Lecture 8
Remote User Authentication Protocols

Dr. Alshaimaa Abo-alian
A_alian@cis.asu.edu.eg

Lecture Outline

- Remote User Authentication
- Challenge-Response Authentication
- Remote User-Authentication Using Symmetric Encryption
  - Needham and Schroeder protocol
  - Denning protocol
  - Kerberos
- Remote User-Authentication Using Asymmetric Encryption

Remote User Authentication

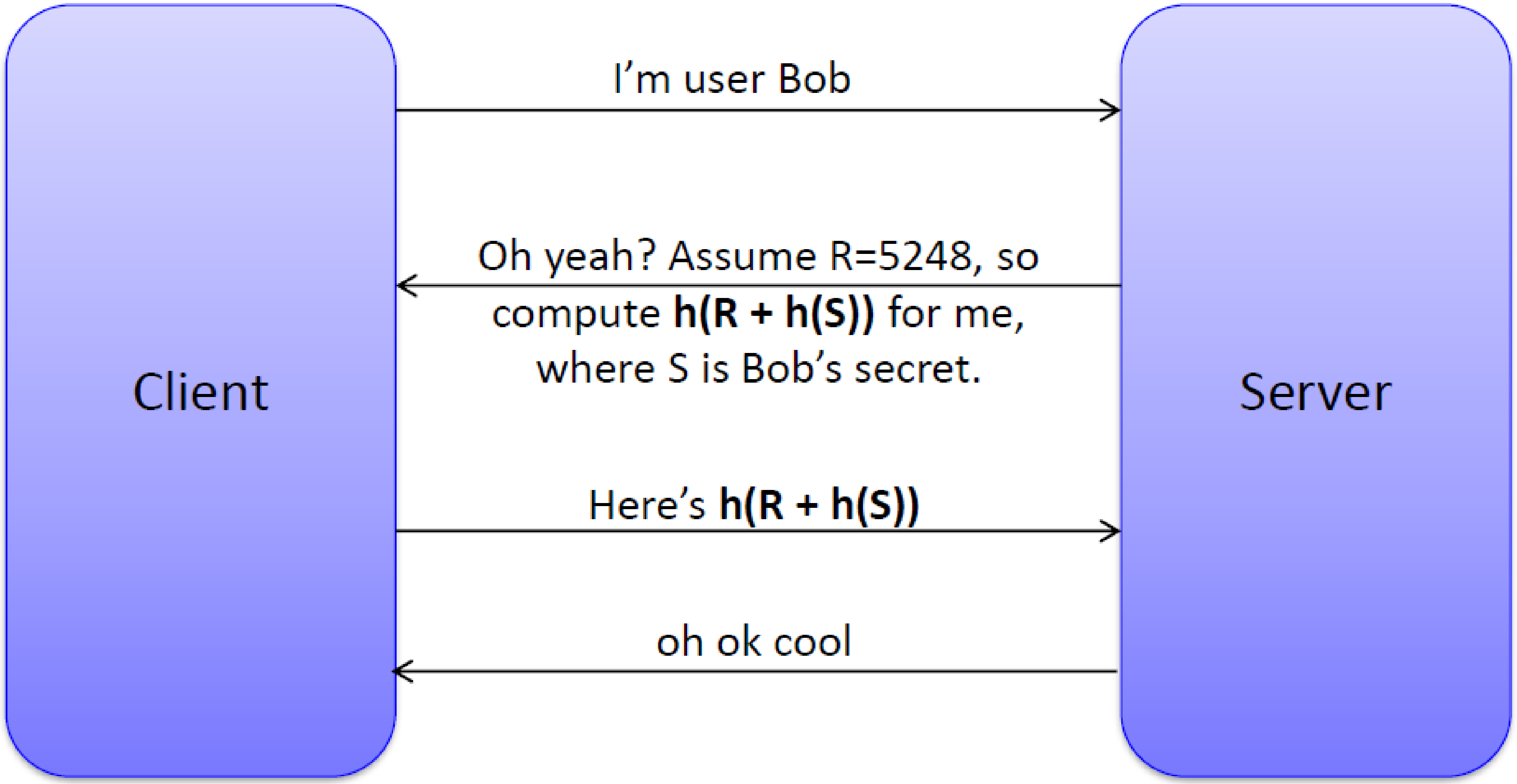
- Authentication over a network, the Internet, or a communications link is more complex
- Additional security threats such as:
  - Eavesdropping & replaying an authentication sequence that has been observed
- Rely on some form of a **challenge-response protocol** to counter threats

Challenge-Response Authentication

- The idea of challenge-response protocols is that one entity (prover) authenticates to other entity (verifier) proving the knowledge of a secret without revealing the secret itself to the verifier during the protocol.
- Assume we have some authentication secret **S**

  Such as: Password, Token value, biometric signature, etc…
- The server (verifier) issues a *challenge* (random value ***R***) to the client (prover) that can only be answered if it has **S** without revealing **S**.

Challenge-Response Authentication



Client
Server
I'm user Bob
Oh yeah? Assume R=5248, so compute **h(R + h(S))** for me, where S is Bob's secret.
Here's **h(R + h(S))**
oh ok cool

Types of Authentication

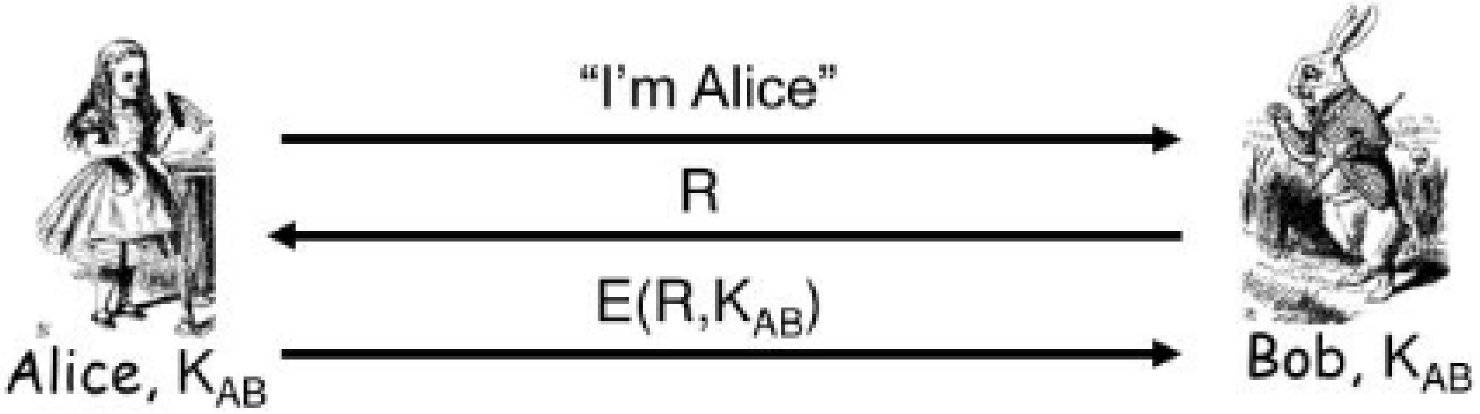
Unilateral (One-way) Authentication

Provides one entity with the assurance with other entity's identity but not vice versa

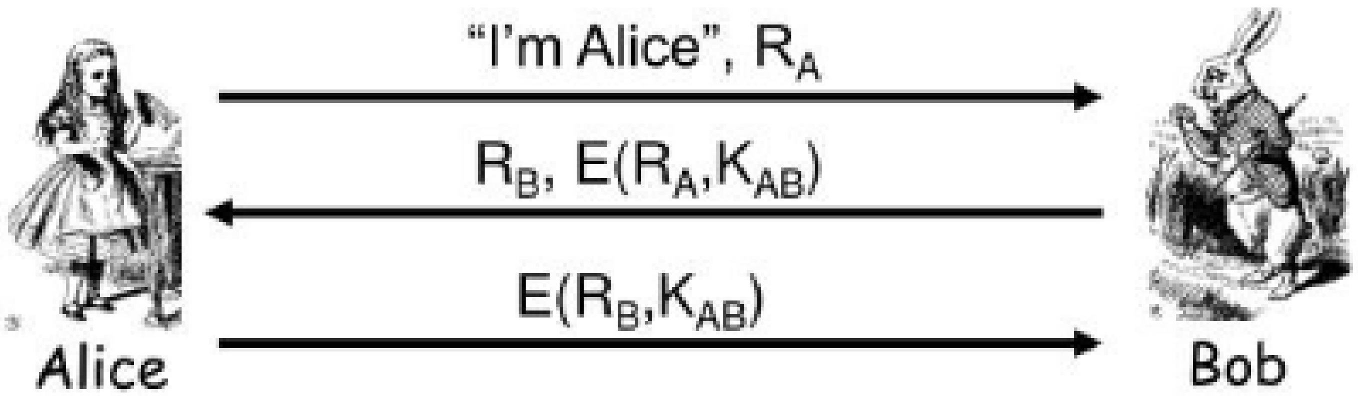
Mutual (Two-way) Authentication

Enable communicating entities to satisfy themselves mutually about each other's identity and to **exchange session keys**

One-way Authentication



Mutual Authentication



Remote Authentication Issues

The **authenticated key exchange** has two issues:

1. **Confidentiality**
   - session-key information must be communicated in encrypted form
   - This requires the prior existence of secret or public keys

2. **Timeliness**
   - The two parties must be assured to be active because of the threat of message **replays**
   - Such replays could allow an attacker to:
     - Compromise a session key
     - Successfully impersonate another party

Forms of Replay Attacks

- The main concern in authentication protocols to prevent replay attacks
- Forms of Replay attacks:
  1. The opponent simply copies a message and replays it later
  2. The opponent replays a timestamped message within the valid time window
  3. The **backward replay** without modification: It is possible if symmetric encryption is used, and the sender cannot easily differentiate between messages sent and messages received based on content

Approaches to Prevent Replay Attacks

The usage of **Challenge-response protocols** with **time-variant parameters**

- Time-variant parameters (TVP) are used to prevent replay attacks and provide uniqueness/timeliness
- There are 3 types of TVPs:

1. Sequence numbers
2. Timestamps
3. Random numbers

Approaches To Prevent Replay Attacks

1. **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
   - **The issue** 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

Approaches to Prevent Replay Attacks

2. Timestamp

- A timestamp is sequence of characters identifying when a certain event occurred, usually giving date and time of day
- Party A accepts a message as fresh only if the message contains a timestamp that is close enough to A's knowledge of current time
- This approach requires that clocks among the various participants be **synchronized**

Approaches to Prevent Replay Attacks

3. **Random numbers/nonce**
   - Party A first sends to party B a ***nonce*** (fresh random number) and requires that the subsequent message (response) received from B contain a function of that nonce.
   - A **nonce** is a value used no more than once for the same purpose. It typically serves to prevent replay.

Remote User Authentication

Remote user authentication is a mechanism in which the remote server verifies the identity of an entity over an insecure communication channel, such as the internet.

Involves the use of a trusted **key distribution center** (**KDC**)

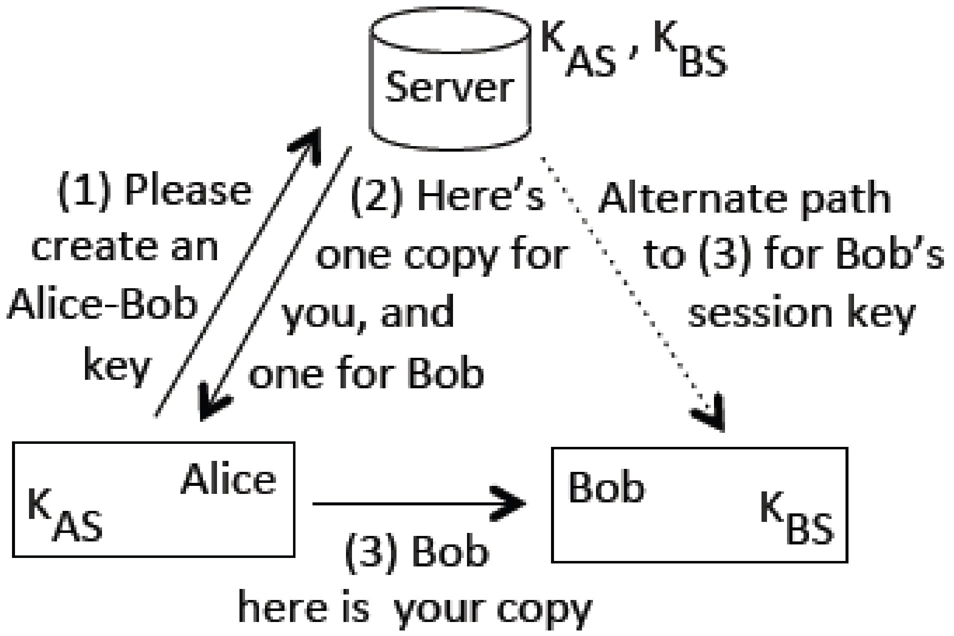
- KDC is responsible for:
  1. Generating keys to be used for a short time over a connection between two parties
  2. Distributing those keys using the master keys/public-private key pair to protect the distribution

There are 2 approaches of Remote user authentication:

1. Using Symmetric encryption
2. Using Public key encryption

Remote User-authentication Using Symmetric Encryption

- Strategy involves the use of a trusted **key distribution center** (**KDC**)
- Each party shares a secret key (a **master key**) with the KDC



Server
K_{AS}, K_{BS}
(1) Please create an Alice-Bob key
(2) Here's one copy for you, and one for Bob
Alternate path to (3) for Bob's session key
Alice
K_{AS}
Bob
K_{BS}
(3) Bob here is your copy

Remote User-authentication Using Symmetric Encryption

Needham and Schroeder protocol

Denning protocol

Kerberos

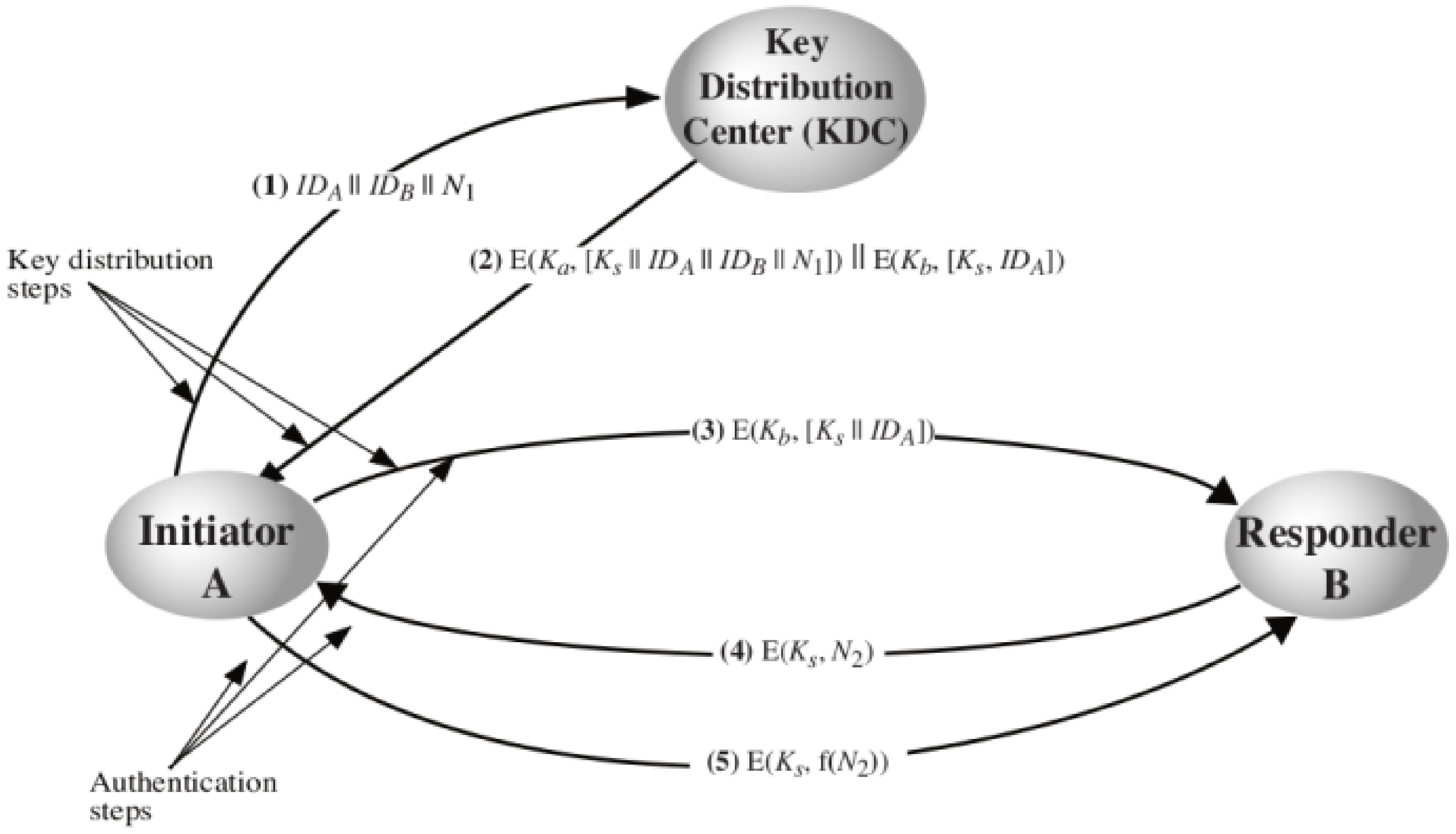
Needham and Schroeder Protocol

- The purpose of the protocol is to distribute securely a session key $\mathbf{K_s}$ to A and B.
- Secret keys $\mathbf{K_a}$ and $\mathbf{K_b}$ are shared between A and the KDC and B and the KDC, respectively.
- N_1 & N_2 are nonce

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

Secure

Needham and Schroeder Protocol

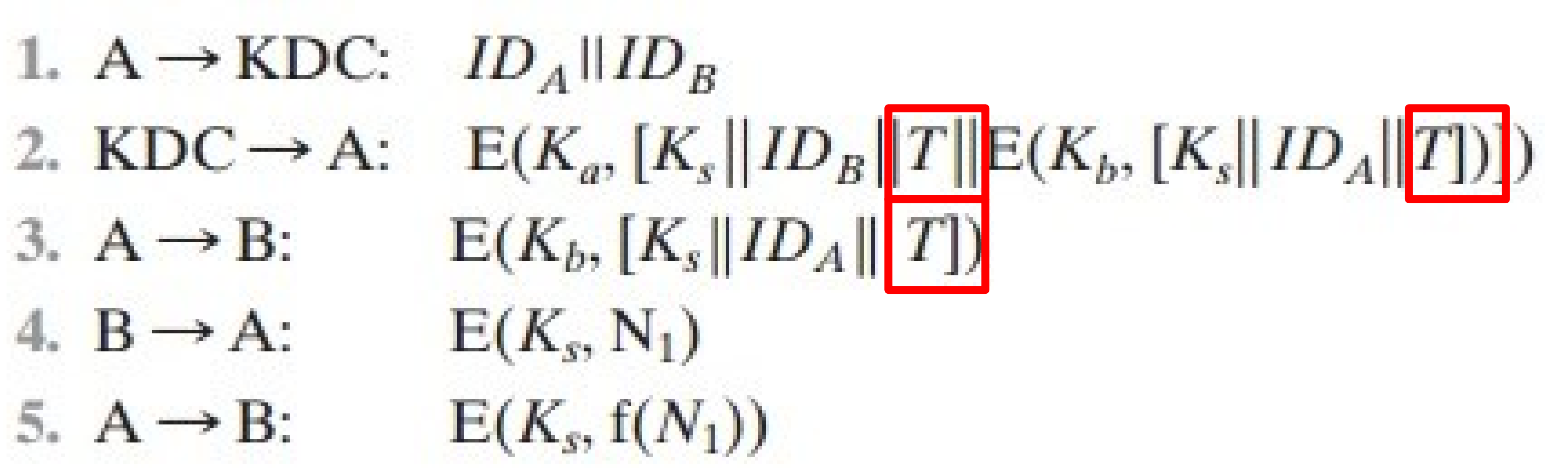

Key Distribution Center (KDC)
(1) $ID_A \| ID_B \| N_1$
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s, ID_A])$
Key distribution steps
(3) $E(K_b, [K_s \| ID_A])$
Initiator A
Responder B
(4) $E(K_s, N_2)$
(5) $E(K_s, f(N_2))$
Authentication steps

Needham and Schroeder Protocol

- The protocol is still vulnerable to **replay attack**.
- Suppose an opponent, X can compromise an old session key
- X can impersonate A and trick B into using the old key by simply replaying step 3.
- Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay.
- Then X can impersonate A's response in step 5 and send any messages to B that appear to B to come from A using an authenticated session key.

Denning Protocol

- Proposes to overcome this weakness of the Needham/Schroeder protocol by addition of a timestamp to steps 2 and 3.



1. A → KDC: $ID_A \| ID_B$
2. KDC → A: $E(K_a, [K_s \| ID_B \| \boxed{T} \| E(K_b, [K_s \| ID_A \| \boxed{T}])])$
3. A → B: $E(K_b, [K_s \| ID_A \| \boxed{T}])$
4. B → A: $E(K_s, N_1)$
5. A → B: $E(K_s, f(N_1))$

Denning Protocol

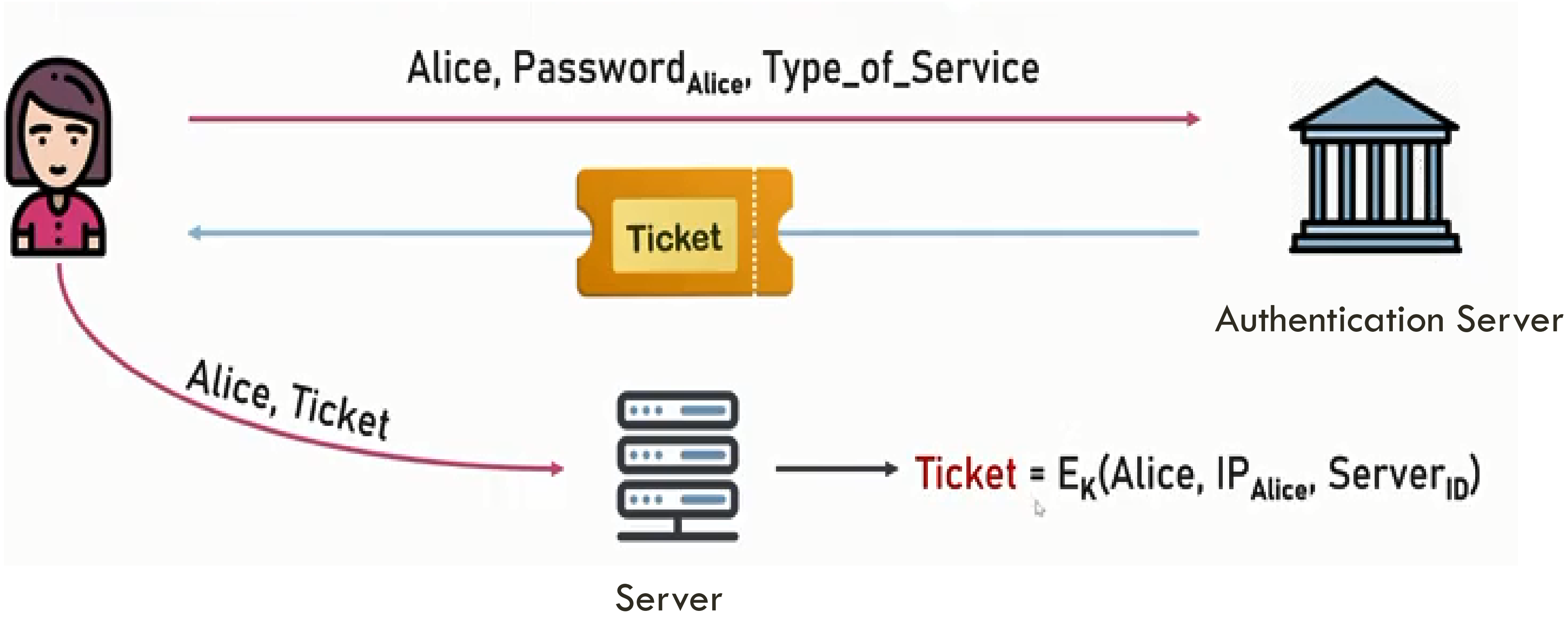
- This new scheme requires clocks to be synchronized throughout the network
- The distributed clocks can become unsynchronized due to faults in the clocks or the synchronization mechanism.
- The problem occurs when a sender's clock is ahead of the intended recipient's clock.
- In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site.

➔ Such attacks is called **suppress-replay attacks.**

Kerberos

- Authentication service developed as part of Project Athena at MIT
- Supports both user authentication and key establishment using **symmetric techniques** and a third party.
- Kerberos assumes a **distributed client/server architecture** and employs provides a centralized **authentication server** whose function is to authenticate users to servers and servers to users.
- Makes use of DES to provide the authentication service
- The basic Kerberos protocol involves 3 parties:
  - C (the client)
  - V (the server and verifier)
  - Trusted server AS (the Kerberos authentication server).

A Simple Kerberos Authentication Dialogue


Alice, Password$_{Alice}$, Type_of_Service
Ticket
Authentication Server
Alice, Ticket
Ticket = E_K(Alice, IP$_{Alice}$, Server$_{ID}$)
Server

A Simple Kerberos Authentication Dialogue

$$\begin{aligned}
&(1)\ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \,\|P_C\|\, ID_V \\
&(2)\ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&(3)\ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \,\|\, Ticket
\end{aligned}$$

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

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

A Simple Kerberos Authentication Dialogue

- At the beginning, C and V share no secret
- AS shares a secret with each user (e.g., a user password transformed into a cryptographic key by an appropriate function).
- AS shares a unique secret key K_v with each server *v*.
- The protocol proceeds as follows
  - C requests from AS appropriate credentials to allow it to authenticate itself to V.
  - AS plays the role of a KDC, returning to C a **ticket** encrypted for V.
  - The ticket, which C forwards on to V, contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server
  - Because the ticket is encrypted, it cannot be altered by C or by an opponent

Kerberos - Ticket Granting Server (TGS)

Although the foregoing scenario solves some of the problems of authentication in an open network environment, it needs some improvements:

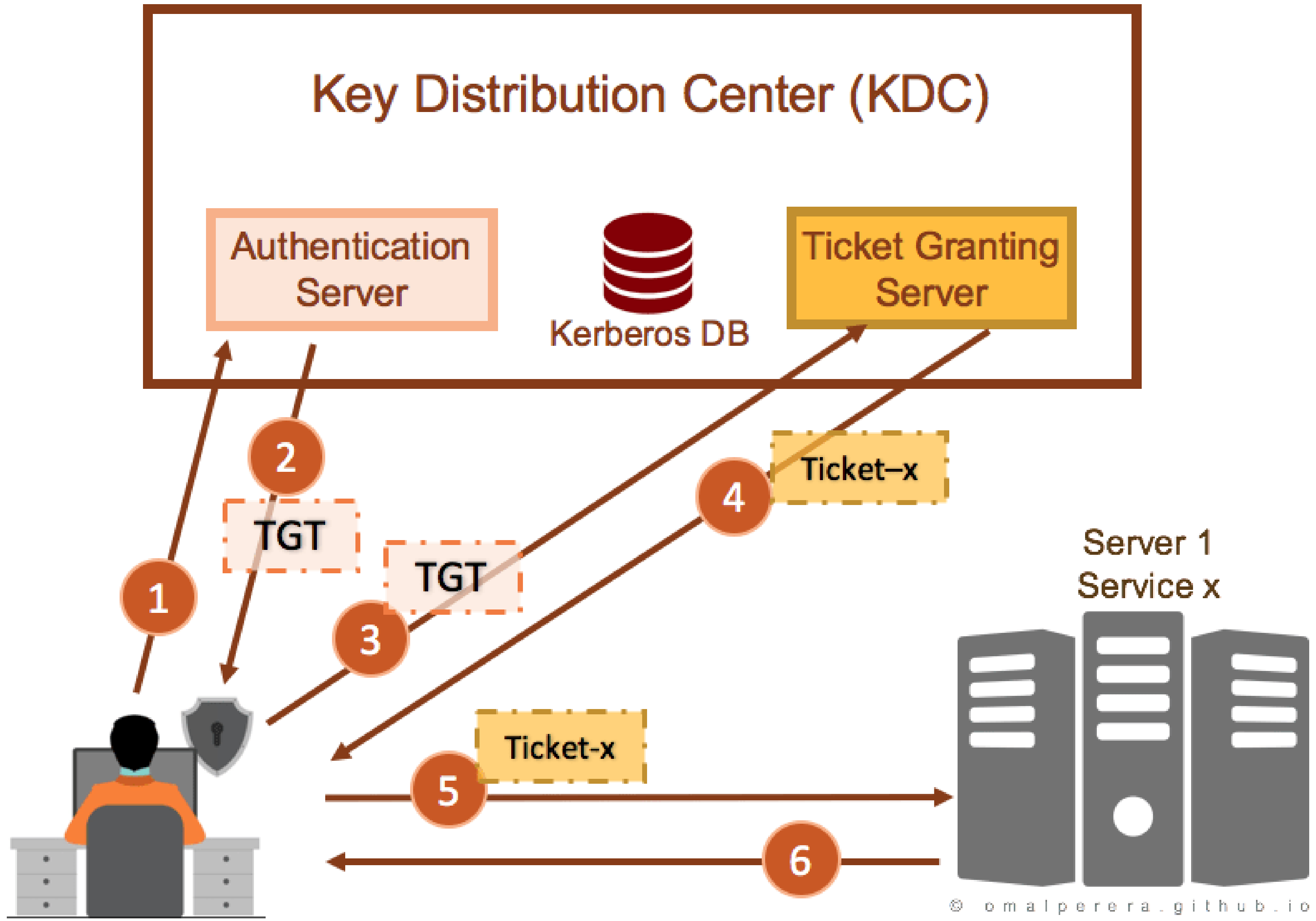
1. Making tickets are **reusable** to minimize the number of times that a user has to enter a password.

   Example: For a single logon session, the workstation can store the mail server ticket after it is received and use it on behalf of the user for multiple accesses to the mail server.

2. Making new ticket for every different service. e.g. print server, a mail server, a file server, and so on

Kerberos - Ticket Granting Server (TGS)

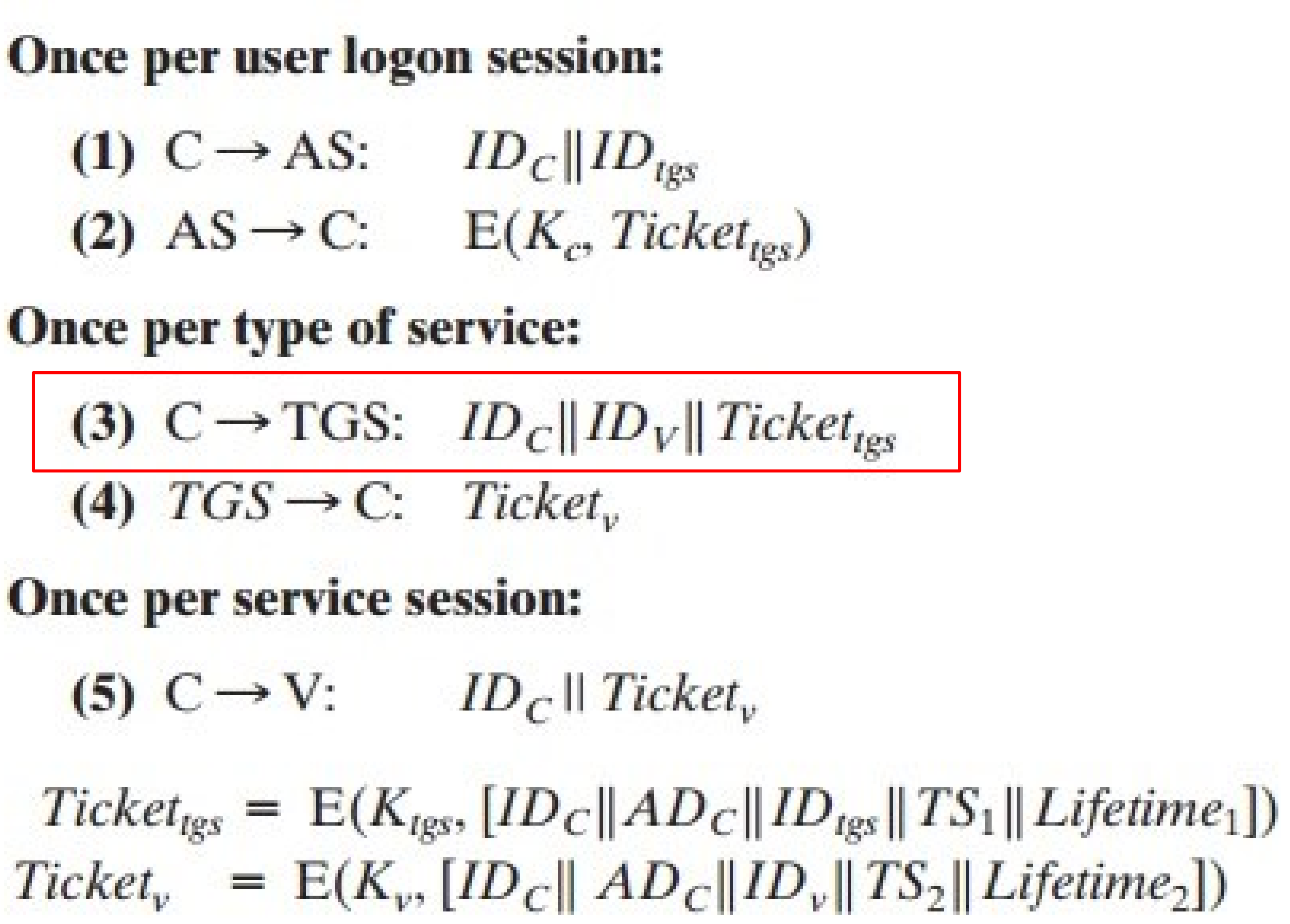
- To solve these issues, a new **ticket-granting server (TGS)** is introduced.
- The user first requests a ticket-granting ticket (**Ticket**$_{\textbf{tgs}}$) from the AS.
- The client module in the user workstation saves this ticket.
- Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself.
- The TGS then grants a ticket for the particular service.


Key Distribution Center (KDC)
Authentication Server
Kerberos DB
Ticket Granting Server
1
2
TGT
3
TGT
4
Ticket–x
Server 1
Service x
5
Ticket-x
6
© omalperera.github.io

Kerberos - Ticket Granting Server (TGS)

- **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-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
- **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

Kerberos - Ticket Granting Server (TGS)



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:

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

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

Once per service session:

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

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

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

A More Secure Authentication Dialogue

There are **two issues** with this scenario:

1. 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

   ➔ That is why we use **authenticators**

2. Servers need to authenticate themselves to users

Kerberos Version 4

(1) $\mathbf{C \rightarrow AS}$ $\quad ID_c \| ID_{tgs} \| TS_1$

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

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

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

(3) $\mathbf{C \rightarrow TGS}$ $\quad ID_v \| Ticket_{tgs} \| Authenticator_c$

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

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

$$Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4])$$

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

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

(5) $\mathbf{C \rightarrow V}$ $\quad Ticket_v \| Authenticator_c$

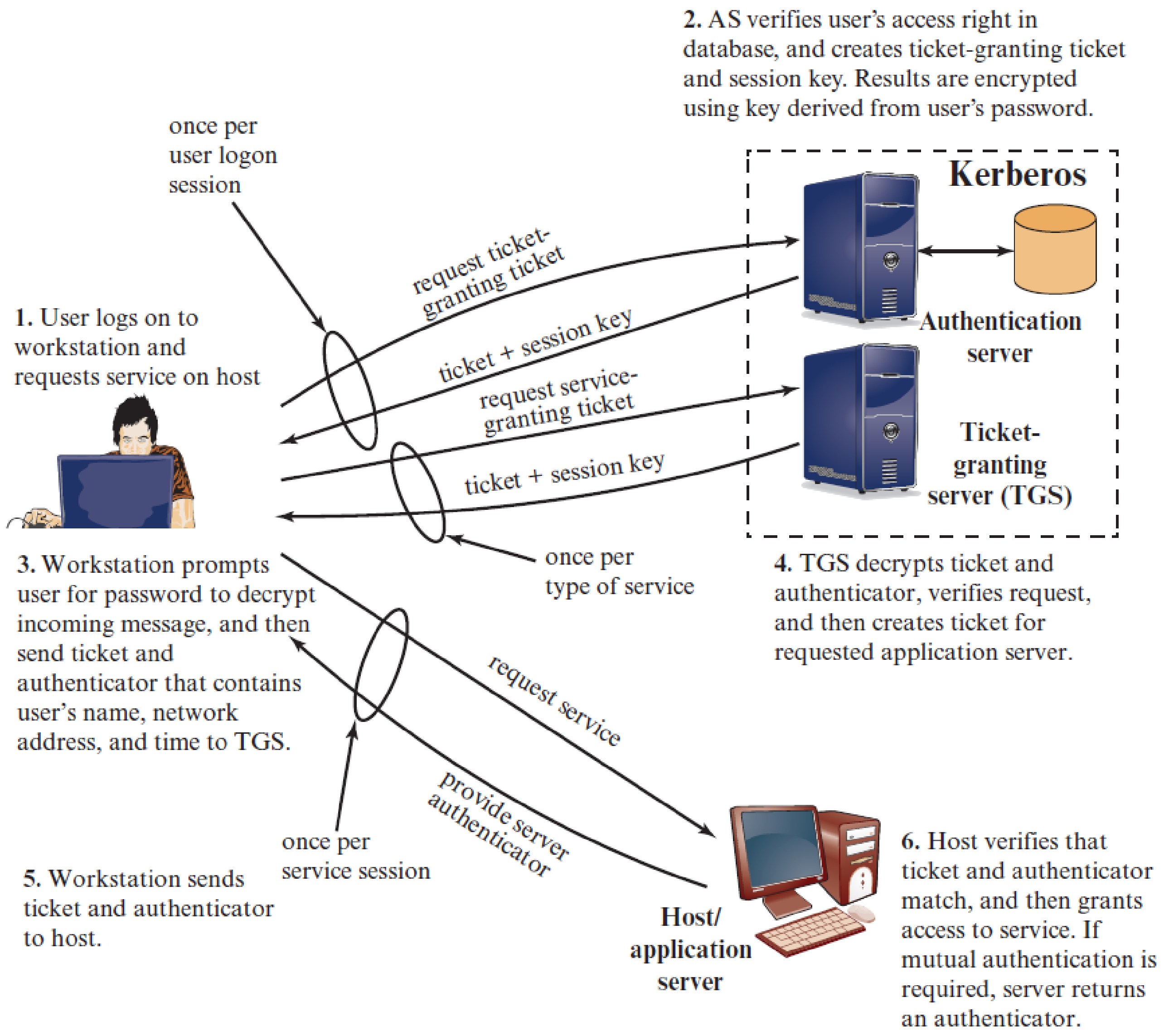
(6) $\mathbf{V \rightarrow C}$ $\quad \mathrm{E}(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

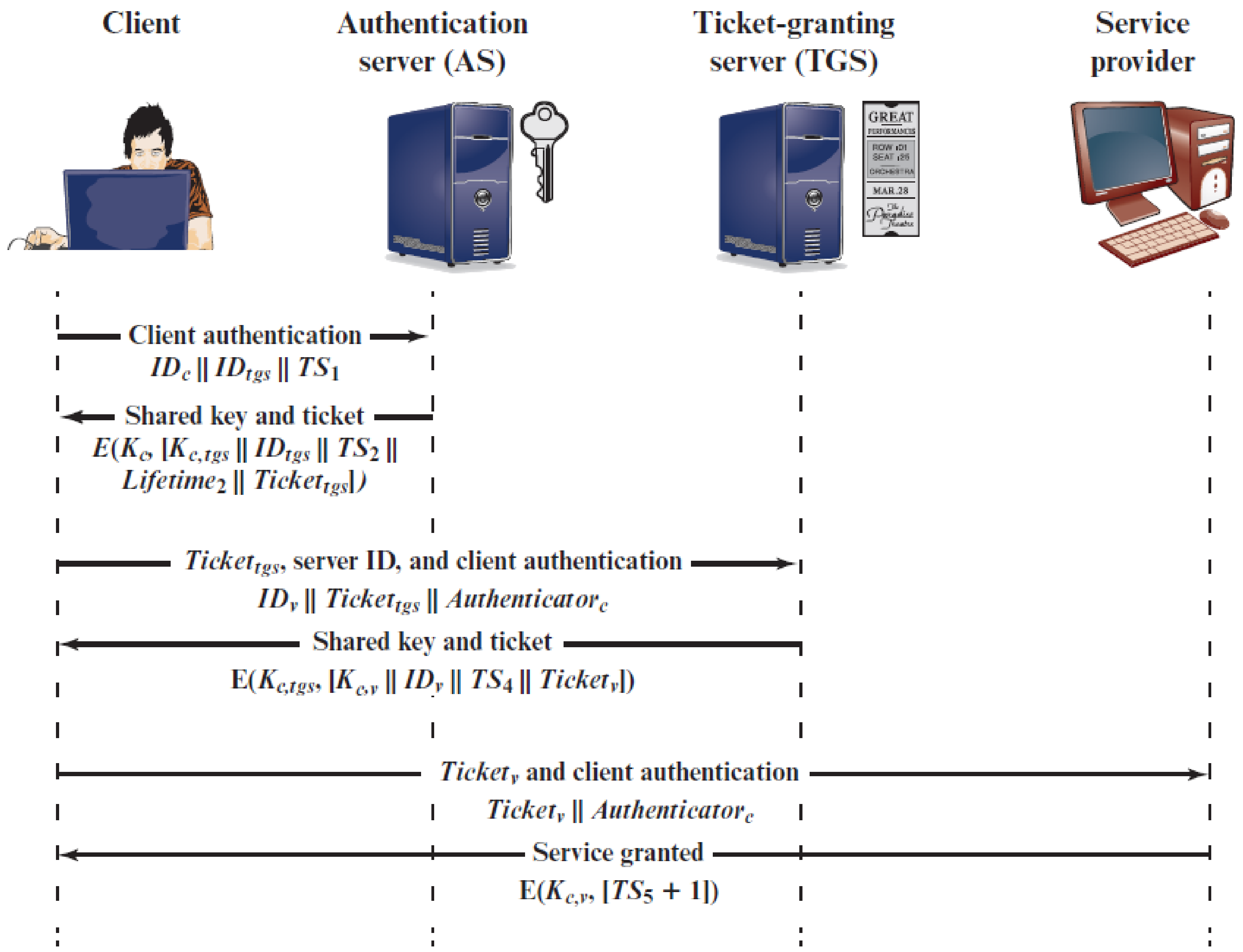
$$Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(\mathrm{K}_{c,v}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{TS}_5])$$

(c) Client/Server Authentication Exchange to obtain service

Overview of Kerberos


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.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
Authentication server
1. User logs on to workstation and requests service on host
request service-granting ticket
ticket + session key
Ticket-granting server (TGS)
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.


Client
Authentication server (AS)
Ticket-granting server (TGS)
Service provider
GREAT PERFORMANCES
ROW D1 SEAT 25
ORCHESTRA
MAR.28
The Paradise Theatre
Client authentication
$ID_c \| ID_{tgs} \| TS_1$
Shared key and ticket
$E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$
$Ticket_{tgs}$, server ID, and client authentication
$ID_v \| Ticket_{tgs} \| Authenticator_c$
Shared key and ticket
$E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$
$Ticket_v$ and client authentication
$Ticket_v \| Authenticator_c$
Service granted
$E(K_{c,v}, [TS_5 + 1])$

KERBEROS VERSION 4

- Makes use of DES to provide the authentication service
- To ensure that the ticket presenter is the same as the client for whom the ticket was issued, the AS provide both the client and the TGS with a **secret session key $K_{c,\ tgs}$** in a secure manner (i.e. encrypted) as in step 2
  - ➢ The same session key is included in the ticket_{tgs}, which can be read only by the TGS.
  - ➢ In effect, the ticket says, “**Anyone who uses Kc,tgs, must be C.**”
- The client C transmits an **authenticator,** which includes the ID and network address of C’s user and a timestamp
  - ➢ the authenticator is intended for use only once and has a very short lifetime.
  - ➢ In effect, the authenticator says, “**At time TS_3, I hereby use $K_{c,tgs}$**”

User Authentication Using Asymmetric Encryption

Assumes that the prover (user) and the verifier (server) is in possession of the current public key of the other.

Protocols using timestamp

Protocols using nonce

User authentication Using Asymmetric Encryption

1. Denning protocol using timestamps

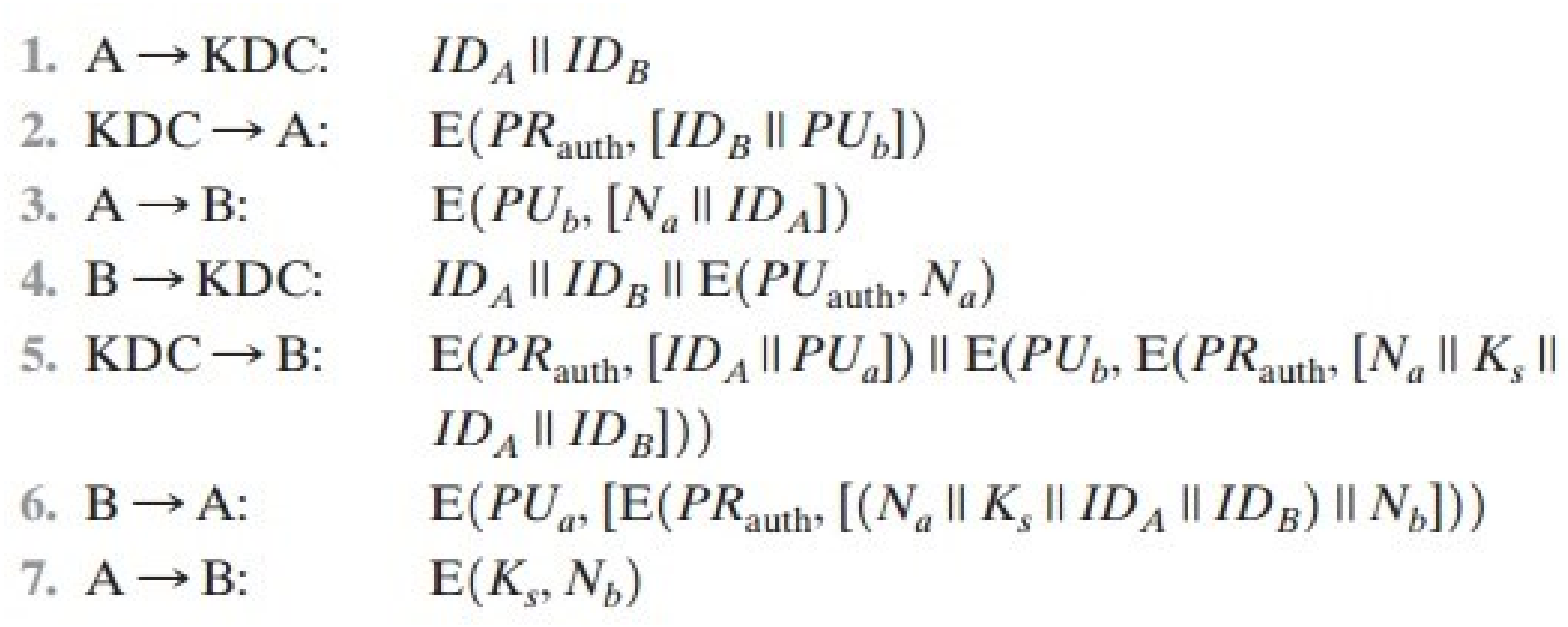
- Uses an authentication server (AS) to provide public-key certificates
- AS is not actually responsible for secret-key distribution
- Requires the synchronization of clocks

1. $A \rightarrow AS$: $ID_A \| ID_B$
2. $AS \rightarrow A$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
3. $A \rightarrow B$: $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \|$ $E(PU_b, E(PR_a, \underline{[K_s \| T]}))$

The session key is chosen and encrypted by A

User-authentication Using Asymmetric Encryption

2. Woo and Lam Protocol using nonce



1. A → KDC:	$ID_A \parallel ID_B$
2. KDC → A:	$E(PR_{auth}, [ID_B \parallel PU_b])$
3. A → B:	$E(PU_b, [N_a \parallel ID_A])$
4. B → KDC:	$ID_A \parallel ID_B \parallel E(PU_{auth}, N_a)$
5. KDC → B:	$E(PR_{auth}, [ID_A \parallel PU_a]) \parallel E(PU_b, E(PR_{auth}, [N_a \parallel K_s \parallel ID_A \parallel ID_B]))$
6. B → A:	$E(PU_a, [E(PR_{auth}, [(N_a \parallel K_s \parallel ID_A \parallel ID_B) \parallel N_b]))$
7. A → B:	$E(K_s, N_b)$

LECTURE REFERENCES

"Computer Security: Principles and Practice", 4th edition, by William Stallings and Lawrie Brown

Chapter 23 "Internet Authentication Applications".

"Cryptography and Network Security: Principles and Practice", 8th Edition, by William Stallings

Chapter 16 "User Authentication".

Thank you