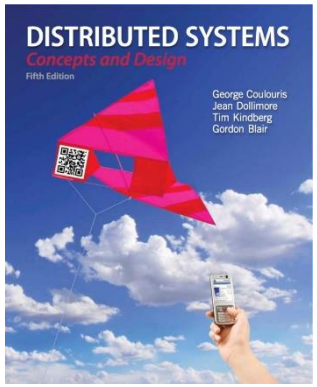
# Distributed Computing Principles: Security

**Dr. Minxian Xu**
**Associate Professor**
**Research Center for Cloud Computing**
**Shenzhen Institute of Advanced Technology, CAS**
**http://www.minxianxu.info/dcp**




Most concepts are drawn from Chapter 11

子墨子解带为城，以牒为械，公输盘九设攻城之机变，子墨子九距之；公输盘之攻械尽，子墨子之守围有余。
——《墨子·公输》

# Review

Q1: What are Name Services and why do we need them?

# Review

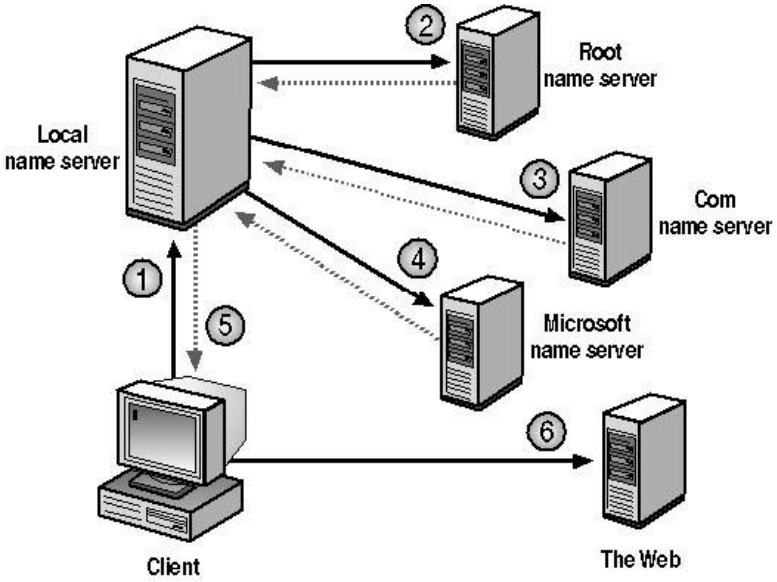
- The major operation of a name service is to resolve a name, i.e to lookup the attributes that are bound to the name.
- Name management is separated from other services largely because of the openness of distributed systems, which brings the following motivations:
  - Unification: Resources managed by different services use the same naming scheme, as in the case of URIs.
  - Integration: To share resources that were created in different administrative domains requires naming those resources. Without a common naming service, the administrative domains may use entirely different name formats.

# Review

## Q2: What is the process of name resolution?

# Review

- Name resolution is in general an iterative process. A name either resolves to a set of primitive attributes or it resolves to another name.
- The use of aliases make it possible for resolution cycles to occur and the potential for the resolution process to never terminate. Two solutions to overcome this include:
  - Abandon the resolution process after some number of iterations.
  - Require administrators to ensure that no cycles occur



## Q3: What is navigation and what are the approaches to navigation?

# Review

- When the name service is distributed then a single server may not be able to resolve the name. The resolve request may need to propagate from one server to another, referred to as *navigation*.
- Classification of different navigation approaches.
  - ***iterative navigation*** -- The client makes the request at different servers one at a time. The order of servers visited is usually in terms of domain hierarchy. Always starting at the root server would put excessive load on the root.
  - ***multicast navigation*** -- The client multicasts the request to the group (or a subset) of name servers. Only the server that holds the named request returns a result.
  - ***non-recursive server-controlled navigation*** -- The client sends the request to a server and the server continues on behalf of the client, as above.
  - ***recursive server-controlled navigation*** -- The client sends the request to a server and the server sends the request to another server (if needed) recursively.

Q4: What is Domain Name System and in what aspects does it improve on a file-based implementation?

# Review

- The Domain Name System (DNS) is a name service design whose main naming database is used across the Internet.
- Before DNS, all host names and addresses were held in a single central master file and downloaded by FTP to all computers that required them.
- The problems with the original name service included:
  - It did not scale to large numbers of computers.
  - Local organizations wished to administer their own naming systems.
  - A general name service was needed -- not one that serves only for looking up computer addresses.
- DNS is designed for use in multiple implementations, each of which may have its own name space, though in practice the Internet DNS name space is the one in widespread use.

# Some Cyber Security Facts

- **1. 95% of breached records came from only three industries in 2016**
  - Government, retail, and technology (high level of personal identifying information contained in their records)
- **2. There is a hacker attack every 39 seconds**
- **3. 43% of cyber attacks target small business**
  - 64% of companies have experienced web-based attacks. 62% experienced phishing & social engineering attacks. 59% of companies experienced malicious code and botnets and 51% experienced denial of service attacks.

data: https://www.cybintsolutions.com/cyber-security-facts-stats/

# Some Cyber Security Facts

- **4. The average cost of a data breach in 2020 exceeded $150 million**
  - As more business infrastructure gets connected, Juniper Research data suggests that cybercrime will cost businesses over $2 trillion total in 2019.
  - **June 2019:** The Australian National University has been hit by a massive data hack, with unauthorised access to significant amounts of personal details dating back 19 years.
- **5. Since 2013 there are 3,809,448 records stolen from breaches every day**
  - 158,727 per hour, 2,645 per minute and 44 every second of every day reports Cybersecurity Ventures.

# Some Cyber Security Facts

- **6. Over 75% of healthcare industry has been infected with malware over past years**
- **7. Large-scale DDoS attacks increase in size by 500%**
- **8. Approximately $6 trillion is expected to be spent globally on cybersecurity by 2021**

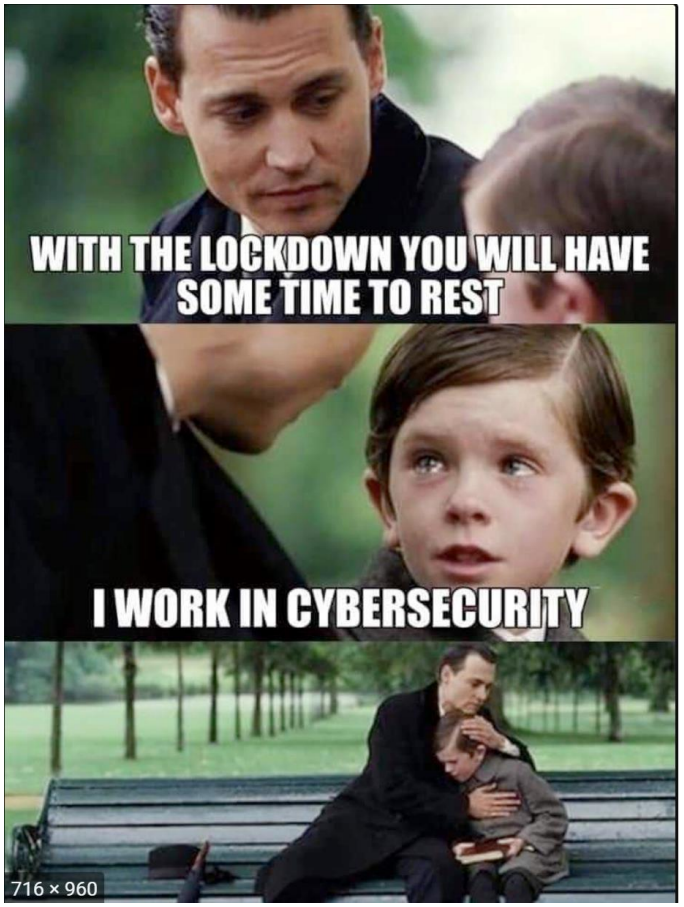
# Some Cyber Security Facts

- **9. Unfilled cybersecurity jobs worldwide reached 3.5 million by 2021**
  - More than 300,000 cybersecurity jobs in the U.S. are unfilled, and postings are up 74% over the past five years.
- **10. By 2020 there will be roughly 200 billion connected devices**
  - The risk is real with IoT and its growing
  - Smart Healthcare
  - Smart City
  - Smart Transport

# Some Cyber Security Facts

- **11. 95% of cybersecurity breaches are due to human error**
  - Cyber-criminals and hackers will infiltrate your company through your weakest link, which is almost never in the IT department.
- **12. Only 38% of global organizations claim they are prepared to handle a sophisticated cyber attack**
  - What's worse? An estimated 54 percent of companies say they have experienced one or more attacks in the last 12 months.
- **13. Total cost for cybercrime committed globally has added up to over $1 trillion dollars in 2018**
  - As long as you're connected to the Internet, you can become a victim of cyber attacks.

# Security memes


# **‘Zoom bombers’ invade virtual classrooms’:** Unauthorized Participants and Disruptions

# Zoom Challenges

1. Security challenge: Zoom bombings
2. Security challenge: Data leakages
3. Security challenge: Privacy shortcomings
4. Performance challenge: Ensuring quality of service
5. Reliability challenge: Ensuring availability at all times

https://www.sumologic.com/blog/zoom-security-challenges/

# Learning objectives

- Security model
  - Types of threat
- Basic techniques
  - Cryptographic techniques
    - *Secrecy*
    - *Authentication*
    - *Certificates and credentials*
    - *Access control*
  - Audit trails
- Symmetric and asymmetric encryption concepts
- Digital signatures
- Approaches to secure system design
- Pragmatics and case studies (Kerberos and Secure Socket Layer)

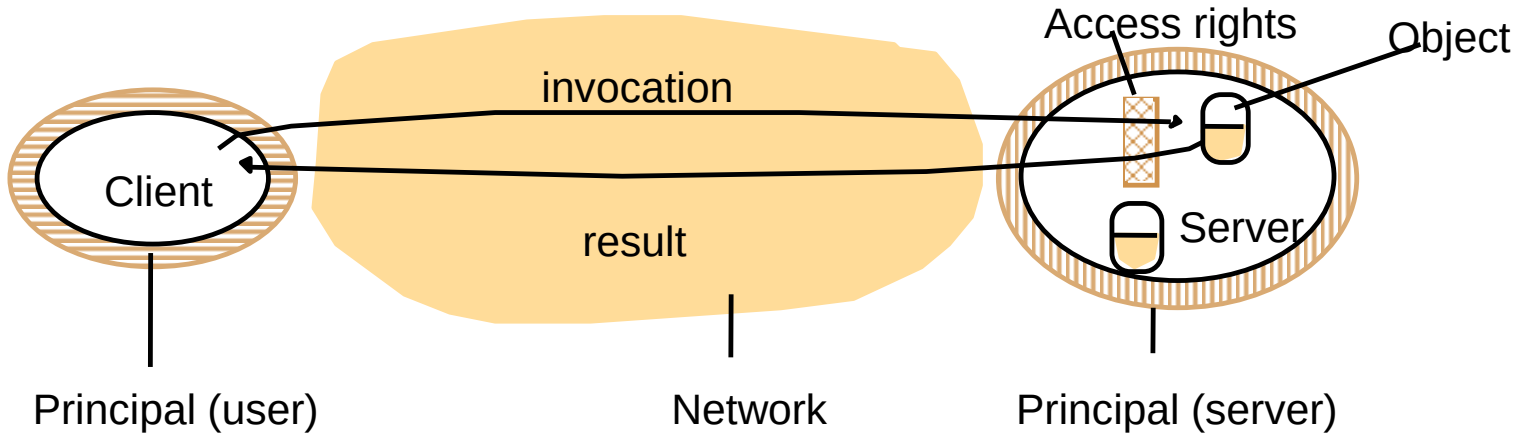
# Why Security is so important in DS?

- Security Goal: Restrict access to information/resources to just to those entities that are authorized to access.
- There is a pervasive need for measures to guarantee the privacy, integrity, and availability of resources in DS.
- Security attacks take various forms: **Eavesdropping, masquerading, tampering, and denial of service**.
- Designers of secure distributed systems must cope with the exposed interfaces and insecure network in an environment where attackers are likely to have knowledge of the algorithms used to deploy computing resources.
- Cryptography provides the basis for the authentication of messages as well as their secrecy and integrity.

# How is security in real world?

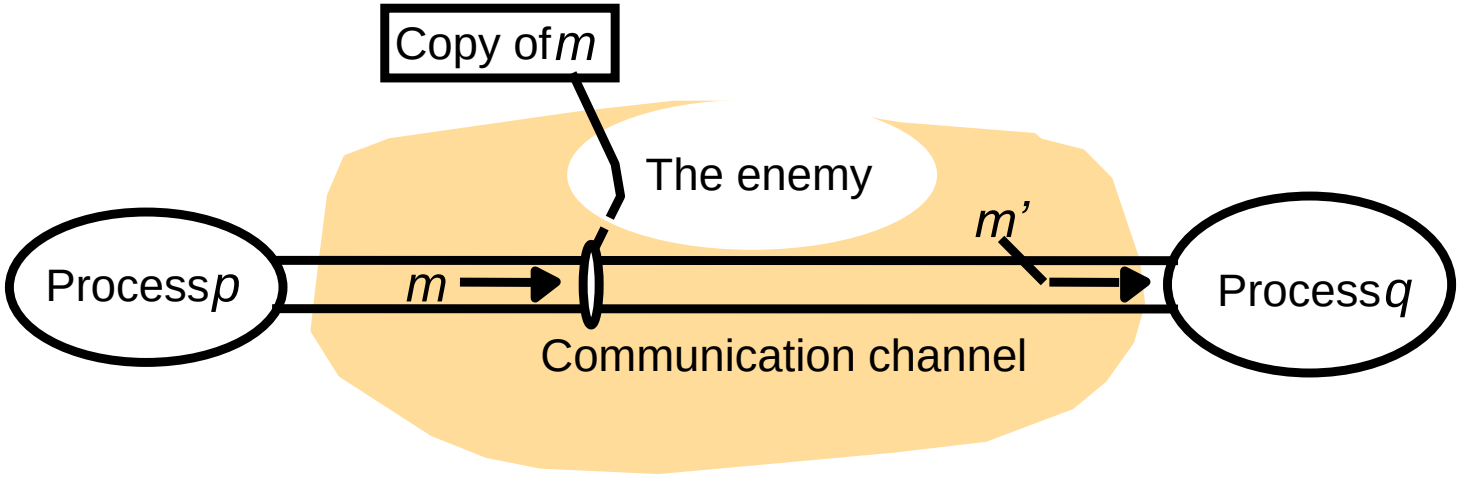
- In the physical world, organisations adopt “security policies” that provide for the sharing of resources within specified limits.
  - Company may permit entry to its building for its employees and for accredited visitors.
  - A security policy for documents may specify groups of employees who can access classes of documents or it may be defined for individual documents and users.
- Security policies are enforced with *security mechanisms*.
  - Access to building may be controlled by a **reception clerk**, who issues badges to accredited visitors, and enforced by security guard or by electronic door locks.
- In electronic world, the distinction between **security policy** and **mechanisms** is equally important.

# Chapter 2 Revision: Objects and principals



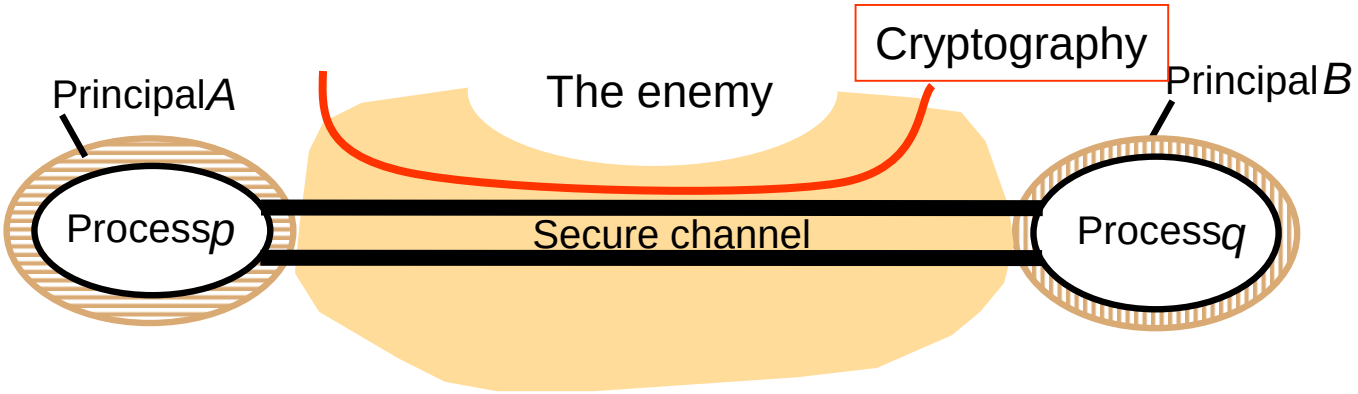

- Object (or resource)
  - Mailbox, system file, part of a commercial web site
- Principal
  - User or process that has authority (rights) to perform actions
  - Identity of principal is important

# Chapter 2 Revision: The enemy




- Attacks
  - On applications that handle financial transactions or other information whose secrecy or integrity is crucial
- Enemy (or adversary)
- Threats
  - To processes, to communication channels, denial of service

# Chapter 2 Revision: Secure channels




- Ownership of secrets:
  - Conventional shared crypto keys
  - Public/private key pair
- *Secrecy based on cryptographic concealment*
- *Authentication based on proof of ownership of secrets*

# Threats and Attacks

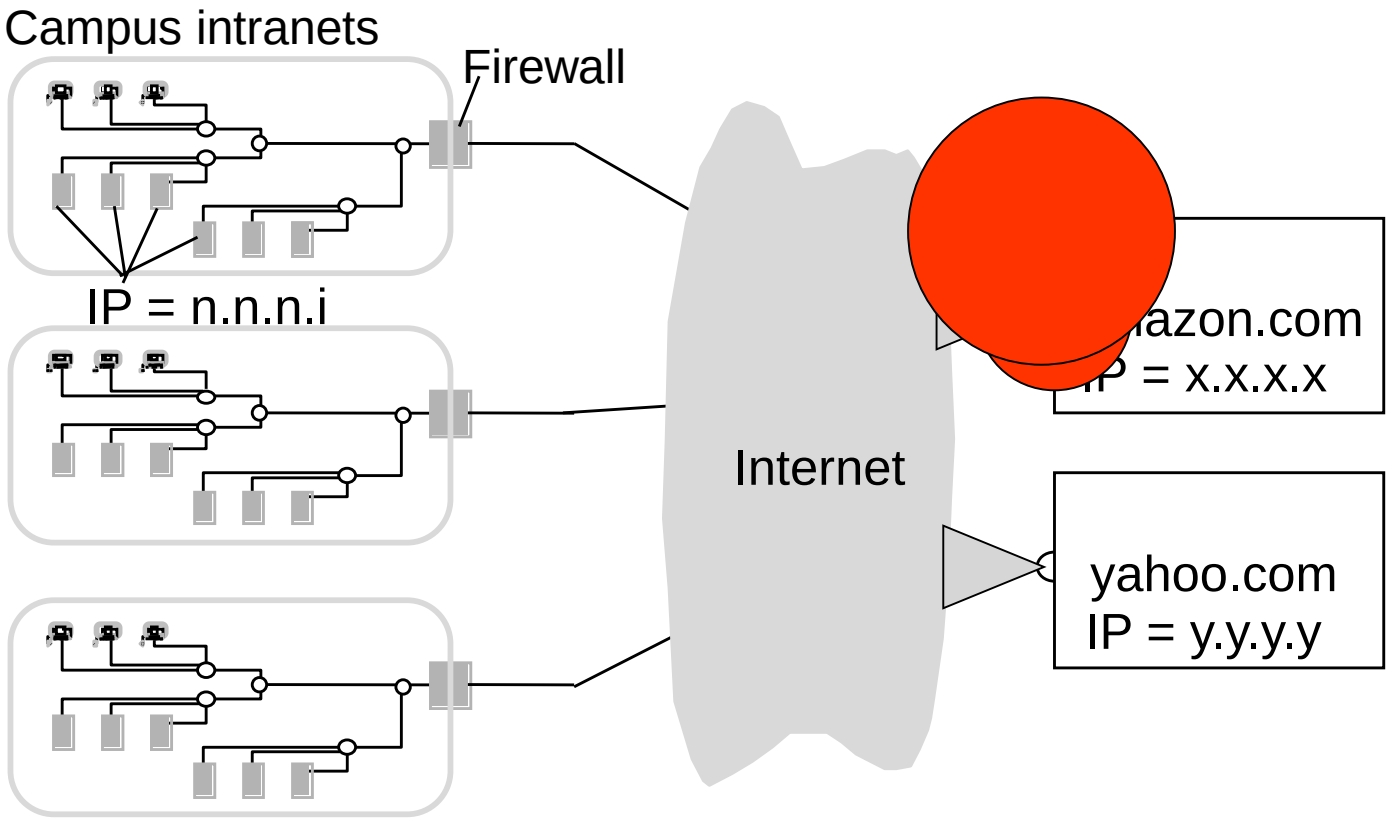
- Security Threats - Three broad Classes:
  - Leakage: Acquisition of information by unauthorised recipients
  - Tampering: Unauthorised alteration of information
  - Vandalism: Interference with the proper operation of systems
- Method of Attacks are listed below:
- Eavesdropping - A form of leakage
  - obtaining private or secret information or copies of messages without authority.
- Masquerading – A form of impersonating
  - assuming the identity of another user/principal – i.e, sending or receiving messages using the identity of another principal without their authority.
- Message tampering
  - altering the content of messages in transit
    - *man in the middle attack (tampers with the secure channel mechanism)*
- Replaying
  - storing secure messages and sending them at a later date
- Denial of service - Vandalism
  - flooding a channel or other resource, denying access to others

# Threats not defeated by secure channels

or other cryptographic techniques

- Denial of service (DoS) attacks
  - **Deliberately excessive use of resources** to the extent that they are not available to legitimate users
    - *E.g. the Internet 'IP spoofing' attack, February 2000*
- Trojan horses and other viruses
  - Viruses can only enter computers when program code is imported.
  - But users often require new programs, for example:
    - *New software installation*
    - *Mobile code downloaded dynamically by existing software (e.g. Java applets)*
    - *Accidental execution of programs transmitted surreptitiously*
  - Defences: code authentication (signed code), code validation (type checking, proof), sandboxing.

# The February 2000 IP Spoofing DDoS attack (creation of IP packets with a false source IP address)




*Compromised host on each local network sends repeatedly (for all **i**):*

Untrue!

| Echo request | source = x.x.x.x | destination = n.n.n.**i** |
|---|---|---|

*resulting in:*

| Echo reply | source = n.n.n.**i** | destination = x.x.x.x |
|---|---|---|

# Securing Electronic Transactions

- Email
  - Traditionally no support for security.
  - But it is important to Keep messages secret.
  - Modern mail clients incorporate cryptography.
- Purchase of goods and services
- Banking transactions
- Micro-transactions
  - Currently access to Web pages is not charged, but the development of Web as a highly quality publishing medium surely needs it.
  - The price of such services may amount to only a fraction of cent and the payment overhead must be low (for this to be feasible).
  - How to manage “fraudulent vendors” – who obtain payment with no intension of supplying good.

# Sensible security policies for Internet vendors and buyers leads to various requirements

- Authenticate the vendor to the buyer
- Keep buyer's credit card details secure
- Ensure that content is delivered to the buyer
- Authenticate the identity of account holder before giving them access to their account

# Designing Secure Systems

- Immense strides have been made in recent years in the development of cryptographic techniques and their applications, yet design of secure systems remains an inherently difficult task.
  - Aim: exclude *all* possible attacks and loop holes.
  - This looks like programmer aiming to *exclude* all bugs from his/her program.
- Security is about avoiding disasters and minimizing mishaps. When designing for security it is necessary to assume the worst.
- The design of security system is an exercise in balancing costs against threats:
  - A cost (computational and network usage) is incurred for their use.
  - Inappropriately specified security measures may exclude legitimate users from performing necessary actions.

# Worst case assumptions and design guidelines

- Interfaces are exposed
  - DSs made up of processes with open interfaces
- Networks are insecure
  - Messages sources can be falsified.
- Limit the lifetime and scope of each secret
  - Passwords and keys validity – needs to be time restricted.
- Algorithms and code are available to hackers
- Attackers may have access to large resources
- Minimise the trusted base.

# Overview of Security Techniques

- Digital cryptography provides the basis for most computer security mechanisms, but it is important to note that *computer security* and *cryptography* are distinct subjects.
  - Cryptography is an art of encoding information in a format that only intended recipient can access.
  - Cryptography can be used to provide a proof of authenticity of information in a manner analogous to the use of signature in conventional transactions.
- We will focus more on security of distributed systems and applications rather than *cryptography* algorithms.

# Cryptography: Introduction

- Cryptography: encryption and decryption
- Encryption is the process of encoding a message in such a way as to hide its contents.
- Modern cryptography includes several secure algorithms for encrypting and decrypting messages. They are based on keys.
- A cryptography key is a parameter used in an encryption algorithm in such a way that the encryption cannot be reversed without a knowledge of the key.

# Hu Fu: a cryptography approach in traditional China

- Tiger symbol as a certificate of dispatching troops in Chinese history originated very early.

# Classes of Cryptography Algorithms

- There are two main classes:
  - **Shared Secret Keys:**
    - *The sender and recipient share a knowledge of the key and it must not be revealed to anyone.*
  - **Public/Private Key Pair:**
    - *The sender of a message uses a recipient's public key to encrypt the message.*
    - *The recipient uses a corresponding private key to decrypt the message.*
- Uses of Cryptography:
  - Secrecy and integrity (to stop eavesdropping and tampering) + also use redundant information (checksums) for maintaining integrity.
  - Authentication
  - Digital Signatures

# Security notations – Familiar names and notations in security literature

| | |
|---|---|
| $K_A$ | Alice's secret key |
| $K_B$ | Bob's secret key |
| $K_{AB}$ | Secret key shared between Alice and Bob |
| $K_{Apriv}$ | Alice's private key (known only to Alice) |
| $K_{Apub}$ | Alice's public key (published by Alice for all to read) |
| $\{M\}_K$ | Message $M$ encrypted with key $K$ |
| $[M]_K$ | Message $M$ signed with key $K$ |

| | |
|---|---|
| Alice | First participant |
| Bob | Second participant |
| Carol | Participant in three- and four-party protocols |
| Dave | Participant in four-party protocols |
| Eve | Eavesdropper |
| Mallory | Malicious attacker |
| Sara | A server |

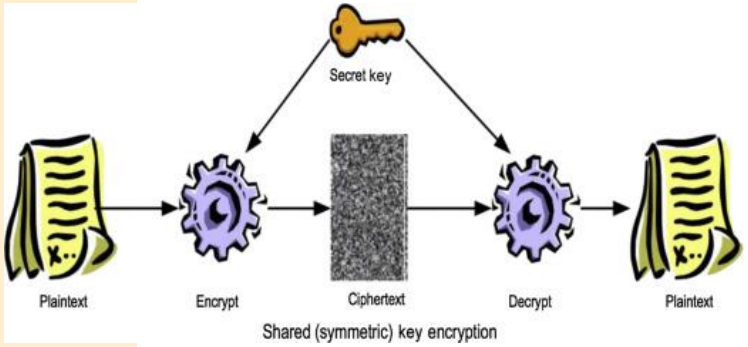
# Scenario 1:
# Secret communication with a shared secret key

Alice wishes to send some information secretly.

Alice and Bob share a secret key $K_{AB}$.

1. Alice uses $K_{AB}$ and an agreed encryption function $E(K_{AB}, M)$ to encrypt and send any number of messages $\{M_i\}_{K_{AB}}$ to Bob.
2. Bob reads the encrypted messages using the corresponding decryption function $D(K_{AB}, M)$.

Alice and Bob can go on using $K_{AB}$ as long as it is safe to assume that $K_{AB}$ has not been *compromised*.




## Issues:

*Key distribution*: How can Alice send a shared key $K_{AB}$ to Bob securely?

*Freshness of communication*: How does Bob know that any $\{M_i\}$ isn't a copy of an earlier encrypted message from Alice that was captured by Mallory and replayed later? Problem: if the message is a request to pay some money to someone. Mallory might trick Bob into paying twice?

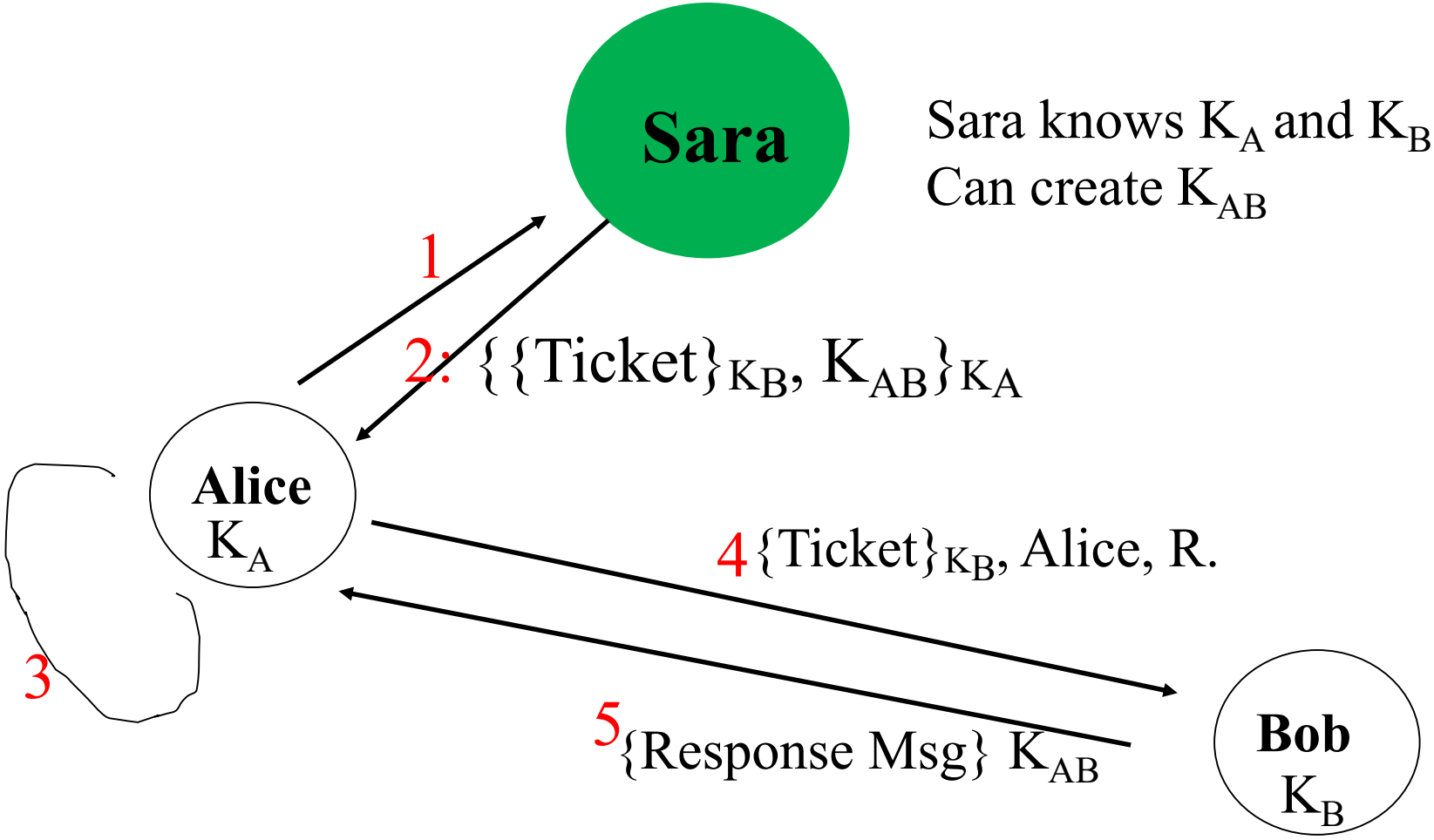
# Scenario 2: Authenticated communication with a server (Server knows secret keys of all parties)

Bob is a file server; Sara is an authentication service. Sara shares secret key $K_A$ with Alice and secret key $K_B$ with Bob.

1. Alice sends an (unencrypted) message to Sara stating her identity and requesting a *ticket* for access to Bob. ➲
2. Sara sends a response to Alice. $\{\{Ticket\}_{K_B}, K_{AB}\}_{K_A}$. It is encrypted in $K_A$ and consists of a ticket (to be sent to Bob with each request for file access) encrypted in $K_B$ and a new secret key $K_{AB}$.
3. Alice uses $K_A$ to decrypt the response.
4. Alice sends Bob a request R to access a file: $\{Ticket\}_{K_B}$, Alice, R.
5. The ticket is actually $\{K_{AB}, Alice\}_{K_B}$. Bob uses $K_B$ to decrypt it, checks that Alice's name matches and then uses $K_{AB}$ to encrypt responses to Alice.

- This is a simplified version of the Needham and Schroeder (and Kerberos) protocol.
- Timing and replay issues – addressed in N-S and Kerberos.
- Not suitable for e-commerce because authentication service doesn't scale…

# Scenario 2: Illustration




The ticket is actually $\{K_{AB}, Alice\}_{K_B}$.

# Limitation of Needham and Schroeder Protocols

- It depends upon prior knowledge by the authentication server Sara of Alice's and Bob's keys. This is feasible in a single organisation where Sara run a physically secure computer and is managed by a trusted principal.
  - Not suitable in E-commerce or other wide area applications.
- Usefulness of challenges: They introduced the concept of a *cryptographic challenge*. That means in step 2 of our scenario, where Sara issues a ticket to Allice encypted in Alice's secret key, $K_A$ .

# Scenario 3:
# Authenticated communication with public keys

Bob has a public/private key pair <$K_{Bpub}$, $K_{Bpriv}$> & establishes $K_{AB}$ as follows:

1. Alice obtains a certificate that was signed by a trusted authority stating Bob's public key $K_{Bpub}$
2. Alice creates a new shared key $K_{AB}$ , encrypts it using $K_{Bpub}$ using a public-key algorithm and sends the result to Bob.
3. Bob uses the corresponding private key $K_{Bpriv}$ to decrypt it.

(If they want to be sure that the message hasn't been tampered with, Alice can add an agreed value to it and Bob can check it.)

- Mallory might intercept Alice's initial request to a key distribution service for Bob's public-key certificate and send a response containing his own public key. He can then intercept all the subsequent messages.

# Scenario 4:
# Digital signatures with a secure digest function

Alice wants to publish a document M in such a way that anyone can verify that it is from her.

1. Alice computes a fixed-length digest of the document Digest(M).
2. Alice encrypts the digest in her private key, appends it to M and makes the resulting signed document (M, $\{Digest(M)\}_{K_{Apriv}}$) available to the intended users.
3. Bob obtains the signed document, extracts M and computes Digest(M).
4. Bob uses Alice's public key to decrypt $\{Digest(M)\}_{K_{Apriv}}$ and compares it with his computed digest. If they match, Alice's signature is verified.

# A Certificate with Digital signatures

**Transcript**

Student: Minxian Xu

1. COMP90015: Distributed Systems: 95 marks

2….

Computed message digest:

$\{491103ea18660f4c4d9f3c32af5d28f5\}_{K_{UniMelbpriv}}$

IBM (employer) uses UniMelb's public key to decrypt $\{Digest(M)\}_{K_{UniMelbpriv}}$ and compares it with his computed digest of Transcript content. If they match, UniMelb's signature is verified. Then IBM trusts and hires you☺

Algo MD5 128 bit from: https://www.freeformatter.com/message-digest.html

The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message.

# Cryptographic Algorithms

Message M, key K, published encryption functions E, D

- Symmetric (secret key)

$$E(K, M) = \{M\}_K \qquad D(K, E(K, M)) = M$$

Same key for E and D

M must be hard (infeasible) to compute if K is not known.

Usual form of attack is brute-force: try all possible key values for a known pair M, $\{M\}_K$. Resisted by making K sufficiently large ~ 128 bits

- Asymmetric (public key)

Separate encryption and decryption keys: $K_e$, $K_d$

$$D(K_d. E(K_e, M)) = M$$

depends on the use of a *trap-door function* to make the keys. E has high computational cost. Very large keys > 512 bits

- Hybrid protocols - used in SSL (now called TLS)

Uses asymmetric crypto to transmit the symmetric key that is then used to encrypt a session.

# Public Key Infrastructure (PKI)

- PKI allows you to know that a given public key belongs to a given user
- PKI builds on asymmetric encryption:
  - Each entity has two keys: public and private
  - Data encrypted with one key can only be decrypted with other.
  - The private key is known only to the entity
- The public key is given to the world encapsulated in a X.509 certificate

# Public Key Infrastructure (PKI) Overview

- X.509 Certificates
- Certificate Authorities (CAs)
- Certificate Policies
  - Namespaces
- Requesting a certificate
  - Certificate Request
  - Registration Authority

# Certificates

Certificate: a statement signed by an appropriate authority.

Certificates require:

- An agreed standard format
- Agreement on the construction of chains of trust.
- Expiry dates, so that certificates can be revoked.

Public-key certificate for Bob's Bank

| | |
|---|---|
| 1. *Certificate type*: | Public key |
| 2. *Name*: | Bob's Bank |
| 3. *Public key*: | $K_{Bpub}$ |
| 4. *Certifying authority*: | Fred – The Bankers Federation |
| 5. *Signature*: | $\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$ |

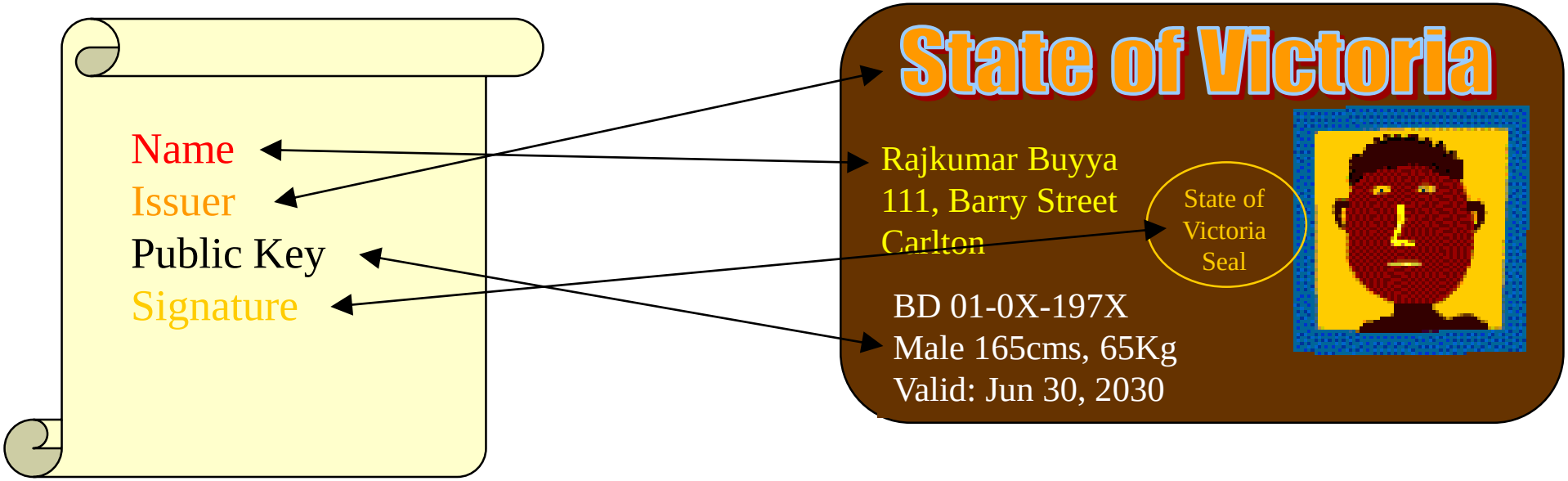
# X509 Certificate format

| | |
|---|---|
| *Subject* | Distinguished Name, Public Key |
| *Issuer* | Distinguished Name, Signature |
| *Period of validity* | Not Before Date, Not After Date |
| *Administrative information* | Version, Serial Number |
| *Extended Information* | |

It provides a public key to a named entity called the subject. The binding is in the signature, which is issued by another entity called issuer (CA, Certificate Authority)

# Certificates

- Similar to passport or driver's license



# An example of Certificate

**Validity Start**

**Certificate:**
**Data:**
**Version: 3 (0x2)**
**Serial Number: 28 (0x1c)**
**Signature Algorithm: md5WithRSAEncryption**
**Issuer: C=US, O=Globus, CN=Globus Certification Authority**
**Validity**
**Not Before: Apr 22 19:21:50 2020 GMT**
**Not After : Apr 22 19:21:50 2030 GMT**
**Subject:** /O=Grid/O=Globus/OU=cis.unimelb.edu.au/CN=Rajkumar Buyya
**Subject Public Key Info:**
**Public Key Algorithm: RSAEncryption**
**RSA Public Key: (1024 bit)**
**Modulus (1024 bit):**
**00:bf:4c:9b:ae:51:e5:ad:ac:54:4f:12:52:3a:69:**
**<snip>**
**b4:e1:54:e7:87:57:b7:d0:61**
**Exponent: 65537 (0x10001)**
**Signature Algorithm: md5WithRSAEncryption**
**59:86:6e:df:dd:94:5d:26:f5:23:c1:89:83:8e:3c:97:fc:d8:**
**<snip>**
**8d:cd:7c:7e:49:68:15:7e:5f:24:23:54:ca:a2:27:f1:35:17:**

# Certificates as credentials

- Certificates can act as *credentials*
  - Evidence for a principal's right to access a resource
- The two certificates shown in the last slide could act as credentials for Alice to operate on her bank account
  - She would need to add her public key certificate

Alice's bank account certificate

| | |
|---|---|
| 1. *Certificate type* | Account number |
| 2. *Name*: | Alice |
| 3. *Account*: | 6262626 |
| 4. *Certifying authority*: | Bob's Bank |
| 5. *Signature*: | $\{Digest(field\ 2 + field\ 3)\}_{K_{Bpriv}}$ |

Public-key certificate for Bob's Bank

| | |
|---|---|
| 1. *Certificate type* | Public key |
| 2. *Name*: | Bob's Bank |
| 3. *Public key*: | $K_{Bpub}$ |
| 4. *Certifying authority*: | Fred – The Bankers Federation |
| 5. *Signature*: | $\{Digest(field\ 2 + field\ 3)\}_{K_{Fpriv}}$ |

# Access control

- Protection domain
  - A set of <resource, rights> pairs
- Two main approaches to implementation:
  - Access control list (ACL) associated with each object
    - *E.g. Unix file access permissions* ➲
    - *For more complex object types and user communities, ACLs can complex*

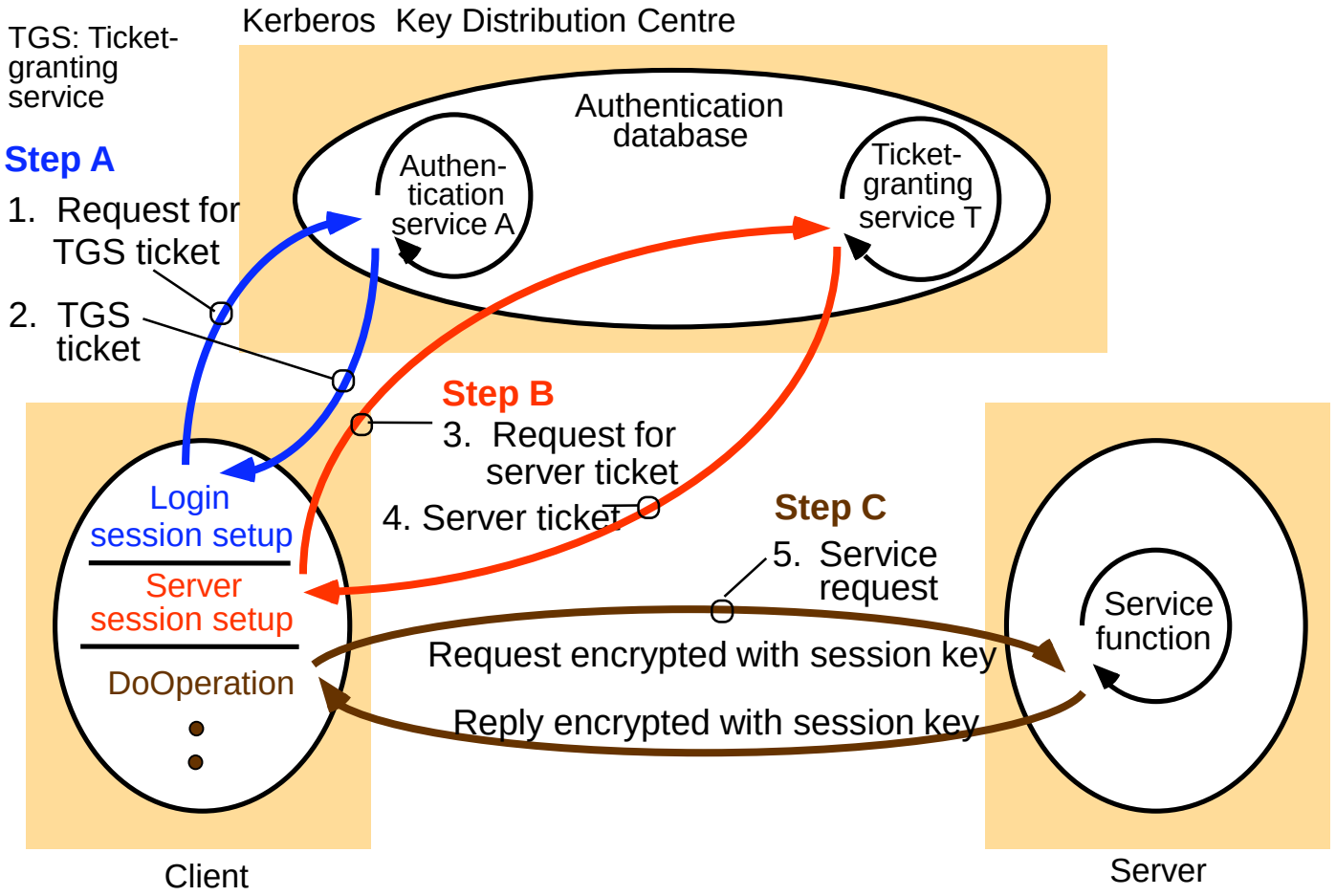
```
drwxr-xr-x  gfc22  staff       264 Oct 30 16:57 Acrobat User Data
-rw-r--r--  gfc22  unknown       0 Nov  1 09:34 Eudora Folder
-rw-r--r--  gfc22  staff    163945 Oct 24 00:16 Preview of xx.pdf
drwxr-xr-x  gfc22  staff       264 Oct 31 13:09 iTunes
-rw-r--r--  gfc22  staff       325 Oct 22 22:59 list of broken apps.rtf
```

  - Capabilities associated with principals
    - *Like a key – allowing the holder access to certain operations on a specified resource.*
    - *Format: <resource id, permitted operations, authentication code>*

# Case study: Kerberos authentication and key distribution service

- Secures communication with servers on a local network
  - Developed at MIT in the 1980s to provide security across a large campus network > 5000 users
  - based on Needham - Schroeder protocol
- Standardized and now included in many operating systems
  - Internet RFC 1510, OSF DCE
  - BSD UNIX, Linux, Windows 2000, XP, **Windows 8, Windows 10**
- Kerberos server creates a shared secret key for any required server and sends it (encrypted) to the user's computer
- User's password is the initial secret shared with Kerberos

# System architecture of Kerberos




Needham - Schroeder protocol

1. A->S: $A, B, N_A$
2. S->A: $\{N_A, B, K_{AB}, \{K_{AB}, A\}K_B\}K_A$
3. A->B: $\{K_{AB}, A\}K_B$
4. B->A: $\{N_B\}K_{AB}$
5. A->B: $\{N_B - 1\}K_{AB}$

**Step A** once per login session

**Step B** once per server session

**Step C** once per server transaction

# Kerberized NFS

- Kerberos protocol is too costly to apply on each NFS operation
- Kerberos is used in the mount service:
  - to authenticate the user's identity
  - User's UserID and GroupID are stored at the server with the client's IP address
- For each file request:
  - UserID and GroupID are sent encrypted in the shared session key
  - The UserID and GroupID must match those stored at the server
  - IP addresses must also match
- This approach has some problems
  - can't accommodate multiple users sharing the same client computer
  - all remote filestores must be mounted each time a user logs in

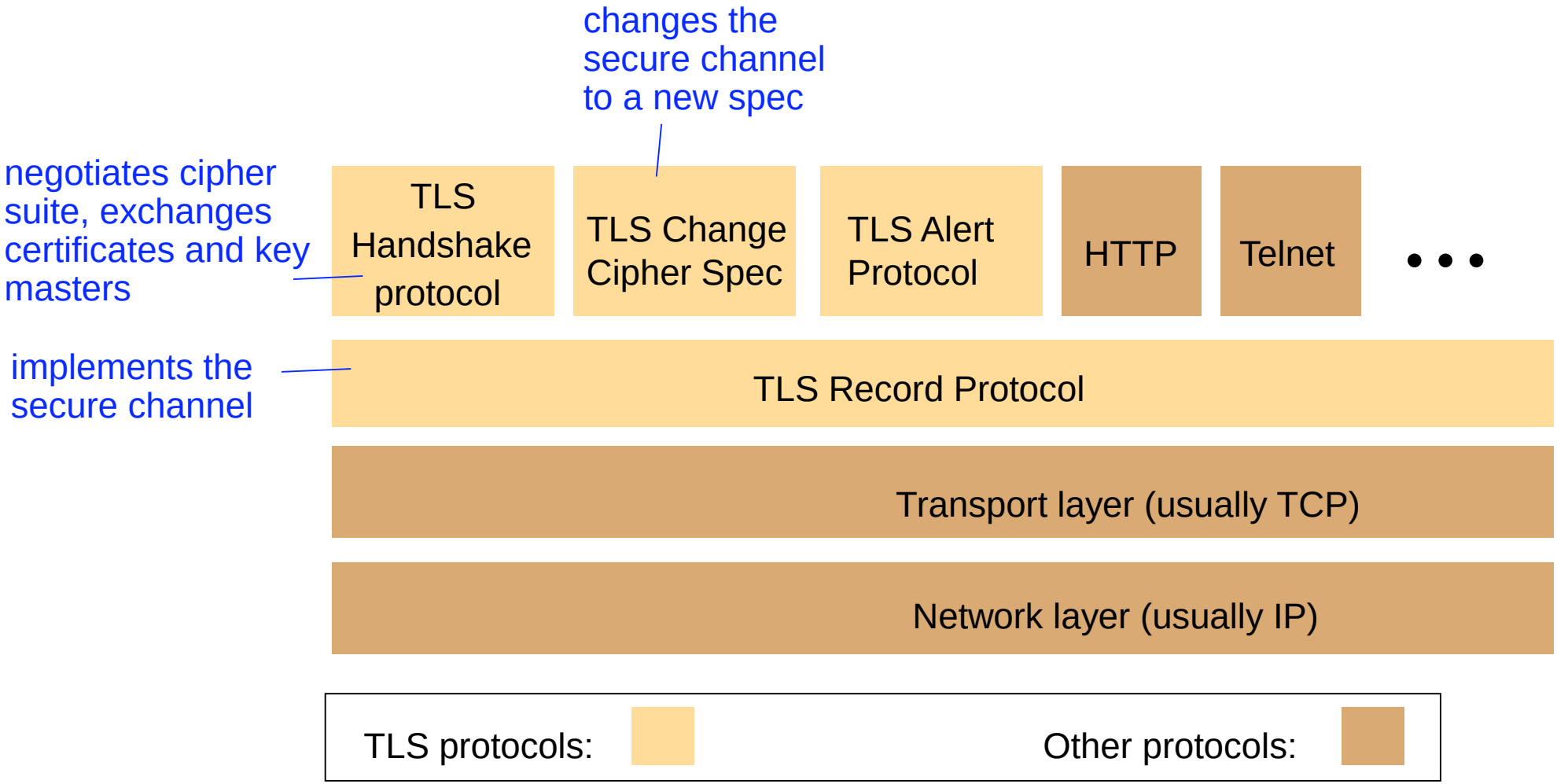
# Case study: The Secure Socket Layer (SSL)

- Key distribution and secure channels for Internet commerce
  - Hybrid protocol; depends on public-key cryptography
  - Originally developed by Netscape Corporation (1994) and supported by most browsers and is widely used in Internet commerce.
  - Extended and adopted as an Internet standard with the name **Transport Level Security (TLS)** – RFC 2246
  - Provides the security in all web servers and browsers and in secure versions of Telnet, FTP and other network applications
- Key Feature
  - **Negotiable encryption and authentication algorithms**. In an open network we should NOT assume that all parties use the same client software or all client/server software includes a particular encryption algorithms.
- Design requirements
  - Secure communication without prior negotiation or help from 3rd parties
  - Free choice of crypto algorithms by client and server
  - communication in each direction can be authenticated, encrypted or both

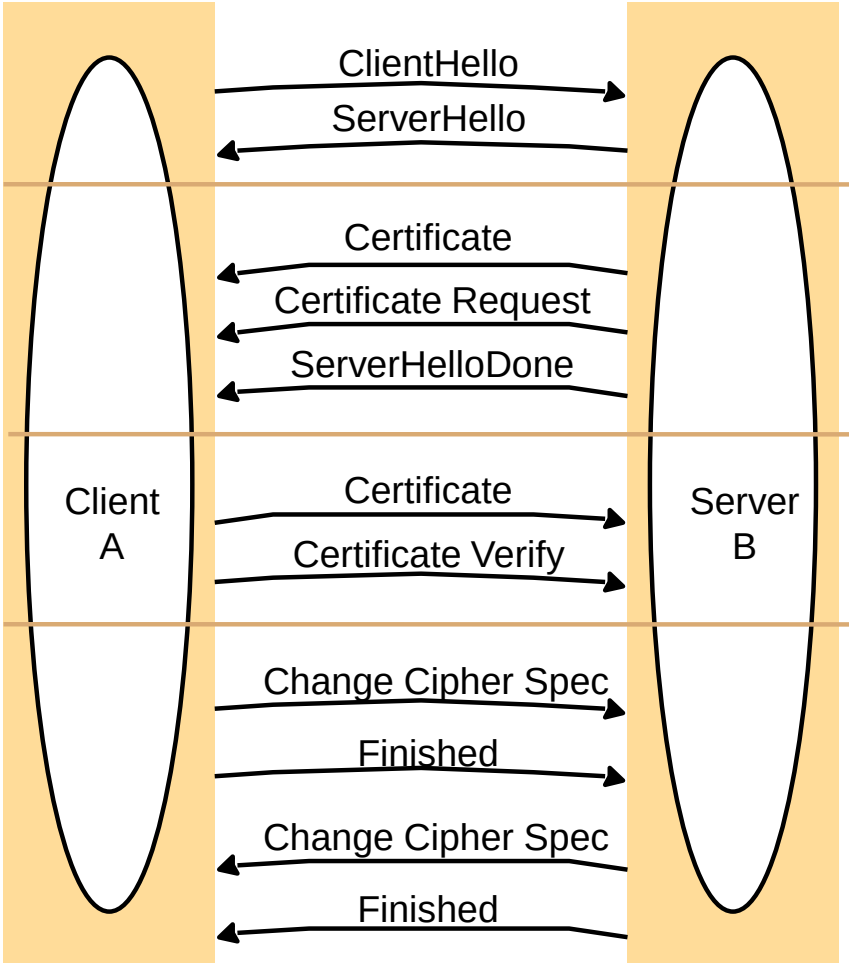
# Bootstrapped secure communication

- To meet the need for secure communication without previous negotiation/help from 3rd parties, the secure channel is established using a hybrid schemes.
- The secure channel is fully configurable.
- The details of TLS protocols are standardized and several software libraries and toolkits are available to support it [www.openssl.org]
- TLS consists of two layers:
  - **TLS Record Protocol Layer**: implements a secure channel, encrypting and authenticating messages transmitted through any connection oriented protocol. It is realized at session layer.
  - **Handshake Layer**: Containing Handshake protocol and two other related protocols that establish and maintain a TLS session (i.e., secure channel) between client and server.
  - Both are implemented by software libraries at application level in the client and the server.

# TLS protocol stack



# TLS/SSL handshake protocol
# (Handshake is performed over an existing connection)




Establish protocol version, session ID, cipher suite, compression method, exchange random start values

Optionally send server certificate and request client certificate

Send client certificate response if requested

Change cipher suite and finish handshake

Includes key master exchange.
Key master is used by both A and B to generate:

*2 session keys* $K_{AB}$ $K_{BA}$

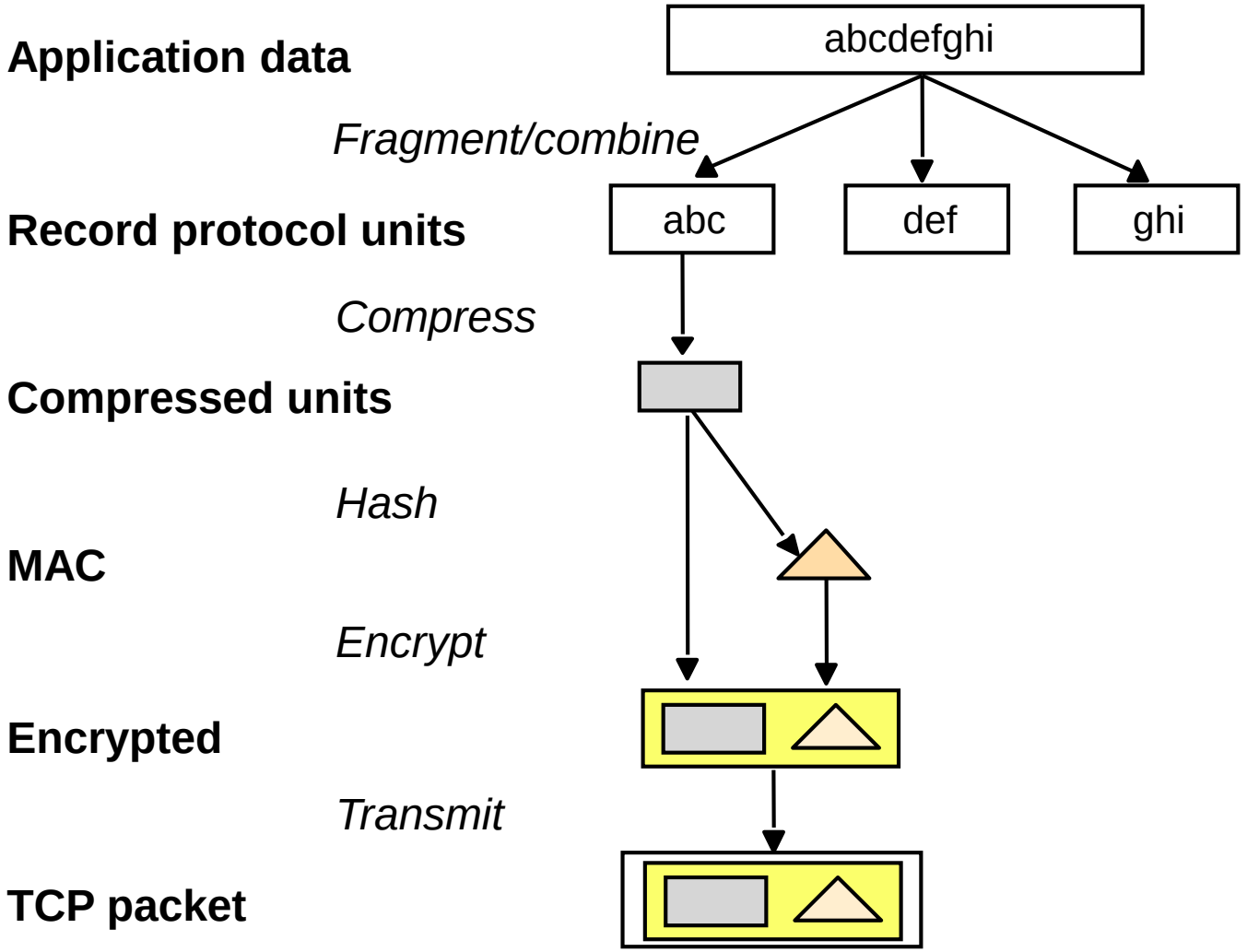
*2 MAC keys* $M_{AB}$ $M_{BA}$

Message Authentication Code (**MAC**)

# TLS handshake configuration options

| *Component* | *Description* | *Example* |
|---|---|---|
| Key exchange method | the method to be used for exchange of a session key | RSA with public-key certificates |
| Cipher for data transfer | the block or stream cipher to be used for data | IDEA (International Data Encryption Algorithm) |
| Message digest function | for creating message authentication codes (MACs) | SHA (Secure Hash Algorithm) |

# TLS record protocol operation: a pipeline for data transformation



# Summary

- Threats for the security in distributed systems are pervasive.
- It is essential to protect the resources, communication channels and interfaces of distributed systems and applications against attacks.
- This is achieved by the use of access control mechanisms and secure channels.
- Public-key and secret-key cryptography provide the basis for authentication and for secure communication.
- Kerberos and SSL are widely-used system components that support secure and authenticated communication.

- Demo

Client-Server Encryption