between main program and procedures
- Done by “stubs”

Basic RPC Operations

1. Client procedure calls client stub (proxy).
2. Client stub builds message and call local OS.
3. Local OS sends message to remote OS.
4. Remote OS gives message to stub.
5. Stub unpacks message and call server.
6. Server makes local call and return results to stub.
7. Stub constructs a message and call local (server) OS.
8. OS sends message to client's OS.
9. Client's OS passes the message to stub.
10. Client stubs unpacks results and return to client.

RPC



(Client) Caller
(Server) Callee
RPC request
Waiting period
Procedure execution
RPC Reply
Synchronous RPC
(Client) Caller
(Server) Callee
RPC Request
Procedure execution
RPC reply
Interruption to handle the reply
Asynchronous RPC

Authentication and Key Exchange

- The purpose of entity authentication is to prevent impersonation attack.
- Authentication is important in key exchange: the DH protocol suffers the MITM attack.
- Actually, key exchange techniques can also be used to realize authentication. (how?)
- In the literature, the differences between authentication and key exchange are not very clear. Usually, both aspects are addressed in protocol design.

A Scenario

- Users are using workstations in an open, distributed environment.
- They need to be able to access services on servers in different locations.
- There is a distributed client/server architecture.
- Servers should only serve authorised users and should be able to authenticate requests.

What is Kerberos?

- In Greek mythology, Kerberos is the guardian of Hades, a dog with three heads.

- In security community, Kerberos denotes the distributed authentication protocol developed from MIT's project Athena in 1980s.

Kerberos

- Kerberos is an example of an *Authentication and Authorisation Infrastructure (AAI)*.
- Kerberos has been widely accepted in industry.
    - *e.g., Unix systems.*
- Full specification of Kerberos Version 5 is given by RFC 4120.
- Free source code for different releases of Kerberos is available at the Kerberos website:

    http://web.mit.edu/Kerberos/

Motivations

Some threats of distributed networks

- User impersonation
  - *A dishonest user may pretend to be another user from the same workstation.*
- Network address impersonation
  - *A dishonest user may change the network address of his/her workstation to impersonate another workstation.*
- Eavesdropping, replay attack, and so on.
  - *Attackers may try their best to access network service by mounting different attacks.*

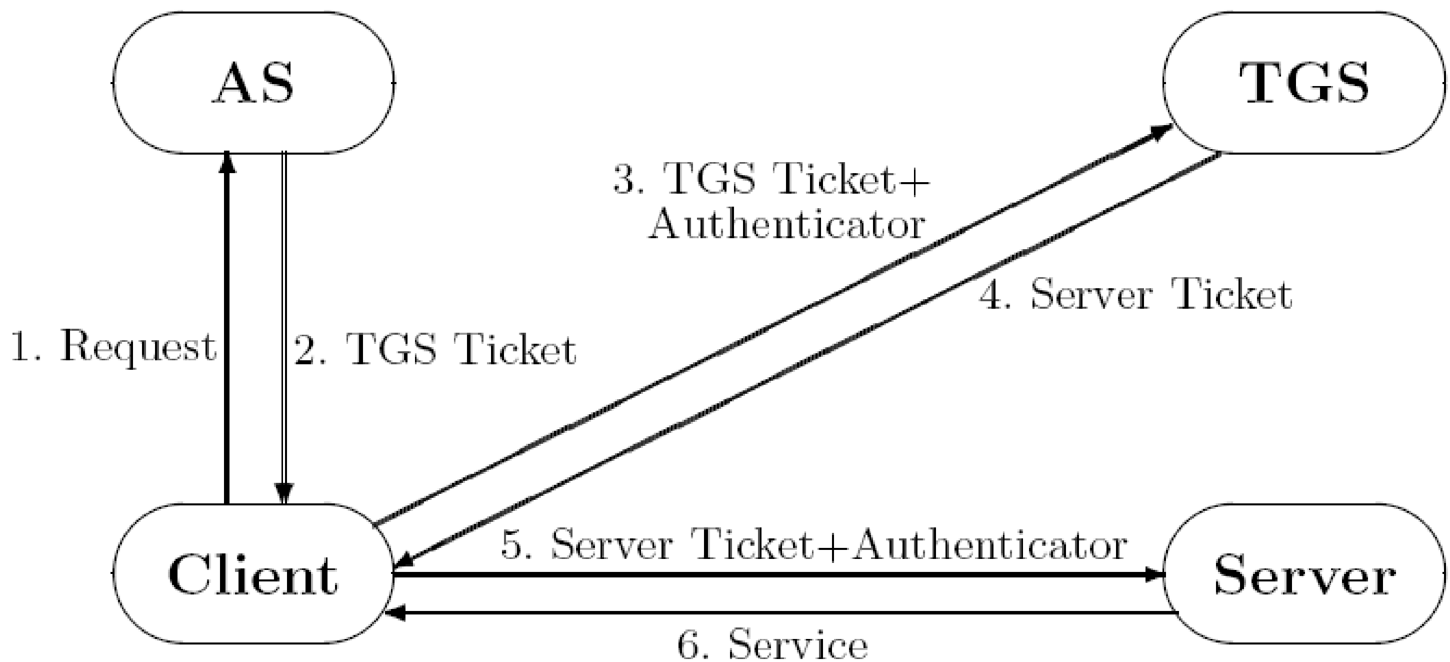
Basic Ideas

- Authentication Server (AS)
  - *A centralised trusted authentication server that issues long lifetime tickets for the whole system.*

- Ticket Granting Servers (TGS)
  - *Issue short lifetime tickets.*

- Service Servers (S)
  - *Provide different services.*

Architecture



The Protocol

Kerberos can be divided into three procedures from the view point of a client.

1. Gaining TGS ticket.
2. Gaining S ticket.
3. Gaining access to a specific service.

Kerberos (V4) Protocol

1: $C \rightarrow AS$: ID_C, ID_{tgs}, TS_1

2: $AS \rightarrow C$: $E_{K_c}[K_{c,tgs}, ID_{tgs}, TS_2, Lifetime_2, Ticket_{tgs}]$

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

3: $C \rightarrow TGS$: ID_V, $Ticket_{tgs}$, $Authenticator_C$

$Authenticator_C = E_{K_{c,tgs}}[ID_C, AD_C, TS_3]$

4: $TGS \rightarrow C$: $E_{K_{c,tgs}}[K_{C,V}, ID_V, TS_4, Ticket_V]$

$Ticket_V = E_{K_V}[K_{c,v}, ID_C, AD_C, ID_V, TS_4, Lifetime_4]$

5: $C \rightarrow V$: $Ticket_V$, $Authenticator_C$

$Ticket_V = E_{K_V}[K_{c,v}, ID_C, AD_C, ID_V, TS_4, Lifetime_4]$

$Authenticator_C = E_{K_{c,v}}[ID_C, AD_C, TS_5]$

6: $V \rightarrow C$: $E_{K_{c,v}}[TS_5 + 1]$

Protocol (V4) – Step 0

- There is actually a Step 0, which is Kerberos independent.

- Step 0: the user enters a password into a local machine (the client machine).

- From this point on the user and client machine are generally just referred to as the client.

Protocol (V4) – Step 1

C $\rightarrow$ AS: ID_C, ID_{tgs}, TS_1

- Once the user is authenticated to the Client (C), the Client sends the authentication server a request on the behalf of the user:
- This request includes a time-stamp (TS_1) and two identities:
  - *ID_C - to inform AS of the user*
  - *ID_{tgs} - to inform AS of the Ticket Granting Service required.*
  - *There may be multiple TGS's.*

Protocol (V4) – Step 2

AS $\rightarrow$ 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 session key, $K_{c,tgs}$, is generated for secure communication with the ticket granting server indicated by ID_{tgs}.
  - *A time-stamp (TS_2) is specified, as is a lifetime ($Lifetime_2$) for the ticket.*
  - *$Ticket_{tgs}$ – This is for access to TGS: It includes:*
    - The same session key, identity, time-stamp and lifetime.
    - ID_C indicating the user.
    - AD_C indicated the address of the client/user.

Protocol (V4) – Step 3

$C \rightarrow TGS: ID_V, Ticket_{tgs}, Authenticator_C$

$Authenticator_C = E_{K_{c,tgs}}[ID_C, AD_C, TS_3]$

- The client now has a ticket to communicate with a ticket granting service, and in this step it communicates with the TGS to request a service ticket.
  - *ID_V indicates the relevant server.*
  - *$Ticket_{tgs}$ is the client's permission to access the TGS.*
  - *$Authenticator_C$*
    - Only C and TGS can open it.
    - It is used by TGS to authenticate C.
    - Contains ID_C, AD_C, TS_3.

Protocol (V4) – Step 4

TGS $\rightarrow$ C: $E_{K_{c,tgs}}[K_{C,V}$, ID_V, TS_4, $Lifetime_4$, $Ticket_V]$

$Ticket_V = E_{K_V}[K_{c,v}$, ID_C, AD_C, ID_V, TS_4, $Lifetime_4]$

The TGS returns a ticket to C, granting access to server or service V.

- *The message is encrypted:*
  - Provides confidentiality and authentication.
- *A key, $K_{C,V}$, for C to talk to V.*
- *ID_V is the identity of the server*
- *There is a new time-stamp (TS_4) and a lifetime for the new ticket.*
- *The ticket itself is $Ticket_V$.*

Protocol (V4) – Step 5

$C \rightarrow V$: Ticket_V, Authenticator_C

$\text{Authenticator}_C = E_{K_{c,v}}[ID_C, AD_C, TS_5]$

- The client now communicates with V for access.
  - *Ticket_V*
  - *Authenticator_C*
    - Only C and V can open it.
    - Used by V to authenticate C.
    - Contains ID_C, AD_C, TS_5.

Protocol (V4) – Step 6

$V \rightarrow C$: $E_{K_{c,v}}[TS_5 + 1]$

- In this step the server acknowledges the message from the client.

Kerberos V4 Limitations

- Encryption: V4 uses DES only. V5 allows any encryption method.
- Restricted ticket lifetime: V4 uses an 8 bit lifetime, for a maximum of about 21 hours. V5 allows the specification of start and end times.
- Authentication forwarding: V4 does not allow credentials issued to one client to be forwarded to another host. Consider the following example of when this might be desirable: A client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials.
- Offline double encryption of the tickets in steps two and four. This is unnecessary and inefficient.
- Dictionary attack: The message from the authentication server to the client (Step 2) can be captured. A password attack against it can be launched where success occurs if the decrypted result is of an appropriate form.

Kerberos (V5)

- N: Nonce; R: Realm

1: $C \rightarrow AS$: Options, ID_C, R_C, ID_{tgs}, Times, N_1

2: $AS \rightarrow C$: R_C, ID_C, $Ticket_{tgs}$,$E_{Kc}[K_{c,tgs}$,ID_{tgs},Times, $N_1]$
$Ticket_{tgs} = E_{K_{tgs}}[Flags, K_{c,tgs}, R_C, ID_C, AD_C, TS_2, Times]$

3: $C \rightarrow TGS$: Options, ID_V,Times,N_2,$Ticket_{tgs}$, $Auth_C$
$Auth_C = E_{K_{c,tgs}}[ID_C, R_C, TS_1]$

4: $TGS \rightarrow C$: R_C, ID_C, $Ticket_V$, $E_{K_{c,tgs}}[K_{C,V}$, Times, N_2, R_V, $ID_V]$
$Ticket_V = E_{K_V}[Flags, K_{c,v}, R_C, ID_C, AD_C, Times]$

5: $C \rightarrow V$: Options, $Ticket_V$, $Auth_C$
$Auth_C = E_{K_{c,v}}[ID_C, R_C, TS_2, Subkey, Seq\#]$

6: $V \rightarrow C$: $E_{K_{c,v}}[TS_2 , Subkey, Seq\#]$

Kerberos (V5)

- **Options** provides a request for certain **flags** (indicating properties) to be set in the returned ticket, e.g.,
  - *PRE-AUTHENT: AS authenticates the client before issuing a ticket*
  - *HW-AUTHENT: Hardware based initial authentication is employed*
  - *RENEWABLE: A ticket with this flag set includes two expiration times*
    - One for this specific ticket
    - One for the latest permissible expiration time
    - A client can have a ticket renewed, if the ticket is not reported stolen
  - *FORWARDABLE: A new ticket-granting ticket with a different network address may be issued based on this ticket.*

Kerberos (V5)

- **Times** are requested by the client for ticket configuration.
  - *from: a start-time.*
  - *till: the requested expiration time.*
  - *rtime: requested renew-till time, i.e. allow continued use until.*
- **Subkey** is an optional sub-encryption key used to protect a specific session of an application. The default is the session key.
- The **Sequence number (Seq#)** is an optional sequence start number to be used by the server. It is used to protect the system from replay attacks.

Inter Realm (Domain)

A **realm** is a Kerberos server, set of clients and a set of application servers, such that:

- *The Kerberos server has the user ID's and hashed passwords of all participating users. All users are registered with the Kerberos server.*
- *The Kerberos server shares a secret key with each server, each of which is registered with the Kerberos server.*

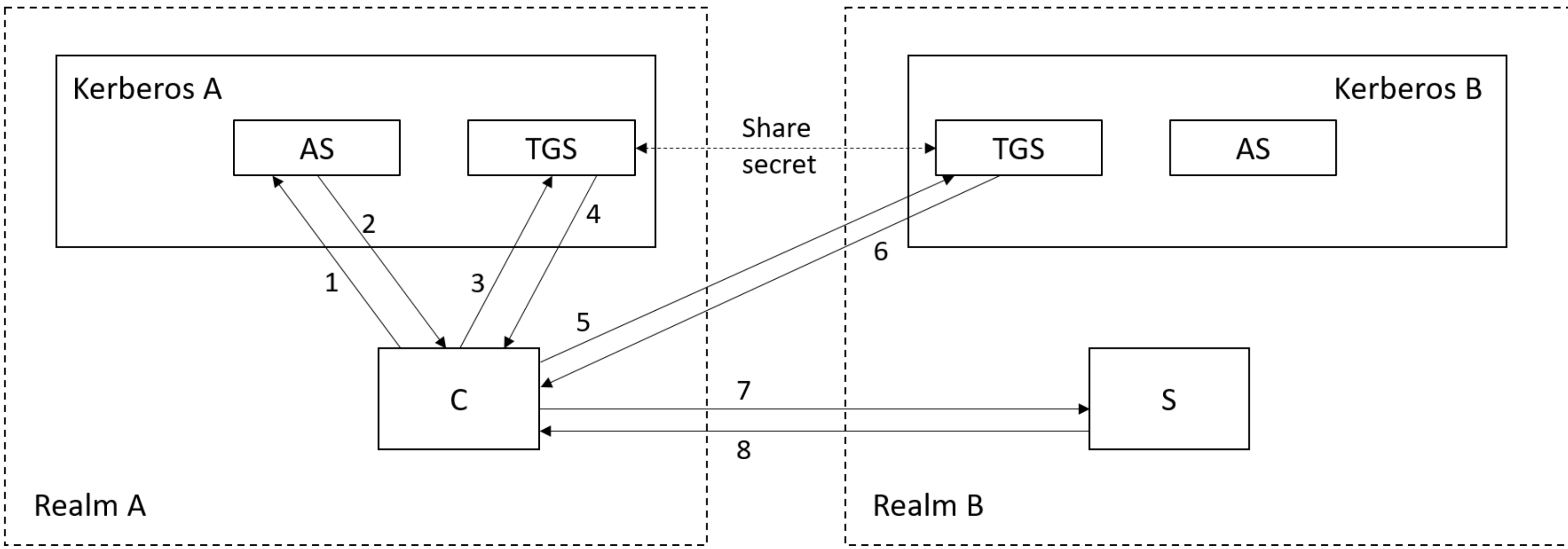
Inter Realm (Domain) Security Issues

- Authentication within a cell less complicated compare to inter-domain
- Larger Domain
  - *If compromised, more risk and more work to repair damage*
- Smaller Cell
  - *Inter-cell key management issues*

Inter Realm (Domain)

- To authenticate across realms we need another property.
  - *Each Kerberos server shares a secret key with the Kerberos servers in other realms.*
- Some changes are needed in the protocol, and some extra steps are needed too.
  - *The ticket requests now reference a service in a remote realm.*

Inter Realm Authentication



Inter Realm Protocol

1 & 2 are the same as before.

3: $C \rightarrow TGS$: ID_{TGS_r}, $Ticket_{TGS}$, $Authenticator_C$

$Authenticator_C = E_{K_{c,tgs}}[ID_C, AD_C, TS_3]$

4: $TGS \rightarrow C$: $E_{K_{c,tgs}}[K_{C,TGS_r}, ID_{TGSr}, TS_4, Lifetime_4, Ticket_{TGS_r}]$

$Ticket_{TGS_r} = E_{K_{TGS,TGSr}}[K_{c,TGS_r}, ID_C, AD_C, ID_{TGS_r}, TS_4, Lifetime_4]$

5: $C \rightarrow TGS_r$: ID_{Vr}, $Ticket_{TGSr}$, $Auth_C$

$Auth_C = E_{K_{c,tgsr}}[ID_C, AD_C, TS_5]$

$Ticket_{TGSr} = E_{K_{TGS,TGSr}}[K_{c,TGSr}, ID_C, AD_C, ID_{TGSr}, TS_4, Lifetime_4]$

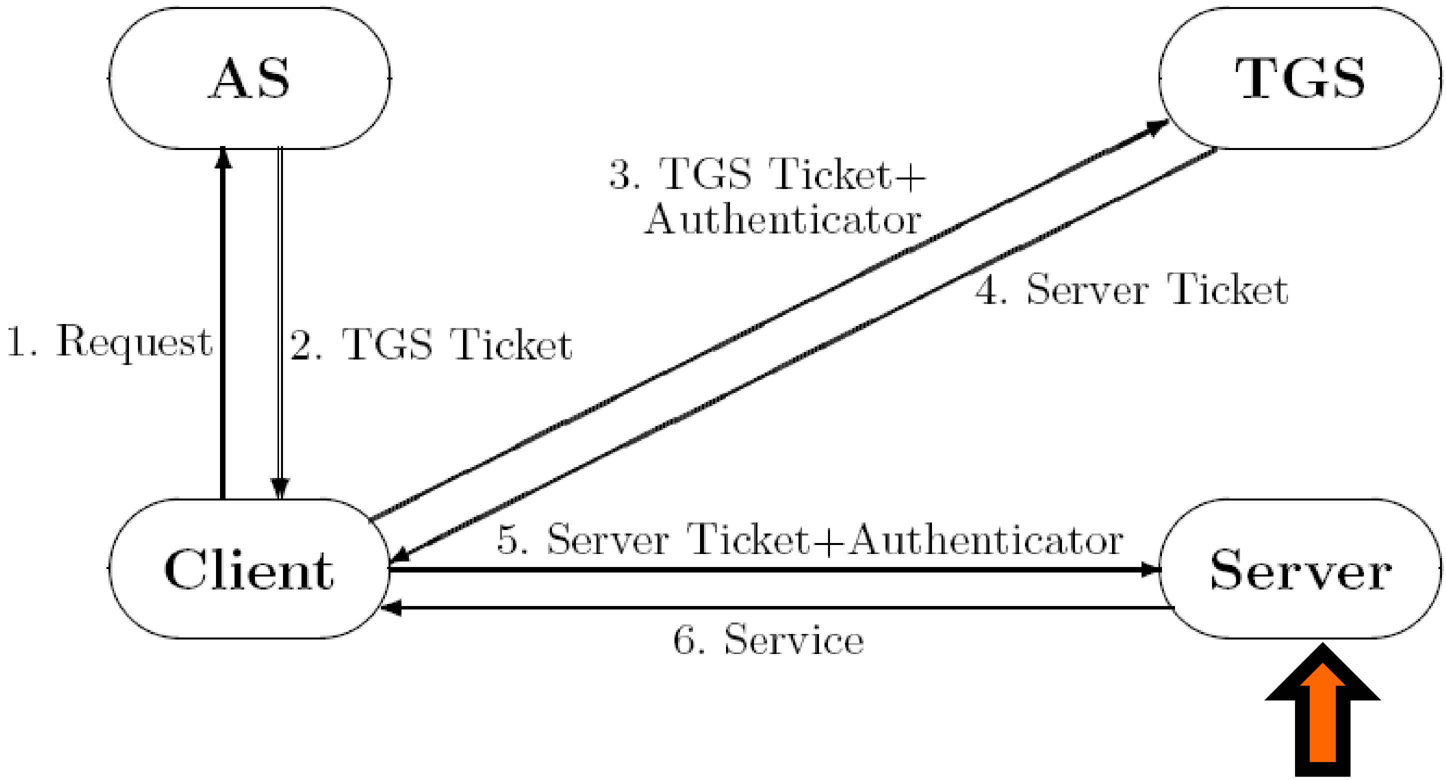
6: $TGS_r \rightarrow C$: $E_{K_{c,tgsr}}[K_{C,Vr}, ID_{Vr}, TS_6, Ticket_{Vr}]$

$Ticket_{Vr} = E_{K_{vr}}[K_{c,v}, ID_C, AD_C, ID_{Vr}, TS_6, Lifetime_6]$

7: $C \rightarrow V_r$: $Ticket_{Vr}$, $Auth_C$

$Auth_C = E_{K_{c,vr}}[ID_C, AD_C, TS_7]$

Authorisation



Summary

- Briefly introduced client/server model and remote procedure call in distributed computing environment.
- Reviewed the Kerberos authentication protocol by its:
  - *Ideas*
  - *Technical details*
  - *Limitations*

Reference

- Maarten van Steen Andrew S. Tanenbaum. “Distributed Systems: Principles and Paradigms”. CreateSpace Independent Publishing Platform, 2017.