中山大學
SUN YAT-SEN UNIVERSITY

Module I. Fundamentals of Information Security

Chapter 3

Authentication Technologies

Information Security: Theory & Applications

School of Data & Computer Science, Sun Yat-sen University

Outline

- **3.1 Overview**
  - Introduction to Authentication Technologies
  - The Weak/Strong Authentication Scheme
  - The Application of Authentication Technologies
  - The Attack to Authentication
  - The Security Guidelines to Protect Authentication Schemes
- **3.2 Public Key Infrastructure**
  - Introduction to PKI
  - PKIX
  - The Management of PKIX
  - Public Key Certificate
  - Trust Hierarchy Model

Outline

- **3.3 Kerberos**
  - Introduction
  - Kerberos Process
  - Drawbacks & Limitations
- **3.4 X.509**
  - What is X.509
  - Certificate
  - Security problems
  - Application

3.3 Kerberos

3.3.1 Introduction

- **What is Kerberos**
  - *Kerberos* (ITU-T) is a computer network authentication protocol which works on the basis of “tickets” (票据) to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.
    - ✧ Its designers aimed primarily at a client–server model, and it provides mutual authentication - both the user and the server verify each other's identity.
    - ✧ Kerberos protocol messages are protected against eavesdropping and replay attacks.

3.3 Kerberos

3.3.1 Introduction

- **What is Kerberos**
  - *Kerberos* is based on *Needham* & *Schroeder*'s protocol, and as part of MIT's *Athena* project.
  - Kerberos builds on **symmetric key cryptography** and requires a trusted third party, and optionally may use public-key cryptography by utilizing **asymmetric key cryptography** during certain phases of authentication.
  - Kerberos was named by Greek mythological character, the watching dog of the gate of Hell with three heads representing the three A's functions:
    - ✧ Authentication (complete)
    - ✧ Accounting (not complete)
    - ✧ Audit (not complete)

3.3 Kerberos

3.3.1 Introduction

- **What is Kerberos**
  - Free Software
  - Used widely in different platforms
  - Version 1-3: occurred internally at MIT
  - Version 4: designed by *Steve Miller* and *Clifford Neuman,* published in the late 1980s, also targeted at Project Athena.
  - Version 5: published by *Clifford Neuman* and *John Kohl* in 1993, appeared as RFC 1510, and obsoleted by RFC4120 in 2005.

3.3 Kerberos

3.3.1 Introduction

- ***Needham-Schroeder* Symmetric Key Protocol**
  - The *Needham–Schroeder* Symmetric Key Protocol
    - ✧ (*Roger Needham* and *Michael Schroeder*, 1978.) It aims to establish a session key between two parties on a network, typically to protect further communication. It is based on a symmetric encryption algorithm and forms the basis for the Kerberos protocol.
  - Scenario: *Alice* wishes to communicate with *Bob*.
    - ✧ TA is a trusted third party (KDC, Key Distributing Centre) that provides session keys (e.g., K_{AB} between *Alice* and *Bob*).
    - ✧ E(K, -) is a Symmetric Key Cryptographic Algorithm (e.g., DES).
    - ✧ TA has a key K_{AT} in common with *Alice* and a key K_{BT} with *Bob*.
    - ✧ *Alice* authenticates TA using a nonce r_A and obtains a session key K_{AB} from TA.
    - ✧ *Alice* authenticates to *Bob* (*Alice* 得到 *Bob* 的认证) and transports the session key securely.

3.3 Kerberos

3.3.1 Introduction

- ***Needham-Schroeder* Symmetric Key Protocol**
  - The *Needham-Schroeder* symmetric key protocol
    1. *Alice* $\rightarrow$ TA : *Alice*, *Bob*, r_A
    2. TA $\rightarrow$ *Alice* : $E(K_{AT}, (K_{AB}, r_A, Bob, E(K_{BT}, (K_{AB}, Alice))))$

       Alice decrypts with K_{AT} and checks r_A and "*Bob*", holds K_{AB} for future correspondence with *Bob.*
    3. *Alice* $\rightarrow$ *Bob* : $E(K_{BT}, (K_{AB}, Alice))$

       Bob decrypts with K_{BT} .
    4. *Bob* $\rightarrow$ *Alice* : $E(K_{AB}, r_B)$

       Alice decrypts with K_{AB} .
    5. *Alice* $\rightarrow$ *Bob* : $E(K_{AB}, r_B\text{-}1)$

       Bob checks r_B-1 .
  - Attack on the protocol
    - ✧ Suppose that *Darth* stole an old K_{AB} and replay step 3. *Bob* will accept it, being unable to tell that the key is not fresh.

3.3 Kerberos

3.3.1 Introduction

- ***Needham-Schroeder* Symmetric Key Protocol**
  - An improved *Needham-Schroeder* symmetric key protocol
    1. $Alice \to Bob : Alice$
    2. $Bob \to Alice$: E(K_{BT}, $Alice$, r_B^0)
    3. $Alice \to$ TA : $Alice$, Bob, r_A, E(K_{BT}, $Alice$, r_B^0)
    4. TA $\to Alice$: E(K_{AT}, (K_{AB}, r_A, Bob, E(K_{BT}, (K_{AB}, $Alice$, r_B^0))))
    5. $Alice \to Bob$: E(K_{BT}, (K_{AB}, $Alice$, r_B^0))

       Bob decrypts with K_{BT} and checks r_B^0
    6. $Bob \to Alice$: E(K_{AB}, r_B)

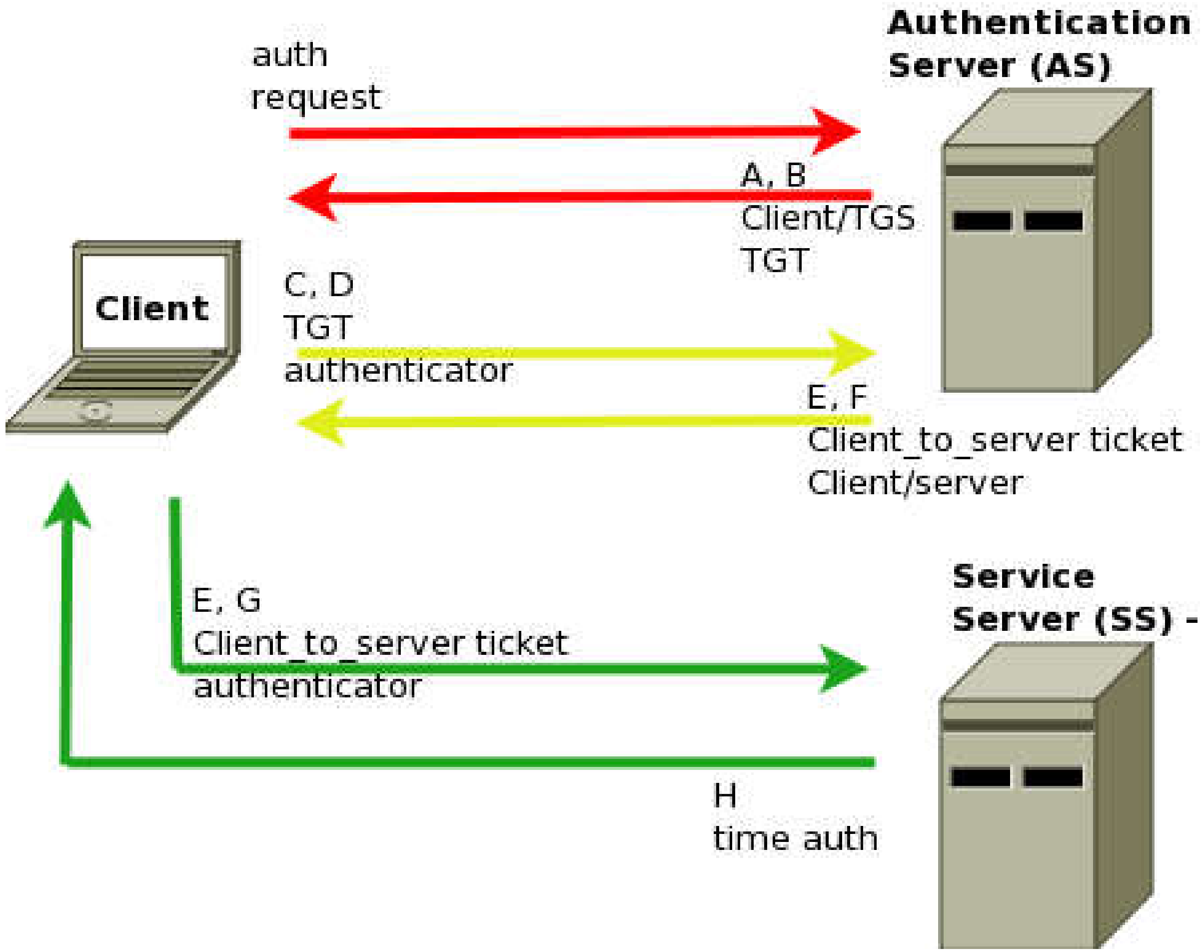
       Alice decrypts with K_{AB}
    7. $Alice \to Bob$: E(K_{AB}, r_B-1)

       Bob checks r_B-1
  - The improved process fixed the attack on the protocol: the session key K_{AB} is bound with r_B^0 . *Bob* can recognize that the key is fresh or not in step 5.

3.3 Kerberos

3.3.1 Introduction

- **Description**



3.3 Kerberos

3.3.1 Introduction

- **Description**
  - The client authenticates to the AS once using a long-term shared secret (e.g. a password) and receives a TGT from the AS.
  - Later, when the client wants to contact some SS, it can (re)use this ticket to get additional tickets from TGS for SS, without resorting to using the shared secret. These tickets can be used to prove authentication to SS.

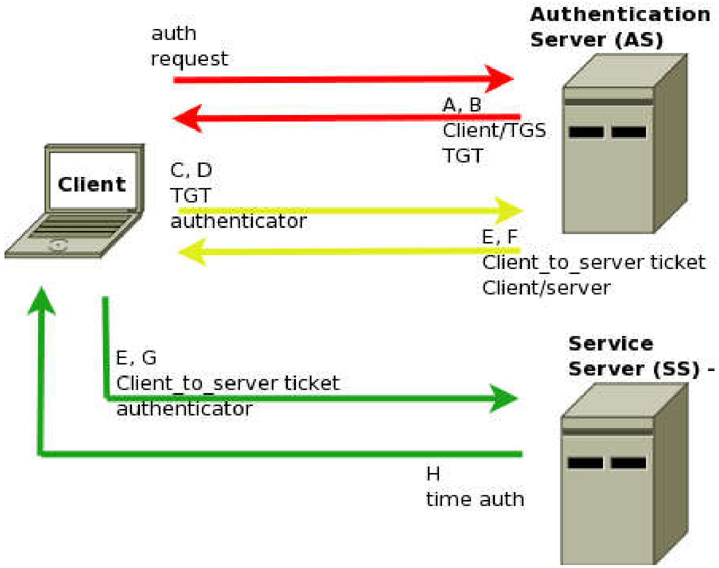
✧ AS = Authentication Server

✧ SS = Service Server

✧ TGS = Ticket-Granting Server

✧ TGT = Ticket-Granting Ticket

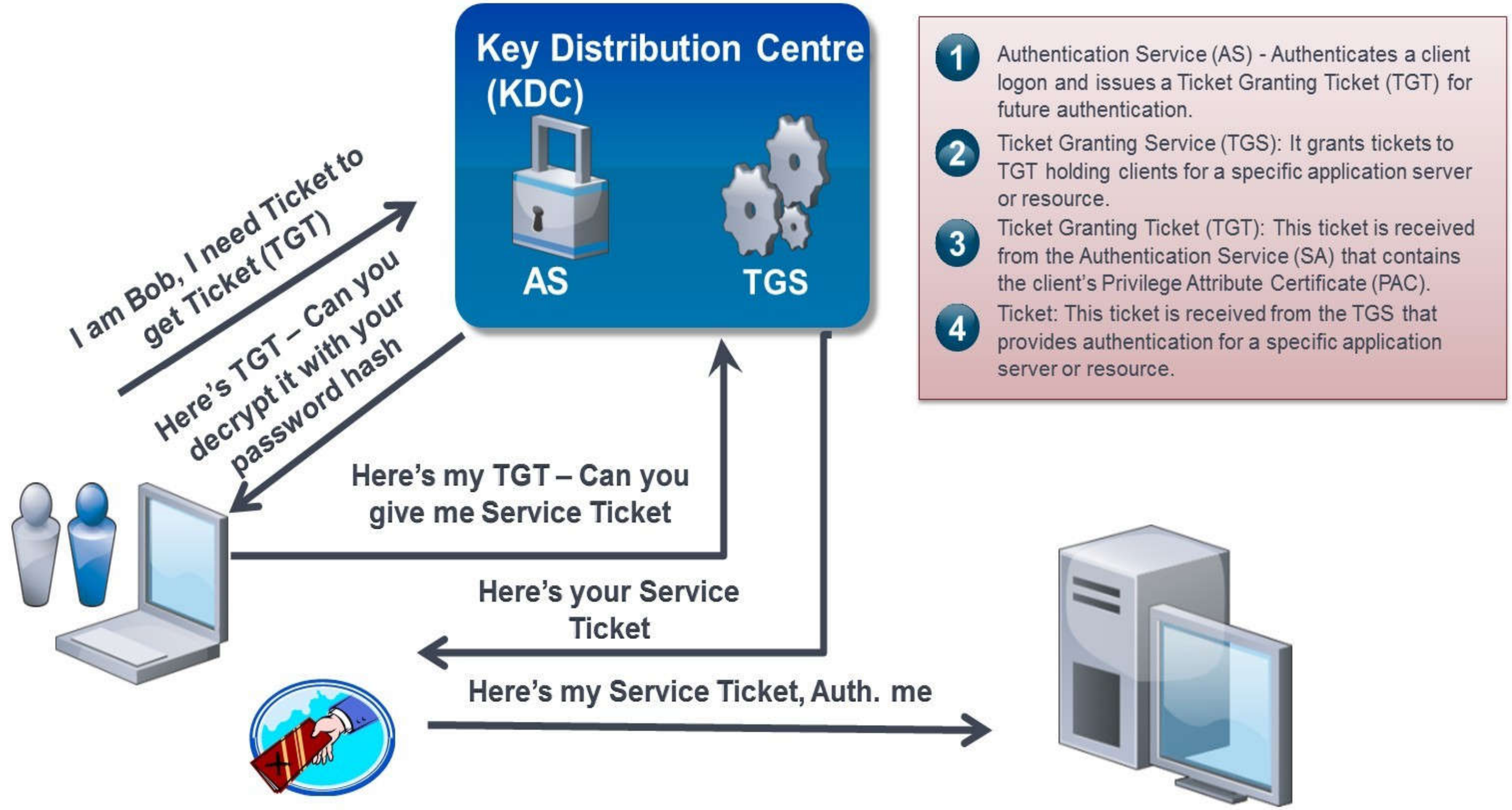
✧ ST = Service Ticket



3.3 Kerberos

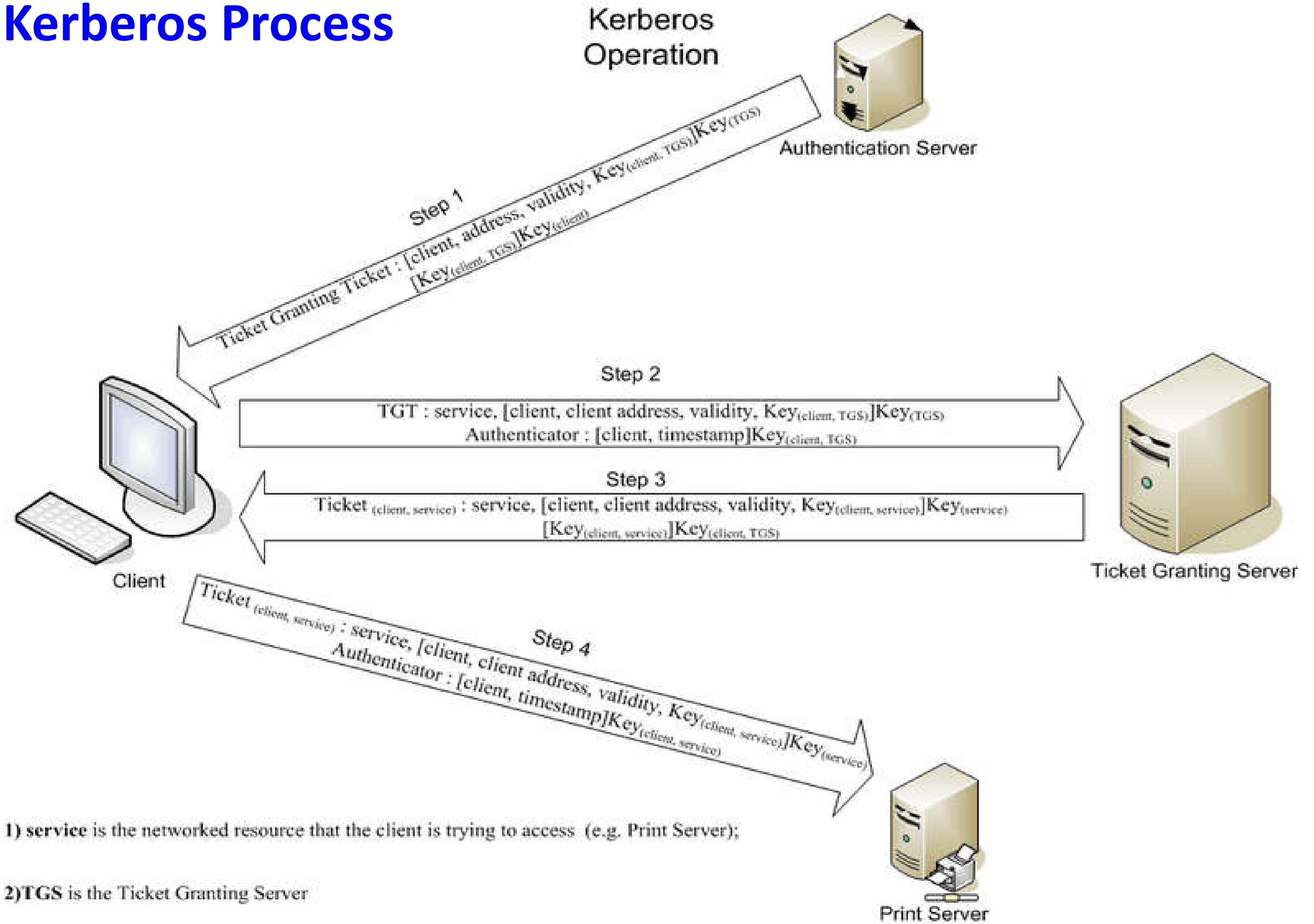
3.3.1 Introduction

Kerberos



3.3 Kerberos

3.3.2 Kerberos Process



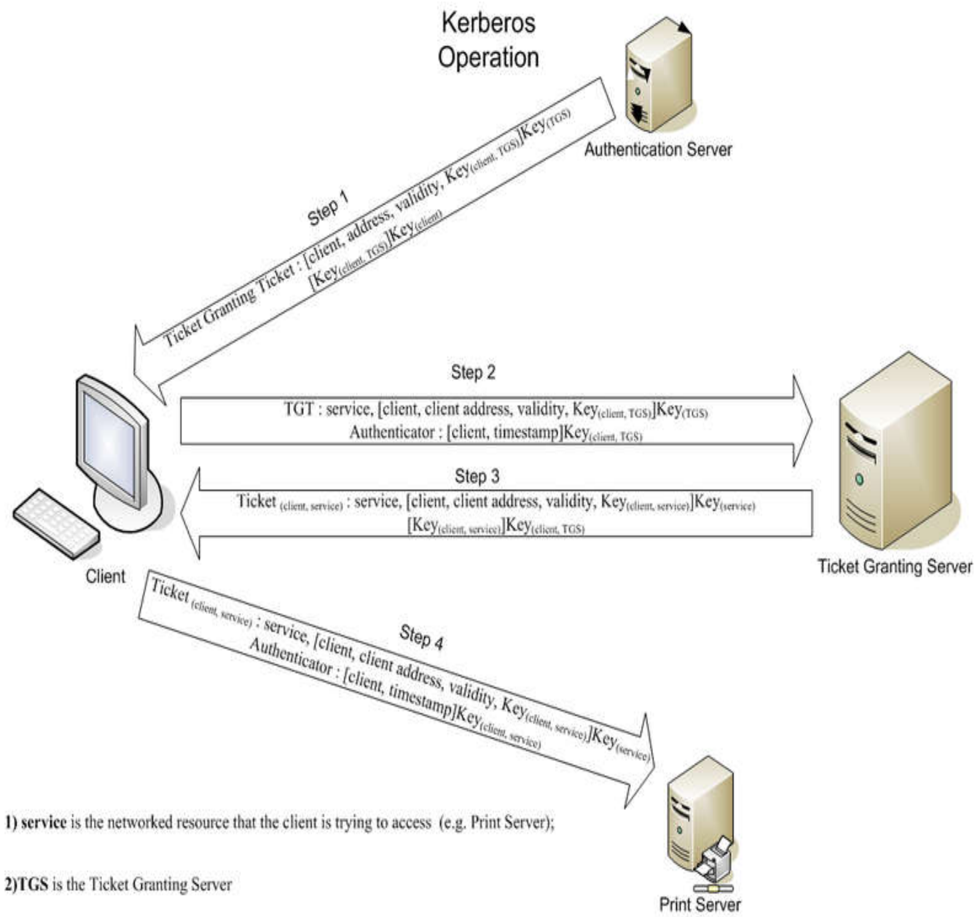
1) service is the networked resource that the client is trying to access (e.g. Print Server);

2)TGS is the Ticket Granting Server

3.3 Kerberos

3.3.2 Kerberos Process

- **The process includes**
  - User Client-based Logon
  - Client Authentication
  - Client Service Authorization
  - Client Service Request



3.3 Kerberos

3.3.2 Kerberos Process

- **Kerberos 过程**
  - 用户基于客户机程序登录的步骤：
    - ✧ 用户 *Bob* 输入用户名和密码到客户机。上述的用户名和密码构成长期密钥 (Long-term Key)，其加密的数据不应该在网络上传输 (以规避暴力攻击)。
    - ✧ 客户机程序在输入的密码上运行一个单向函数 (大多数为 Hash 函数)，得到 [客户机/客户 密钥]，这是 *Bob* 和 AS 预先协商好的主密钥 (Master Key)。由于 Hash 函数的单向性，对主密钥的破解并不会威胁到长期密钥的安全。不同的用户对应于不同的主密钥。主密钥由客户机程序保管。
    - ✧ 后面涉及的 [客户/TGS 会话密钥] 以及 [客户/SS 会话密钥]是短期密钥/会话密钥 (Short-term Key / Session Key)，利用主密钥实现交换/发布。

3.3 Kerberos

3.3.2 Kerberos Process

- **Kerberos 过程**
  - 客户机程序客户认证步骤：
    - ✧ 客户机向 AS 发送一个明文消息，代表用户请求服务，例如 “用户 *Bob* 想请求服务”。
    - ✧ AS 校验 *Bob* 是否在它的数据库里。如果在，AS 返回以下两条信息给客户机：
      - ○ 消息 A：用 [客户机/客户 密钥] (即 AS 与 *Bob* 预先协商好的主密钥) 加密的 [客户/TGS 会话密钥]
      - ○ 消息 B：用 TGS 密钥加密的票据授权票据 TGT (包括客户 ID，客户网络地址，票据有效期，[客户/TGS 会话密钥])
    - ✧ 客户机收到消息 A 和 B 后，用主密钥解密消息 A 得到 [客户/TGS 会话密钥]。该会话密钥用在将来与 TGS 的通信上，这样客户有了足够的信息向 TGS 证明自己的身份 (注意到客户机无法解密消息 B，因为它是用 TGS 的密钥加密的)。

3.3 Kerberos

3.3.2 Kerberos Process

- **Kerberos 过程**
  - 客户服务认证步骤：
    - ✧ 当申请服务时，客户机向 TGS 发送以下两条消息：
      - ◌ 消息 C：由消息 B 和申请的服务的 ID 组成
      - ◌ 消息 D：用 [客户/TGS 会话密钥] 加密的认证 (由客户 ID 和时间戳组成)
    - ✧ 基于收到的消息 C 和 D，TGS 从消息 C 中重新获取消息 B。它用 TGS 密钥解密消息 B，从而得到 [客户/TGS 会话密钥]。 TGS 使用这个密钥解密消息 D (即认证)，而后返回给客户机两条信息：
      - ◌ 消息 E：用 SS 服务器密钥加密的 客户-SS 服务票据 ST (包括客户 ID， 客户网络地址，[客户/SS 会话密钥]，票据有效期)
      - ◌ 消息 F：用 [客户/TGS 会话密钥] 加密的 [客户/SS 会话密钥]

中山大學 SUN YAT-SEN UNIVERSITY

3.3 Kerberos

3.3.2 Kerberos Process

- **Kerberos 过程**
  - 客户服务申请步骤：
    - ✧ 基于从 TGS 收到的消息 E 和 F，客户机有足够的信息向 SS 认证自己。客户机联系 SS，并向其发出以下两条消息：
      - 消息 E：由先前的步骤得到、用 SS 服务器密钥加密的 客户-SS 服务票据 ST
      - 消息 G：用 [客户/SS 会话密钥] 加密的一个新的认证，包括客户 ID 和时间戳
    - ✧ SS 用它自己的密钥解密消息 E (票据)，重新得到 [客户/SS 会话密钥]。用这个会话密钥，SS 解密消息 G 得到认证，返回确认函 H 给客户机，确认其身份真实，并同意向该客户提供服务：
      - 消息 H：在客户认证中找到时间戳，加1， 用 [客户/SS 会话密钥] 加密

3.3 Kerberos

3.3.2 Kerberos Process

- **Kerberos 过程**
  - 客户服务申请步骤：
    - ✧ 客户机使用 [客户/SS 会话密钥] 解密确认函 H，并检查时间戳是否被正确地更新。如果是，客户机可以信赖服务器，并可以向服务器发送服务请求。
    - ✧ 服务器向客户机提供其所请求的服务。

3.3 Kerberos

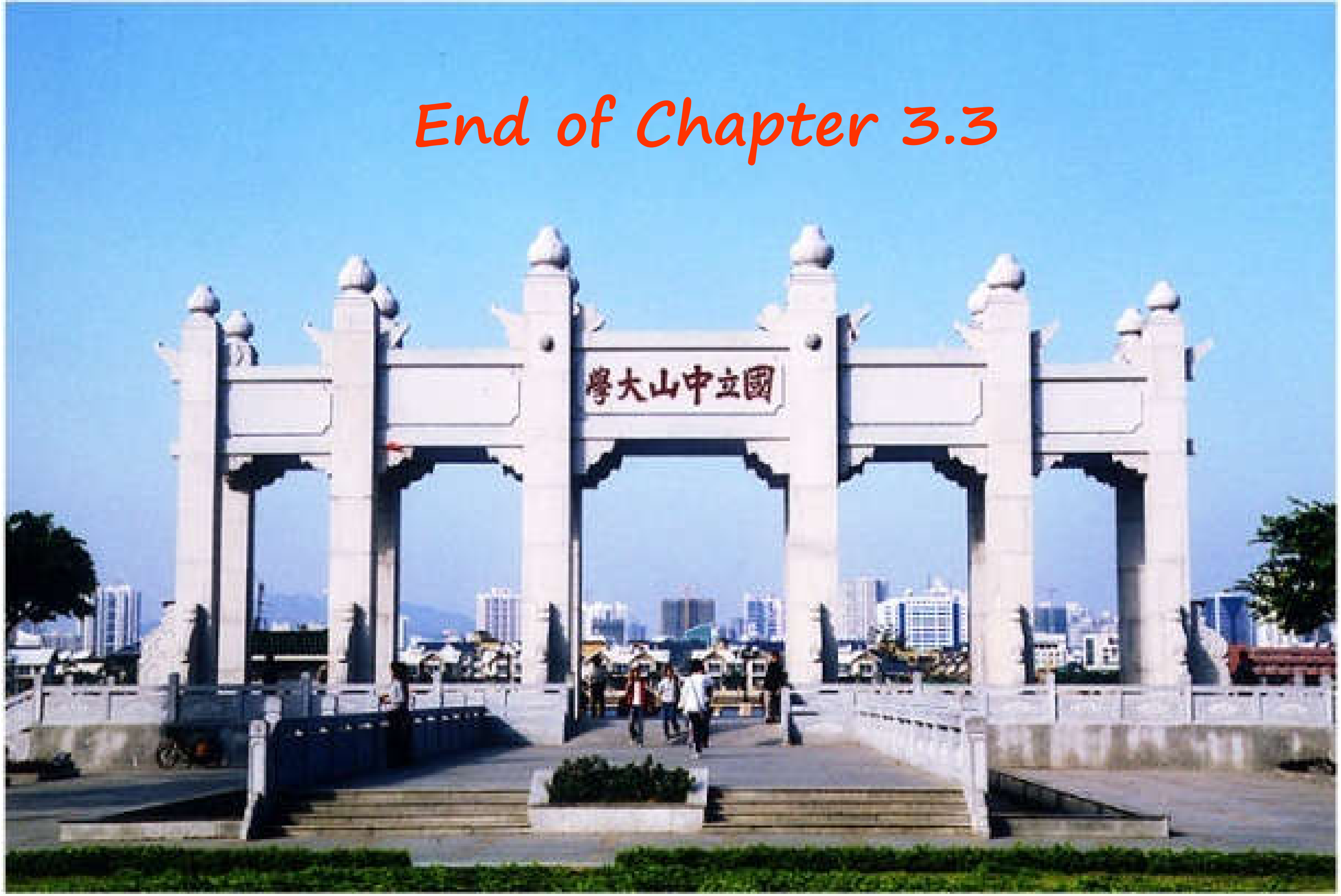
3.3.3 Drawbacks & Limitations

- Single point failure (continuously available server)
- Demand clock synchronization
  - ✧ Network time protocol for synchronization
- No general implementation standard
- All authentication in KDC (Key Distributing Center)

3.3 Kerberos

3.3.3 Drawbacks & Limitations

- **Kerberos 的局限性**
  - 单点失败
    - ✧ 它需要中心服务器的持续响应。这个缺陷可以通过使用复合 Kerberos 服务器和缺陷认证机制弥补。
    - ✧ 所有用户使用的主密钥都存储于中心服务器 (KDC) 中，危及服务器的安全的行为将危及所有用户的密钥。
    - ✧ 一个危险客户机将可能危及用户密码安全。
  - 时钟同步
    - ✧ Kerberos 要求参与通信的主机的时钟同步。票据具有一定有效期，如果主机的时钟与 Kerberos 服务器的时钟不同步将导致认证失败。默认设置要求时钟的时间差不超过10分钟，通常用网络时间协议后台程序保持时钟同步
  - 管理协议未标准化 (RFC 3244 描述了一些更改)。


End of Chapter 3.3
國立中山大學