MONASH University
Information Technology

FIT3031 INFORMATION & NETWORK SECURITY

Information Technology

FIT3031 INFORMATION & NETWORK SECURITY

Lecture 4:
Authentication Applications

Unit Structure: Lecture Topics

- ✓ OSI security architecture
  - **common security standards and protocols for network security applications**
  - **common information risks and requirements**
- ✓ operation of private key encryption techniques
- ✓ operation of public encryption techniques
- ✓ concepts and techniques for digital signatures, authentication and non-repudiation
- security threats of web servers, and their possible countermeasures
- Wireless Security Issues
- security threats of email systems and their possible countermeasures
- IP security
- intrusion detection techniques for security purpose
- risk of malicious software, virus and worm threats, and countermeasures
- firewall deployment and configuration to enhance protection of information assets
- network management protocol for security purpose

Lecture 4: Objectives

- **Appreciate the importance of authentication tools and protocols**
- **Understand Kerberos authentication protocol**
- **Be familiar with X.509 directory authentication service and public key infrastructure**
- **Appreciate the concept of federated identity management**

Lecture 4: Outline

- **Authentication protocol**
  - Requirements
  - Access control to services
  - Simple authentication dialogue
  - More secure authentication dialogue
  - Kerberos v.4 authentication
- **X.509 authentication services**
  - certificates
  - Different authentication procedure
- **Public Key Infrastructure**
- **Federated Identity Management**

Access Control

- **In today's organizations, we mostly have open distributed architecture consisting of**
  - workstations (clients)
  - distributed/centralized servers.
- **Enforcement of access restriction to the services on servers is crucial for security**
- **<u>Three</u> threats exist:**
  - User pretends to be another user
  - User alters the network address of a workstation
  - User eavesdrops on exchanges and uses a replay attack
- **A workstation cannot be trusted for access control purpose**

Access Control …

- **Three approaches to security can be considered:**
  1. rely on **client workstations** to assure user(s) identity and rely on **server to enforce a security policy** based on **user id**
  2. require that **client systems** authenticate themselves to **servers**, but trust the **client systems** to assure **user(s) identity**
  3. **require the user** to prove identity for **each service** on the **server** and also the **server to prove its identity to clients**
- **The first or second strategy may be suitable for a small closed environment**
  - inadequate for larger open environment
- **The third approach is needed**
  - supported by Kerberos authentication protocol

Kerberos

In Greek mythology, a many headed dog, the guardian of the entrance of Hades

Kerberos …

- **Provides a centralized authentication server to authenticate <u>users to servers</u> and <u>servers to users</u>**
  - **> allows users access to services distributed through network**
  - **> without needing to trust all workstations**
  - **> rather all trust a central authentication server**
- **Relies on conventional/Symmetric encryption**
  - makes **<u>no</u>** use of public-key/Asymmetric encryption
- **Two versions: version 4 and 5**
- **Version 4 makes use of DES**

Kerberos Requirement

- **The first published report identified its requirements as:**
  - **Security** - secure enough to prevent eavesdropping
  - **Reliability** - highly reliable to ensure the availability
  - **Transparency** - user should not be aware of authentication taking place
  - **Scalability** - capable of supporting large number of clients and servers

Kerberos Overview

- **Its a basic third-party authentication scheme**
- **Employs an Authentication Server (AS)**
  - users initially negotiate with **AS** to **identify** self
  - **AS** provides a non-corruptible authentication credential (ticket granting ticket **TGT**)
- **Employs a Ticket Granting server (TGS)**
  - *users subsequently request access to other services from TGS on basis of users TGT*
- ***Before going into details, we consider a simple authentication dialogues***

Why Authentication Server?

- **When a request is made by a client for a network service, the server must be able <u>to confirm the identity of the clients</u>**
- **This places a substantial burden on the server where each client/server interaction requires authentication**
- **An alternative is to use an Authentication Server (AS)**
- **AS stores the password of all users in a centralized database and shares a unique key with <u>each server</u>**

A **Simple** Authentication Dialogue(1)

Consider the following notations:

- **C** = Client
- **AS** = authentication server
- **V** = server
- $\mathbf{ID_c}$ = identifier of user on C
- $\mathbf{ID_V}$ = identifier of V
- $\mathbf{P_c}$ = password of user on **C**
- $\mathbf{AD_c}$ = network address of **C**
- $\mathbf{K_V}$ **= secret encryption key shared by AS and V**
- **TS** = timestamp
- **||** = concatenation

A **Simple** Authentication Dialogue(2)

- **C send a message to AS**
- **AS checks the database for user ID and password match, and whether the user has access permission to (Server) V.**
  - If passed, it takes the user as authentic
  - AS creates a ticket.
    - The ticket contains **user ID, server ID and network address**, all encrypted by a secret key shared by AS and V.
- **C sends a message to V with C's ID and the ticket.**
- **V decrypts the ticket**
  - verifies whether the user ID in the ticket is the same as the unencrypted user ID
  - If those matches, the server grants requested service

C → AS: ID_c || P_c || ID_v

AS → C: Ticket

C → V: ID_c || Ticket

Ticket = E_{K_V}[IDc || AD_c || IDv]

The ticket is encrypted to prevent forgery

AD_c is included to counter replay attack

A Simple Authentication Dialogue(3)

- **There are a few problems with this scheme:**
  - the user **needs to enter** password every time a service is accessed
    - > each attempt for the same service requires **reentering** the password
  - a new ticket for **every** different service
  - password is transmitted in **plaintext** format
    - > easy for an eavesdropper to capture the password

$$C \rightarrow AS: ID_c \,||\, P_c \,||\, ID_v$$

A more **secure** authentication Dialogue (1)

- **An improved scheme:**
  - avoiding plaintext password
  - Employ another server, called **ticket-granting server** (TGS)
    - >TGS satisfies **two** requirements
      - **only one password query per session**
      - **protection of the user’s password**

A more **secure** authentication Dialogue (2)

Once per user logon session:

$$\text{(1) C} \rightarrow \text{AS:} \quad ID_C \| ID_{tgs}$$

$$\text{(2) AS} \rightarrow \text{C:} \quad \text{E}(K_c, Ticket_{tgs})$$

Once per type of service:

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

$$\text{(4) TGS} \rightarrow \text{C:} \quad Ticket_v$$

Once per service session:

$$\text{(5) C} \rightarrow \text{V:} \quad ID_C \| Ticket_v$$

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

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

The client requests a **ticket-granting ticket** from **AS**

- **AS** sends a ticket encrypted with a key $\mathbf{K}_c$ derived from the user's password **no password** is transmitted.
- ticket is **reusable - timestamp** is include to counter ticker **spoofing**

The client **C** requests a **service using granting ticket (TGS)** decrypts the ticket using secret shared key $\mathbf{K_v}$ checks lifetime, used id, network address

Client requests access to service V using service granting ticket

A more **secure** authentication Dialogue (3)

- **Problem still remains:**
  - *Lifetime*$_{1\ \&\ 2}$ associated with the ticket-granting ticket
  - If too short → repeatedly asked for password
  - If too long → greater opportunity to replay
- **The threat is that an opponent will steal the ticket and use it before it expires**

- **Kerberos authentication dialogue addresses these problems**

Kerberos v4 Overview

- **a basic third-party authentication scheme**
- **have an Authentication Server (AS)**
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- **have a Ticket Granting server (TGS)**
  - users subsequently request access to other services from TGS on basis of users TGT
- **using a complex protocol using DES**

Kerberos V.4 Authentication dialogue

Table 4.1 Summary of Kerberos Version 4 Message Exchanges

(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)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

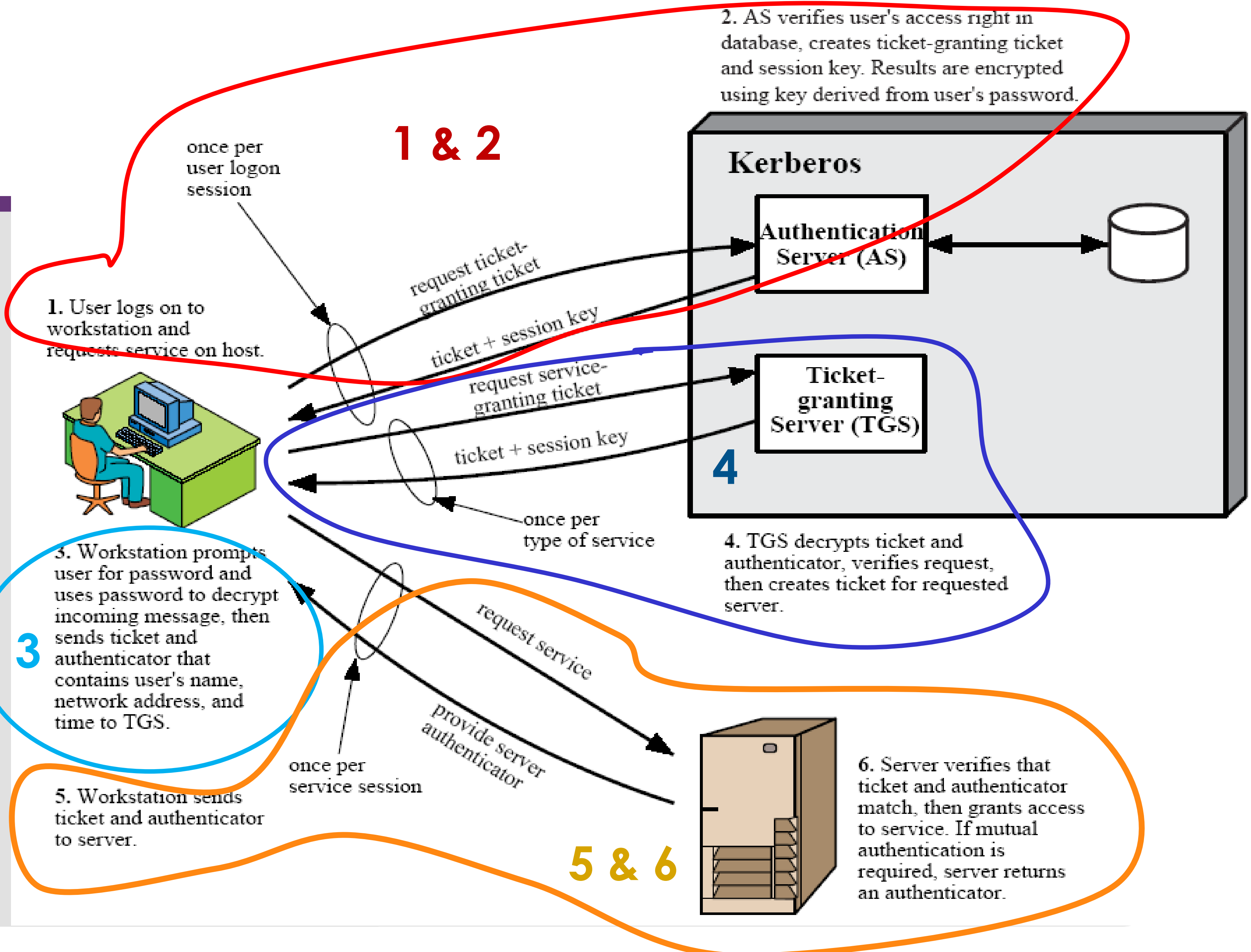
(c) Client/Server Authentication Exchange to obtain service

Kerberos V.4 (1)

- **A session key $K_{c,tgs}$ is introduced**
- **The ticket contains the session key**
- **The authenticator proves client's identity**
  - can be used only once
  - has a short lifetime
  - threat of stealing both the ticket and authenticator for later presentation is removed
- **The authenticator is encrypted by the session key**

MONASH University
Information Technology

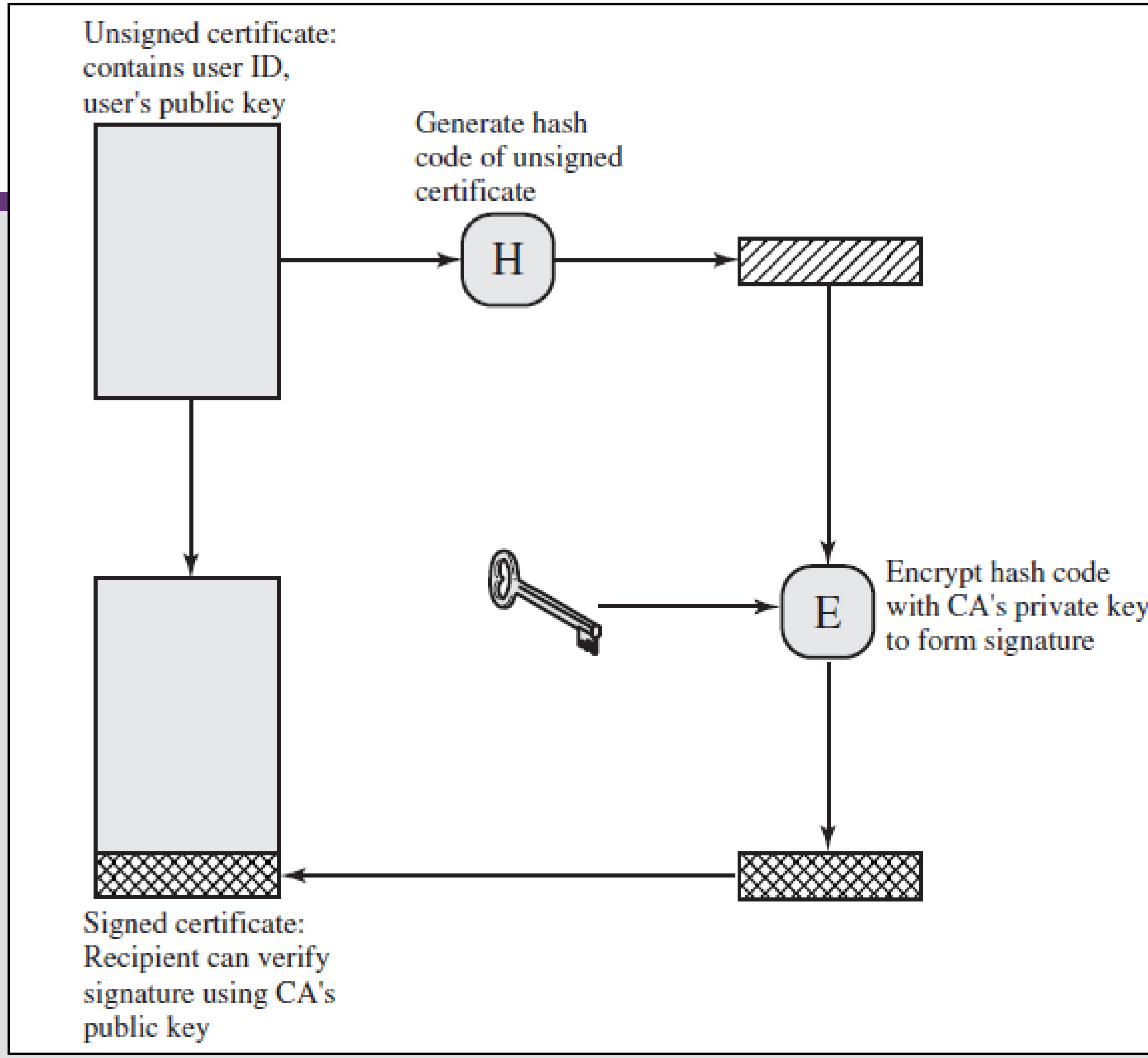
Kerberos V.4 (2)



Kerberos V.4 (3)

- **Currently two Kerberos versions:**
  - V4 and v5 version
  - Kerberos v5 is an Internet standard
  - specified in RFC1510, and used by many utilities
- **To use Kerberos:**
  - need to have a KDC on your network
  - need to have Kerberised applications running on all participating systems
  - major problem - US export restrictions
  - Kerberos cannot be directly distributed outside the US in source format (& binary versions must obscure crypto routine entry points and have no encryption)
  - else crypto libraries must be reimplemented locally

X.509 Certificate Use



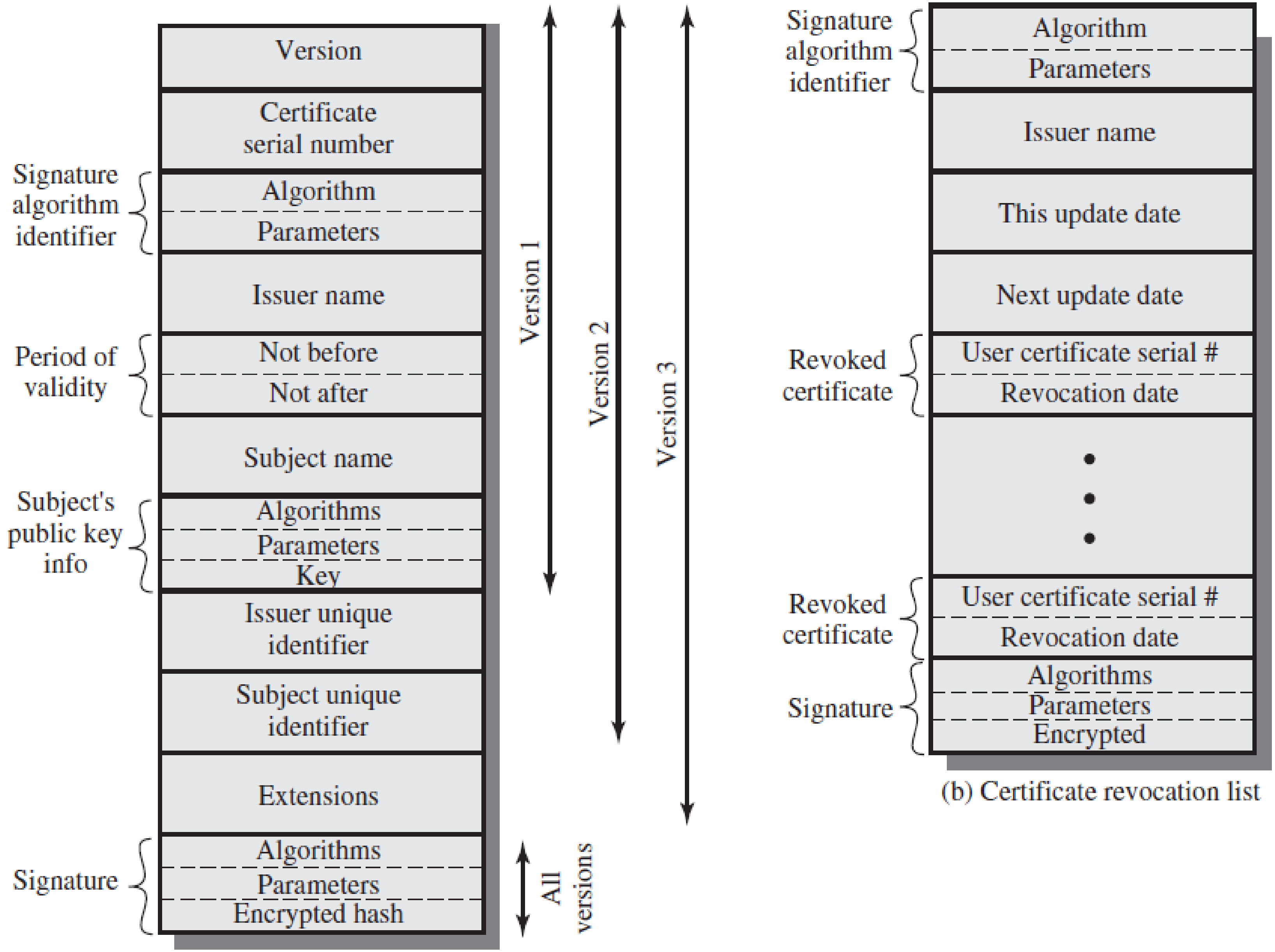
Public-Key Distribution of Secret Keys

- **Conventional encryption requires 2 parties to securely share a secret key**
- **Diffie-Hellman key exchange *does not authenticate* the 2 partners**
- **Alternative is the use of *public key certificates***
  – Encrypt session key using public-key encryption
  – Receiver obtains the sender's public key by means of ***public key certificate that provides assurance of a valid public key***

X.509 Authentication Service

- **X.500 directory is a distributed set of servers that maintains a database about users**
- **X.509 defines a framework of authentication services provided by X.500 to its users**
- **Each certificate contains the public key of a user and is signed with the private key of a CA (Certification Authority)**
- **Used in S/MIME, IP Security, SSL/TLS and SET.**
- **Use of RSA is recommended.**

X.509 Formats



(a) X.509 certificate

(b) Certificate revocation list

Getting a Certificate

- **Any user with *access to CA* can get *any certificate* from it**
- **Only the issuing CA *can modify* a certificate**
  - any modification is detected
- **Because certificates *cannot be forged*, they can be *placed* in a public directory**

CA Hierarchy (1)

- **If both users share a common CA then it is assumed they know its public key**
- **Otherwise CA's must form a hierarchy**
- **Use certificates linking members of hierarchy to validate other CA's**
  - each CA has certificates for clients (forward) and parent (backward)
- **Each client trusts parents certificates**
- **Enables verification of any certificate from one CA by users of all other CAs in hierarchy**

CA Hierarchy (2)

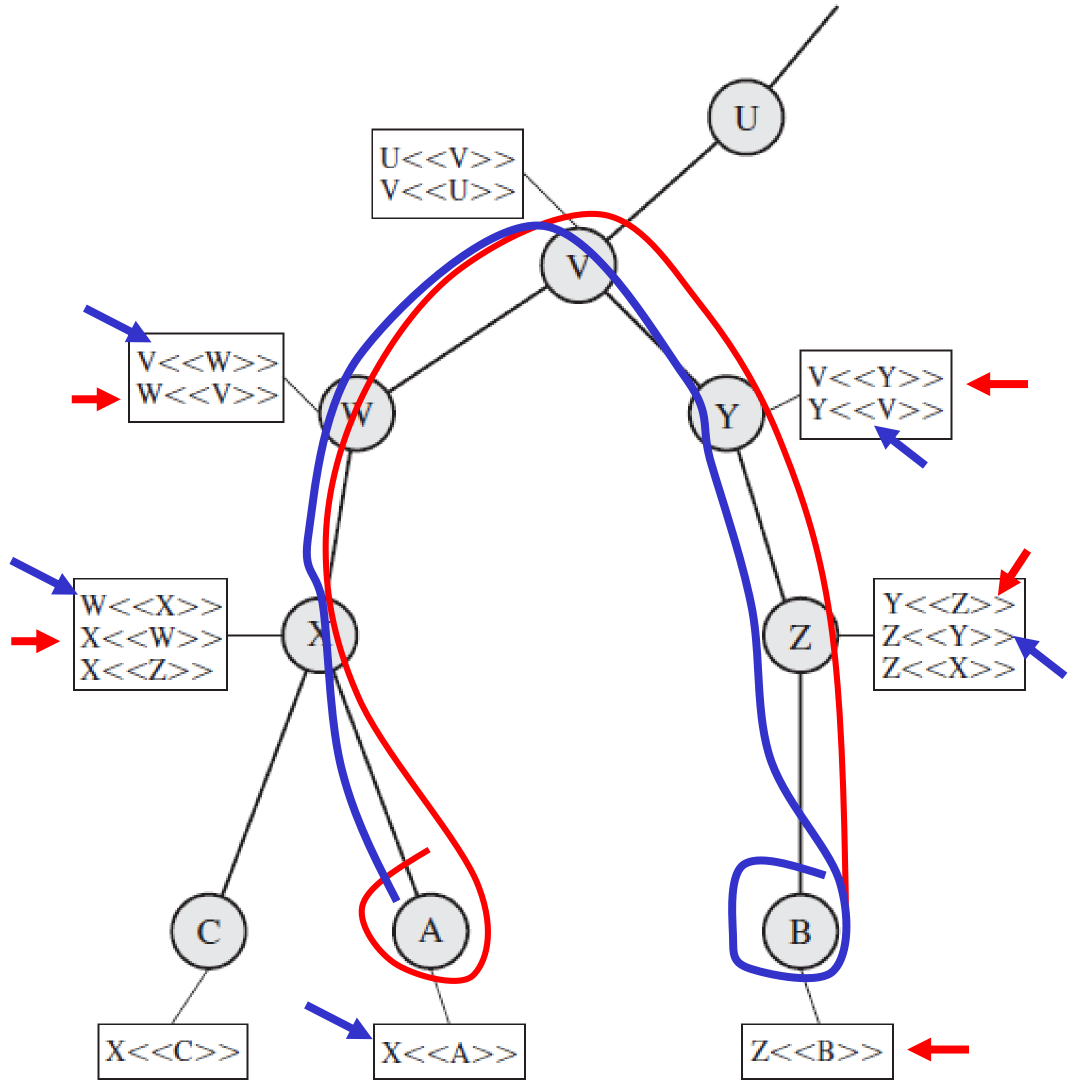
- Figure shows X.509 hierarchy to mutually verify clients certificates
- X <<C>> the certificate of user C issued by certification authority X
- The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry

- User A acquires User B certificate using chain:

X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

- User B acquires User A certificate using chain:

Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>



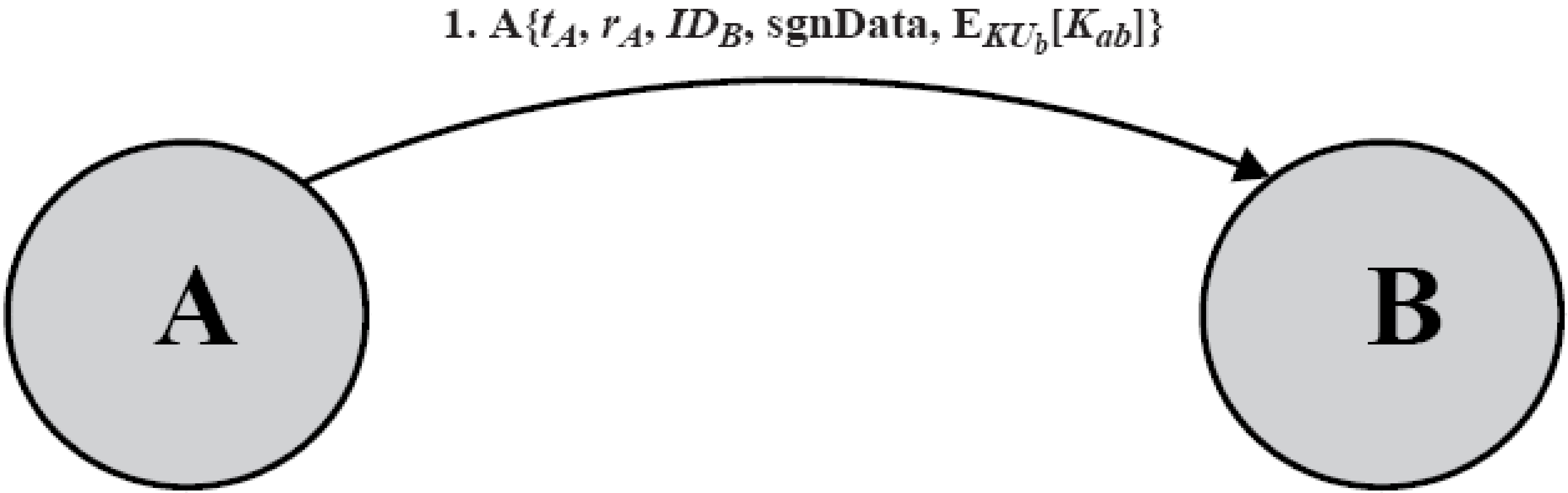
Certificate Revocation

- **Certificates have a period of validity**
- **May need to be revoked before expiry, eg:**
  – user's private key is compromised
  – user is no longer certified by this CA
  – CA's certificate is compromised
- **CA's maintain list of revoked certificates**
  – the Certificate Revocation List (CRL)
- **Users should check certificates with CA's CRL**

Authentication Procedure

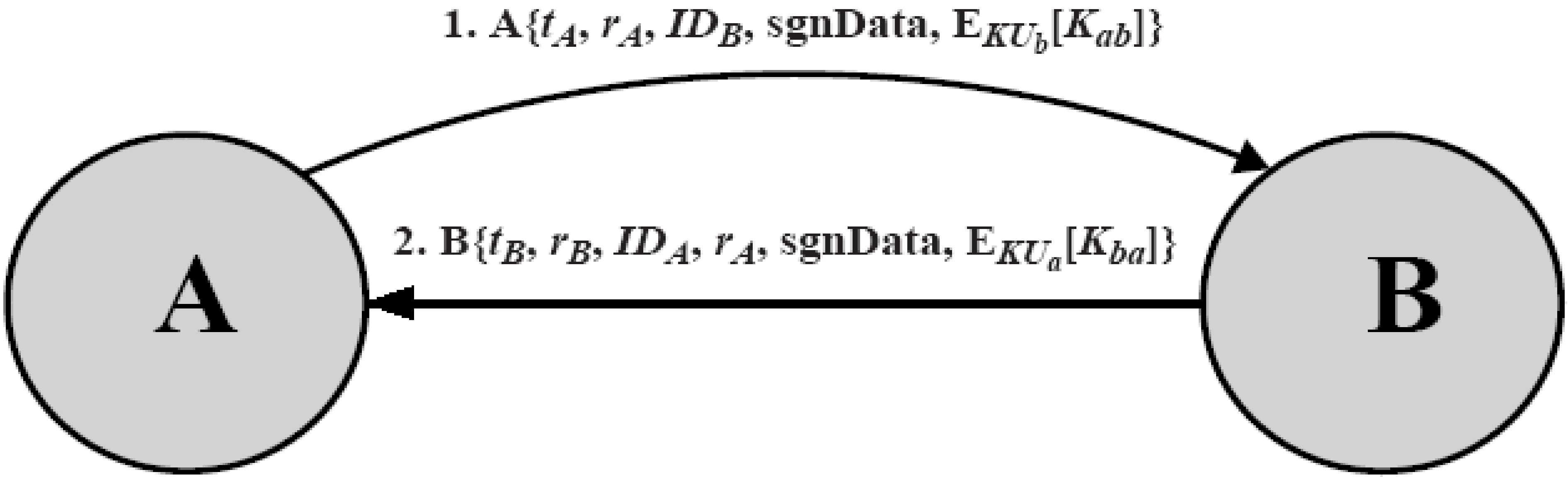
- **X.509 includes three alternative authentication procedures:**
  - One-Way Authentication
    - > for unidirectional messages (like email)
  - Two-Way Authentication
    - > for interactive sessions when timestamps are used
  - Three-Way Authentication
    - > for interactive sessions with no need for timestamps (and hence no synchronised clocks).
- **all use public-key signatures**

X.500 One-Way Authentication



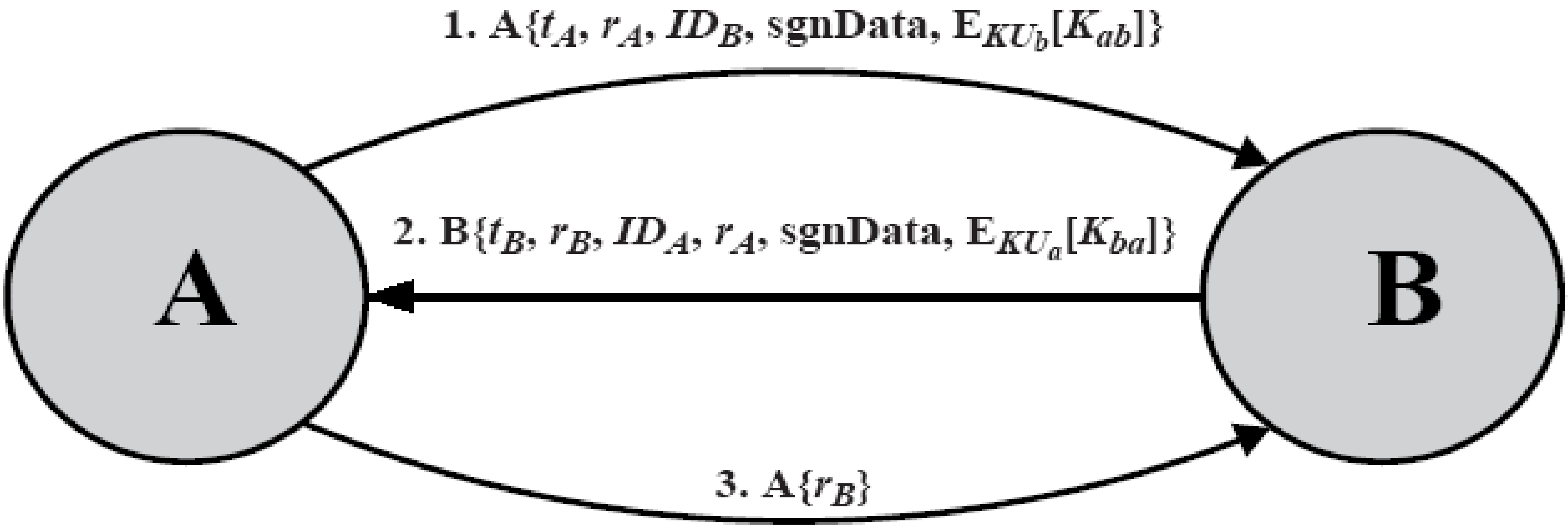
- 1 message (A → B) used to establish with **synchronized clocks**
  - the **identity** of A and that message is from A
  - message was intended for B
  - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A
- timestamp need to be **checked** 'or' **relied** upon

X.500 Two-Way Authentication



- 2 messages (A → B, B → A) for two way authentication, with **synchronized** clocks in addition:
  - **the identity of B and that reply is from B**
  - **that reply is intended for A**
  - **integrity & originality of reply**
- reply includes original nonce from A, also timestamp and nonce from B
- timestamp need to be checked or relied upon

X.500 Three-Way Authentication



- 3 messages (A→B, B→A, A→B) which enables above *authentication without synchronized clocks*
- has reply from A back to B containing signed copy of nonce from B
- implies timestamps need not be checked or relied upon

Further Reading

- **Study Guide 4**
- **Chapter 4 of the textbook: Network Security Essentials- Application & Standards" by William Stallings 5th Edition, Prentice Hall, 2013**
- **Additional resources for this week**

- **Acknowledgement: part of the materials presented in the slides was developed with the help of Instructor's Manual and other resources made available by the author of the textbook.**