# Lecture
# CMPU4021 Distributed Systems

Distributed Systems Security

## Security Goals

- Keep systems, programs, and data secure

- Three areas
  - Confidentiality
    - Keeping data & resources hidden
  - Integrity
    - Protecting against unauthorized changes to the data or resources
  - Availability
    - Ensuring that a system is accessible and capable of working to required performance specifications

## Security in distributed systems

- Two specific concerns that centralized systems do not have
  - use of a network
    - contents may be seen by other, possibly malicious, parties
  - use of servers
    - That authenticate the client and control access to services
    - physical access to the system and the security controls - unknown to the client

## Security models

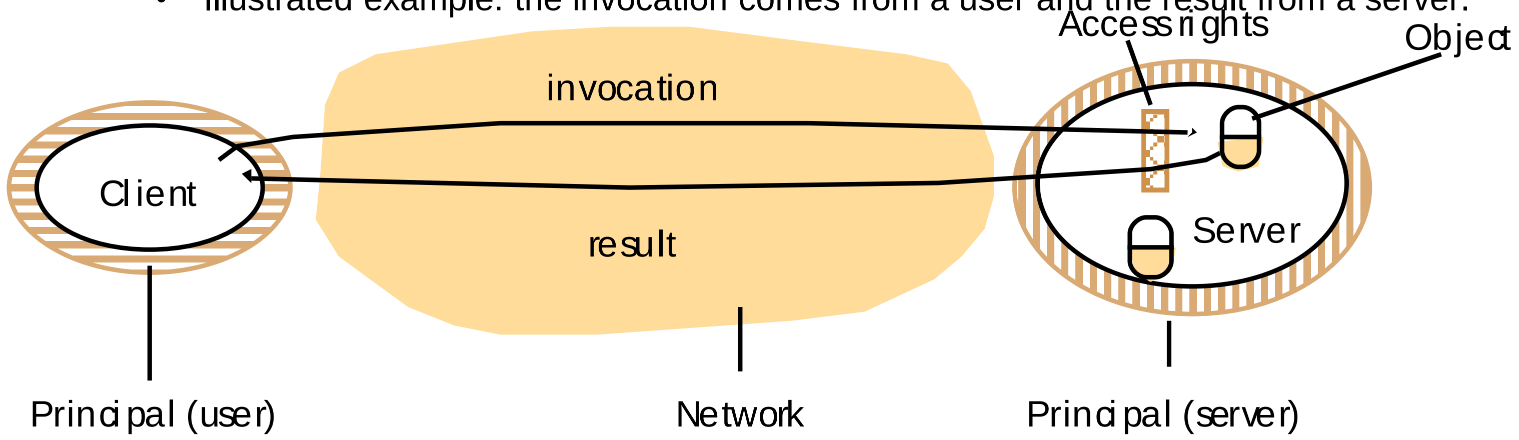
- The security model provides the basis for the analysis and design of secure systems in which these costs are kept to a minimum.

- This analysis involves the construction of a *threat model*
  - listing all the forms of attack to which the system is exposed; and
  - an evaluation of the risks and consequences of each.

# Security model

- Security model:
  - The security of a distributed system can be achieved by securing processes and communication channels, and
  - by protecting objects (or resources of all types) they encapsulate against unauthorized access.

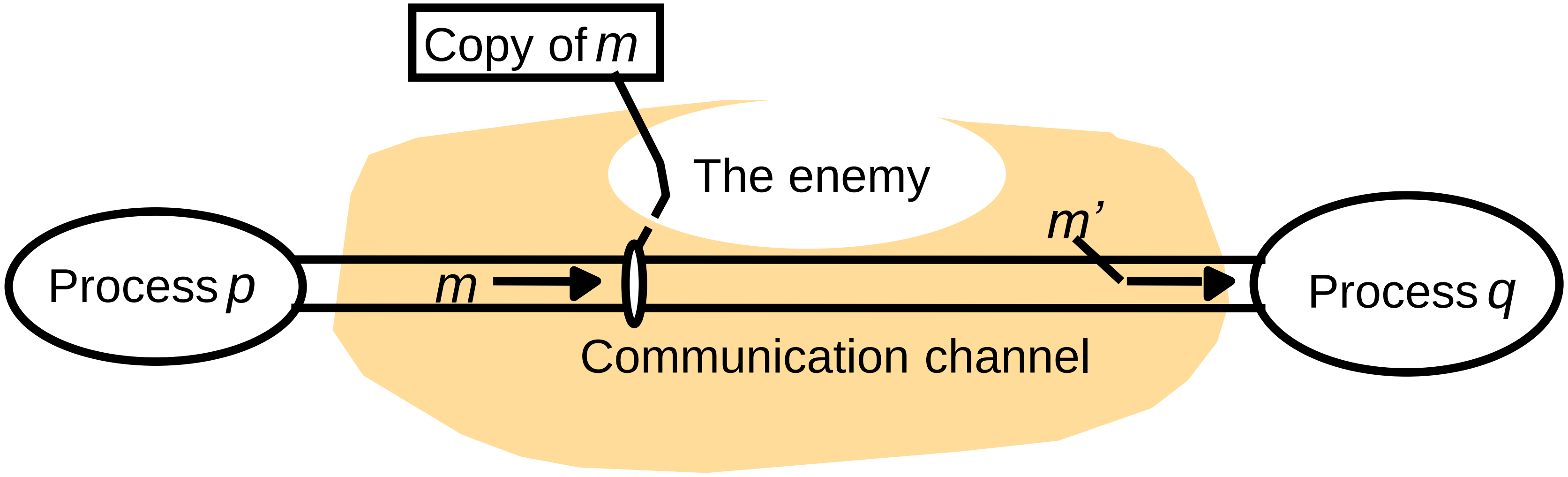
- Protecting objects/resources
  - Uses *access rights* - specify who is allowed to perform the operations of an object/resource
    - E.g., who is allowed to read or to write its state.
  - A principal (the authority issuing access rights) - may be a user or a process.
    - Illustrated example: the invocation comes from a user and the result from a server.



# Security model: Securing processes and their interactions

- Enemy model:
  - Eavesdropping via message interception, repeated trials of network access.
  - Threats to processes:
    - inserting and forwarding incorrect IP addresses in message to servers or clients to confuse or disguise the enemy-sender.
  - Threats to communication channels:
    - copying, altering or inserting incorrect data into message streams to deceive or replicate unauthorized transactions (e.g. banking)

- These threats can be defeated by the use of *secure channels*
  - *which are based on cryptography and authentication*

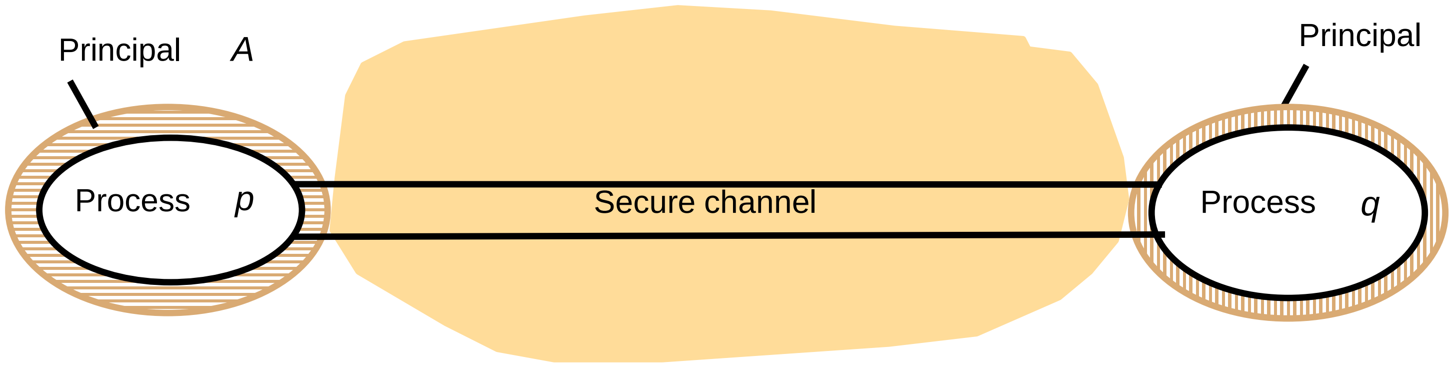


# Security model: Defeating security threats

- *Cryptography and shared secrets*
  - Cryptography – the science of keeping messages secure
  - Encryption – the process of scrambling a message in such a way as to hide its contents

- *Authentication*
  - proving the identities supplied by their senders.
  - establishing and verifying the identity of a user (or a service, process, or server).

- Secure channels
  - Encryption and authentication are used to build secure channels as a service layer on top of existing comms services. A secure channel is a communication channel connecting a pair of processes, each of which acts on behalf of a principal.



## Security model

- Other possible threats from an enemy
  - *Denial of service*
    - excessive and pointless invocations on services or message transmissions in a network, resulting in overloading of physical resources (network bandwidth, server processing capacity).
  - *Mobile code*
    - a Trojan horse role, purporting to fulfil an innocent purpose, but in fact including code that accesses or modifies resources that are legitimately available to the host process, but not to the originator of the code.

# Secure Channels

- Each of the processes knows reliably the identity of the principal on whose behalf the other process is executing.
  - This enables the server to protect its objects correctly and allows the client to be sure that is receiving results from a bona fide server.

- Ensure the privacy and integrity (protection against tampering) of the data transmitted across it.

- Each message includes a physical or logical time stamp to prevent messages from being replayed or reordered.

- Examples:
  - Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols which provide a secure channel between two communication peers.

## Authentication and Authorisation: Difference

- Authentication
  - Get and verify user’s identity

- Authorization
  - deciding if
    - to allow access to the service or its resources
    - what type of access is permitted

# Confidential group communication

- All group members share the **same** secret key
  - Used to encrypt and decrypt all messages transmitted between group members.
  - All members are trusted to indeed keep the key a secret
  - Vulnerable to attacks compared to two-party secure channels.

- Use a **separate** shared secret key between each pair of group members
  - As soon as one member turns out to be leaking information, the others can stop sending messages to that member, but still use the keys they were using to communicate with each other.
  - Instead of having to maintain one key, it is now necessary to maintain $N(N - 1)/2$ keys, which may be a difficult problem by itself.
  - Using a **public-key cryptosystem** can improve matters
    - In that case, each member has its own *(public key, private key)*, pair, in which the public key can be used by all members for sending confidential messages. In this case, a total of *N* key pairs are needed. If one member ceases to be trustworthy, it is simply removed from the group without having been able to compromise the other keys.
  - Key management
    - Establishing and distributing keys is not a trivial matter.

## PUBLIC KEY AUTHENTICATION

## Public key cryptography

- Does not require the parties to share a secret key.

- A message encrypted with your *private* key can be decrypted only with your *public* key.
  - Anyone can perform the decryption - you are the only one who could do the encryption
    - authentication

- A message encrypted with your *public* key can be decrypted only with your *private* key.
  - Anyone can do the encryption - you are the only one that is able to decrypt the message
    - confidentiality and secure communication.

- Public key encryption algorithms examples:
  - RSA (Rivest–Shamir–Adleman) and ECC (Elliptic Curve Cryptography).

# Public Key Authentication

- Demonstrate we can encrypt or decrypt a *nonce* - a random set of bits
  - This shows we know the key

Alice wants to authenticate herself to Bob:

- Bob: generates nonce, *S*
  - Sends it to Alice
- Alice: encrypts *S* with her private key (signs it)
  - Sends result to Bob

# Public key authentication

Bob:

1. Look up “Alice” in a database of public keys
2. Decrypt the message from Alice using Alice’s public key
3. If the result is S, then Bob is sure that he is talking with Alice

For *mutual authentication*

Alice has to present Bob with a nonce that Bob will encrypt with his private key and return

# Public key authentication

- Public key authentication
  - Binding identity = to a public key
  - Identities can be created by generating random [private, public] key pairs

- How do you know it really is Alice’s public key?
  - One option:
    - Get keys from a trusted source
    - Problem: requires always going to the source
      - Can not pass keys around
    - Another option:
      - Sign the public key – that protects it
      - Create digital certificate

## Digital certificates

- Data structure that contains
  - user information
  - the user’s public key
  - a signature of the certification authority

## Certificate standards and certificate authorities

- X.509 is the most widely used standard format for public key certificates

- 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* | |

## Reminder

- Digital signature
  - Hash of a message encrypted with the signer's private key

- A hash function
  - used to only verify the message integrity
  - takes an input (or 'message') and returns a fixed-size string of bytes
    - the string is called the 'hash value', 'message digest', 'digital fingerprint', 'digest' or 'checksum'.
  - if a message changes, the **hash of** a message will change

# Public Key Infrastructure

- Public Key Infrastructure (PKI) is a term used for a framework that enables secure exchange of information based on public key cryptography.

- It allows identities (of people, organizations, etc.) to be bound to digital certificates and provides a means of verifying the authenticity of certificates.

- PKI encompasses
  - Keys
  - Certificates
  - Public key encryption, and
  - trusted Certification Authorities (CAs) who generate and digitally sign certificates.

# What Applications use Certificates?

- Web browsers
  - X.509 certificates with Transport Layer Security (TLS)
  - Code-signing schemes
    - signed Java ARchives, and Microsoft Authenticode.
  - Secure E-Mail standards
    - Privacy Enhanced Mail (PEM) and Secure/Multipurpose internet Mail Extensions (S/MIME).

## How do I Get a Certificate?

- Create one yourself
  - using the right tools, such as keytool

- Ask a Certification Authority to issue you one

## Secure Communication with TLS

## Transport Layer Security (TLS)

- Provides authentication, integrity, and encrypted communication
- 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
- Most widely used to secure HTTP interactions for use in Internet commerce and other security-sensitive applications.

## TLS main features

- Negotiable encryption and authentication algorithms:
  - algorithms negotiated during the initial handshake.
- Bootstrapped secure communication:
  - Unencrypted communication is used for the initial exchanges, then public key-key cryptography and finally switching to secret-key cryptography once a shared key has been established.

- Protocol prefix *https*:
  - in URLs initiates the establishment of an TLS secure channel between a browser and a web server.

## TLS Protocols

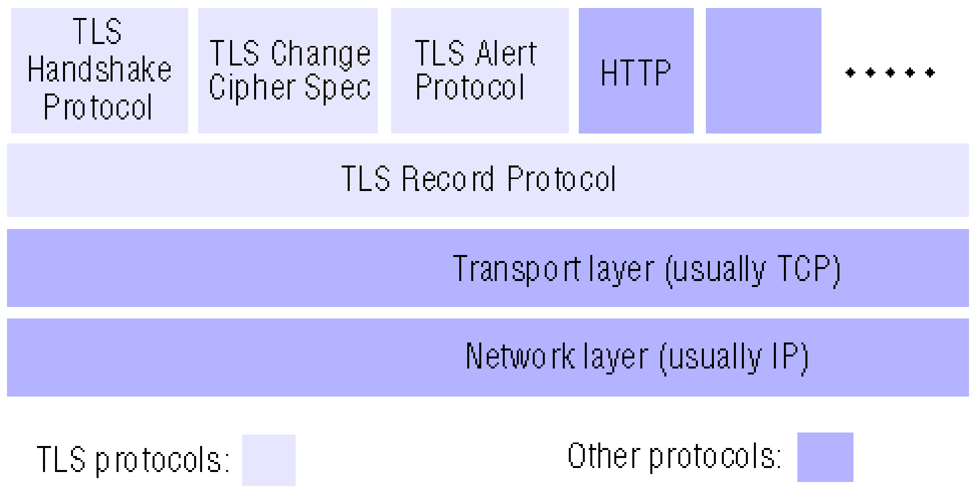
1. Authenticate & establish key
   - Authentication
     - Public keys (X.509 certificates and RSA or Elliptic Curve cryptography)
   - Key exchange options
     - Ephemeral Diffie-Hellman keys (generated for each session)
     - RSA public key, Elliptic Curve public key
     - Pre-shared key

2. Communicate
   - Data encryption options – symmetric cryptography
     - AES GCM, AES CBC, ARIA (GCM/CBC), ChaCha20-Poly1305, …
   - Data integrity options – message authentication codes
     - HMAC-SHA1, HMAC-SHA256/384, …

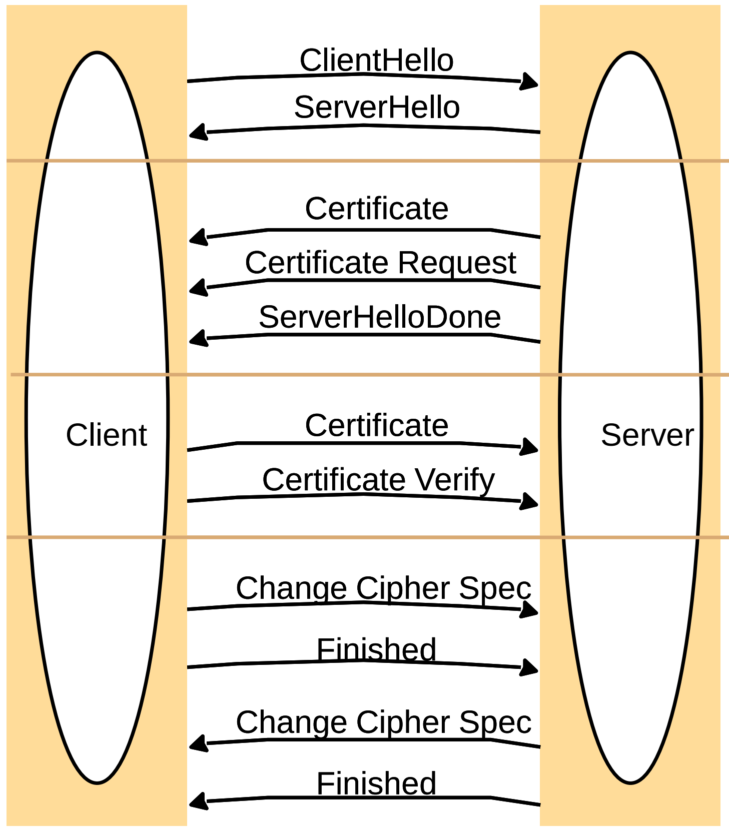
# TLS Layers

- Two layers:
  - *Record Protocol layer* - which implements a secure channel, encrypting and authenticating messages transmitted through any connection-oriented protocol

  - *Handshake layer* – containing the SSL handshake protocol and two other related protocols that establish and maintain an SSL session (that is, a secure channel).



# TLS handshake protocol



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* *2 MAC keys* ☐

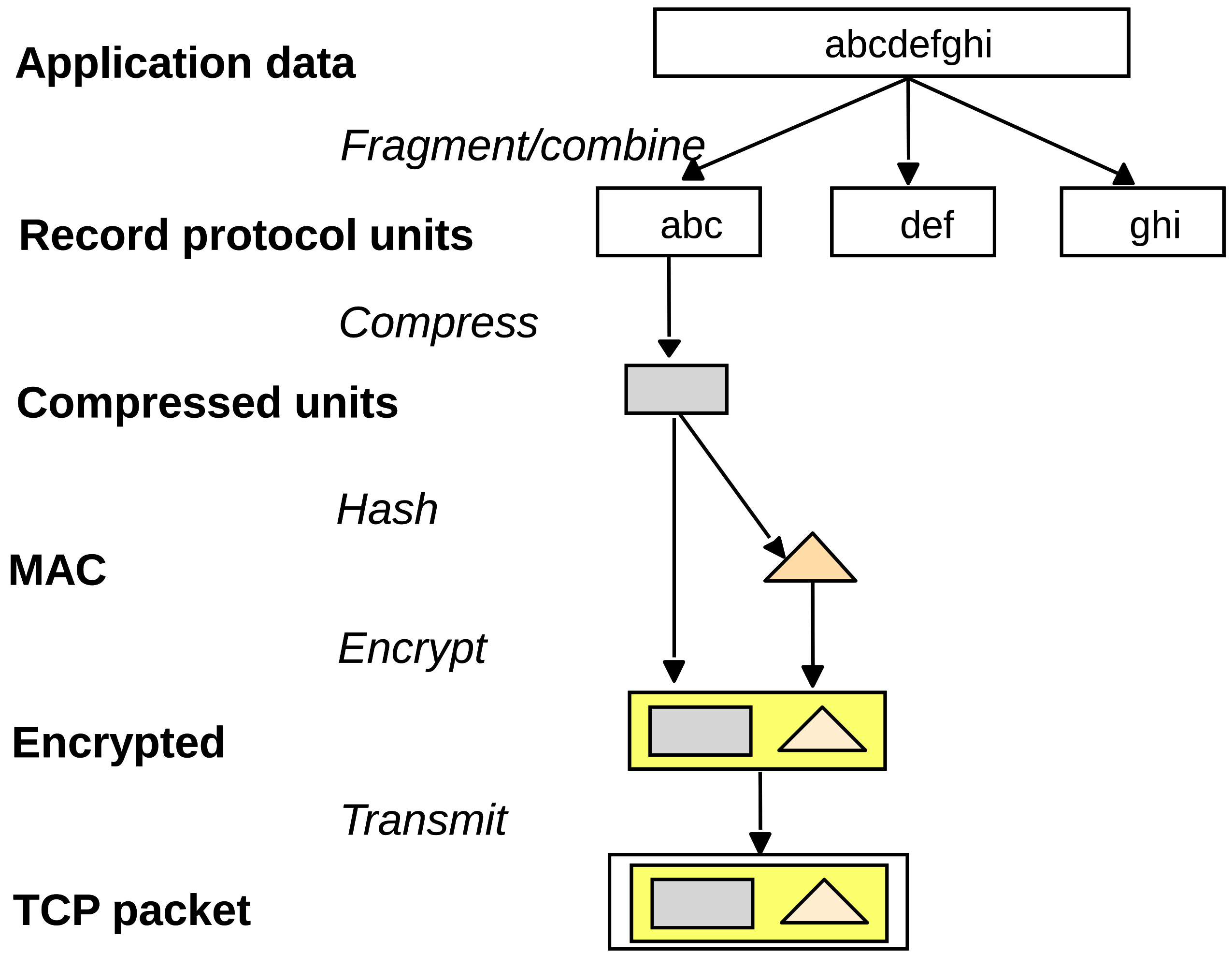
$K_{AB}$ $M_{AB}$

$K_{BA}$ $M_{BA}$

## TLS record protocol

- A message for transmission is first fragmented into blocks of manageable size.
- Then the blocks are optionally compressed.
- The encryption and message authentication (MAC) transformations deploy the algorithms specified in the agreed cipher suite.
- The signed and encrypted block is transmitted to the partner through associated TCP connection, where the transformations are reversed to produce the original data block.

# TLS record protocol



## TLS

- A practical implementation of a *hybrid* encryption scheme with authentication and key exchange based on public keys.
- Because the ciphers are negotiated in the handshake, it does not depend upon the availability of any particular algorithms, nor any secure services at the time of session establishment.
- The only requirement is for public-key certificates issued by an authority that is recognized by both parties.

# TLS

**Advantages**

- Validates the authenticity of the server
  - if you trust the CA
- Protects integrity of communications
- Protects the privacy of communications

**Disadvantages**

- Latency for session setup
- Older protocols had weaknesses
- Attackers also use TLS

# References

- Chapter 11 - Security: Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design