SECURITY

Dr. Padraig Corcoran

Security in distributed systems consists of two parts:

- **Communication security** - concerns communication between users/processes (principal mechanism - secure channel)
- **Authorisation security** - concerns processes only getting those access rights to resources they are entitled to (principal mechanism - access control).

Introduction to security

- A **security incident** is an event that occurs when an organisation's systems or data is compromised.
- A **security threat** is a potential cause of a security incident.
- A **risk** refers to the potential loss or damage resulting from a security incident.

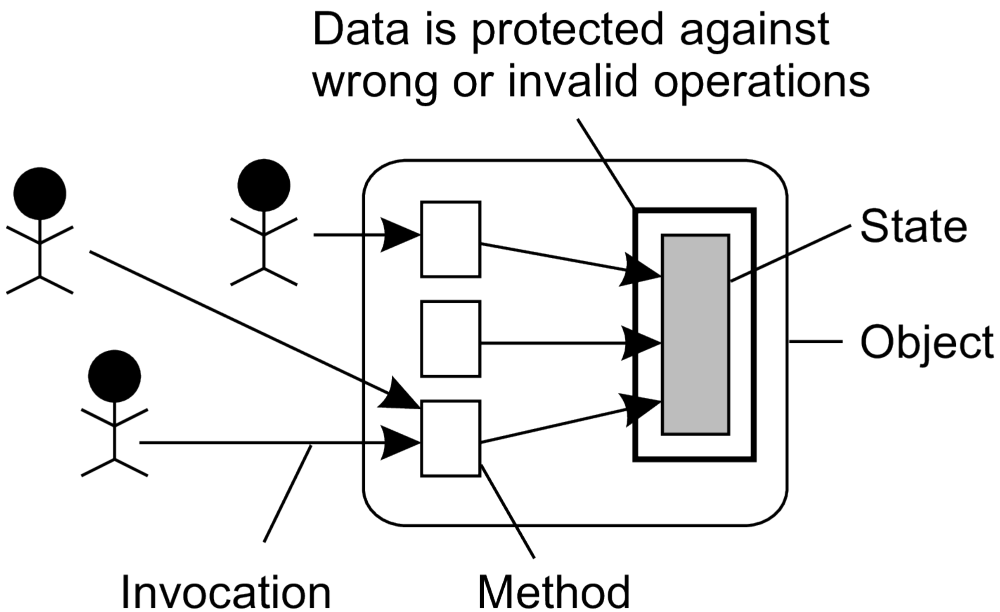
- There are four types of security threats to consider:
  - Interception - unauthorised party gains access to a service or data.
  - Interruption - services or data become unavailable, unusable, destroyed etc.
  - Modification - unauthorised changing of data or tampering with a service so that it no longer adheres to its original specifications.
  - Fabrication - additional data or activity are generated that would normally not exist.

- A **security policy** describes the actions entities in a system are allowed to take and which are prohibited.
- Entities include users, services, data, machines, etc.
- If enforced effectively this will reduce security threats.
- **Security mechanisms** are used to enforce security policies.

- Important security mechanisms are:
  - Encryption - transforms data into something an attacker cannot understand.
  - Authentication - used to verify the claimed identity of a user, client, server, host, or other entity.
  - Authorisation - check whether a client is authorised to perform the action requested.
  - Auditing - used to trace which clients accessed what, and in which way (does not provide protection but is useful).

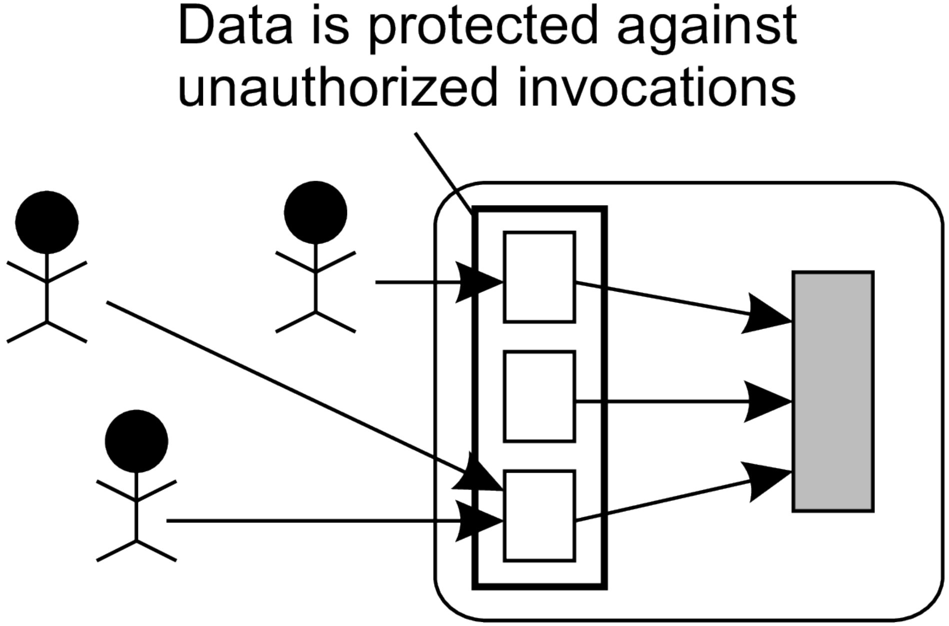
Focus of control

- Three approaches can be considered when protecting a system from security threats.
- First approach - security policy concentrates directly on the protection of data associated with the application.



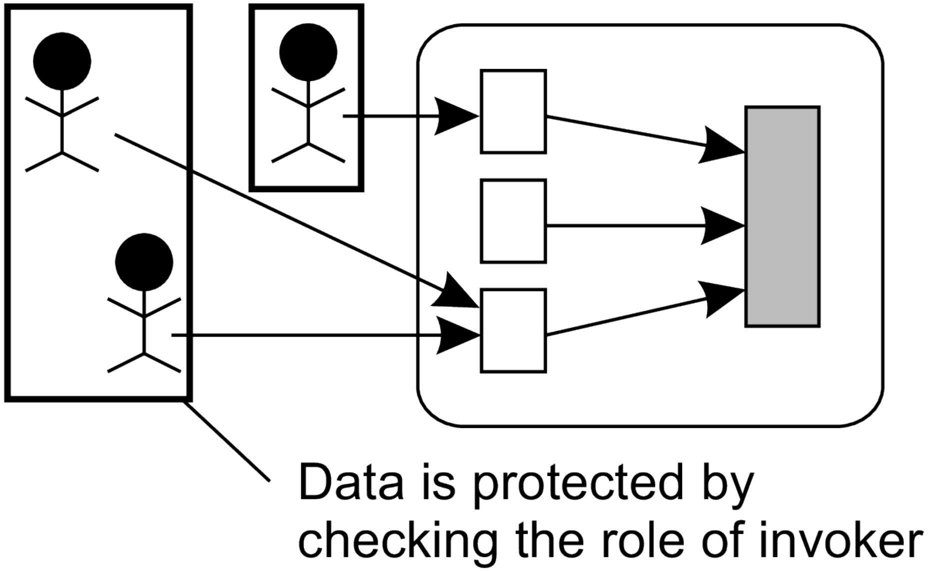
Protection against invalid operations

- Second approach - security policy concentrates on specifying which operations may be invoked, and by whom, when certain data/resources are to be accessed.



Protection against unauthorized invocations.

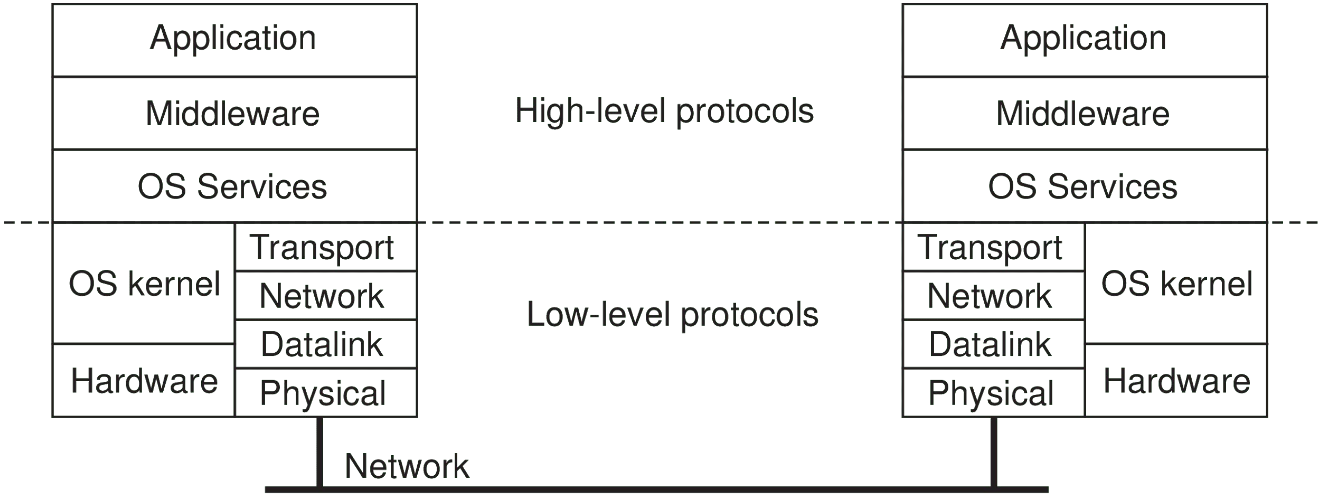
- Third approach - security policy concentrates on users by specifying who has access to the application, irrespective of operations they want to carry out.



Protection against unauthorised invocations.

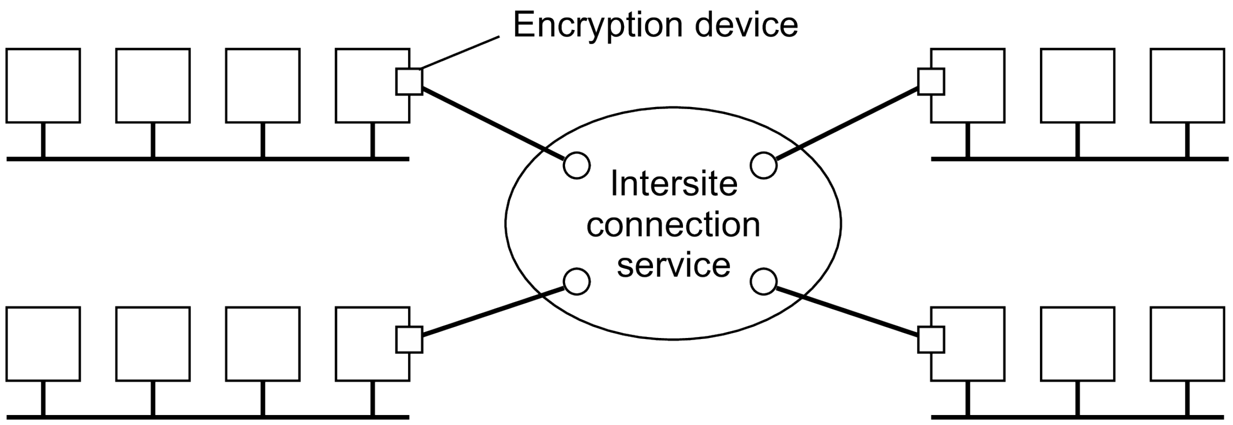
Layering of security mechanisms

- Given organisation of a system into layers, it must be decided at which level to place security mechanisms.
- In which layer security mechanisms are placed depends on trust in how secure the services are in a layer.



The logical organisation of a distributed system into several layers.

- Consider organisation located at different sites connected through a backbone connecting various LANS.
- Security provided by placing encryption devices at each backbone switch.
- If security of inter-site traffic is not trusted, one may use a **Transport Layer Security** (TLS) service.

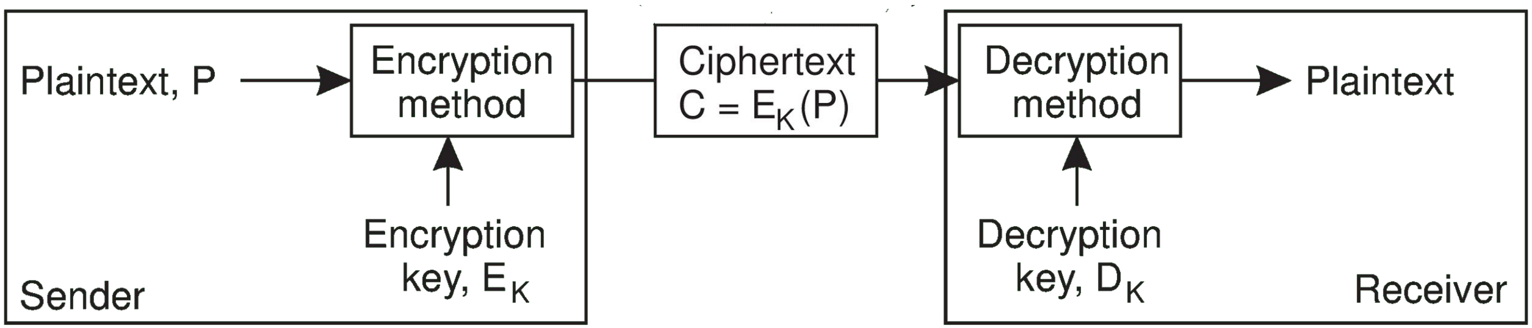


Several sites connected through a wide-area backbone service.

Secure communication channel using cryptography

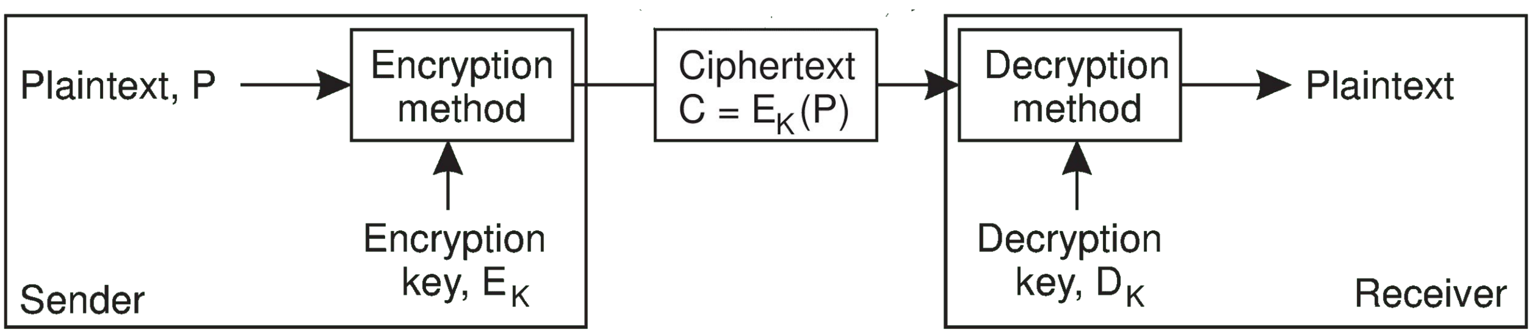
- Consider a sender S wanting to transmit a message m to receiver R.
- To protect against security threats, the sender **encrypts** m into an unintelligible message m’, and subsequently sends m’ to R.
- R, in turn, must **decrypt** the received message m’ into its original form m.

- Encryption and decryption are accomplished by using cryptographic methods parameterised by keys.
- The original form of the message sent is called the **plaintext** and encrypted form is called the **ciphertext**.



P and C equal the plaintext and ciphertext respectively.

- $C=E_K(P)$ denotes that the ciphertext C is obtained by encrypting the plaintext P using the key K.
- $P=D_K(C)$ denotes decryption of the ciphertext C using key K, resulting in the plaintext P.



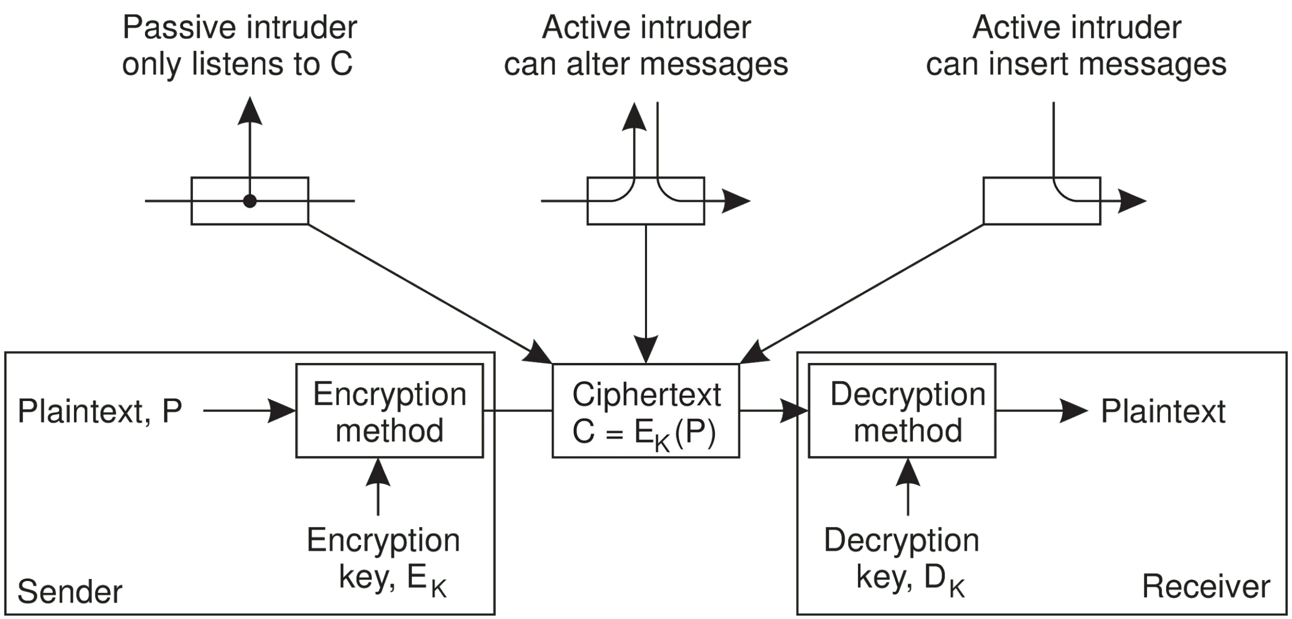
P and C equal the plaintext and ciphertext respectively.

- Caesar's Method is a simple encryption method.
- Each letter of the English alphabet is substituted by a new letter k places ahead of it.
- When k = 3, ‘A’ and ‘Y’ in plaintext become ‘D’ and ‘B’ respectively.
- Formally, we can represent the encryption and decryption mechanisms as:

$$E_K : L \rightarrow (L+k) \; mod \; 26$$
$$D_K : L \rightarrow (L-k) \; mod \; 26$$

where L denotes the numerical index of a letter.

- There are three attacks for which encryption helps:
  - Message is intercepted without either sender or receiver being aware that eavesdropping is happening.
  - The message is modified.
  - Message inserted into the communication system, attempting to make receiver believe this message came from sender.



Intruders and eavesdroppers in communication.

- A **cryptosystem** consists of three algorithms:
  - One for key generation.
  - One for encryption.
  - One for decryption.
- Distinction between cryptosystems based on whether or not the encryption and decryption keys are equal.

- In a **symmetric** (**secret-key**) **cryptosystem**, the same key is used to encrypt and decrypt a message:

$$P=D_K(E_K(P))$$

- Recall, E_K denotes encryption with key K, D_K denotes decryption with key K and P denotes the plaintext.
- Sender and receiver required to share the same key and this shared key must be kept secret.

- In an **asymmetric** (**public-key**) **cryptosystem**, the keys are different but form a unique pair:

$$P=D_{KD}(E_{KE}(P))$$

- Key pair KE and KD are used for encryption and decryption respectively.
- One of the keys in an asymmetric cryptosystem is kept private; the other is made public.
- Which one of the encryption or decryption keys that is actually made public depends on how the keys are used.

Example

- Alice wants to send a confidential message to Bob.
- She uses Bob's public key to encrypt the message.
- Bob is the only one holding the associated and private decryption key.
- Therefore, he is also the only person that can decrypt the message.

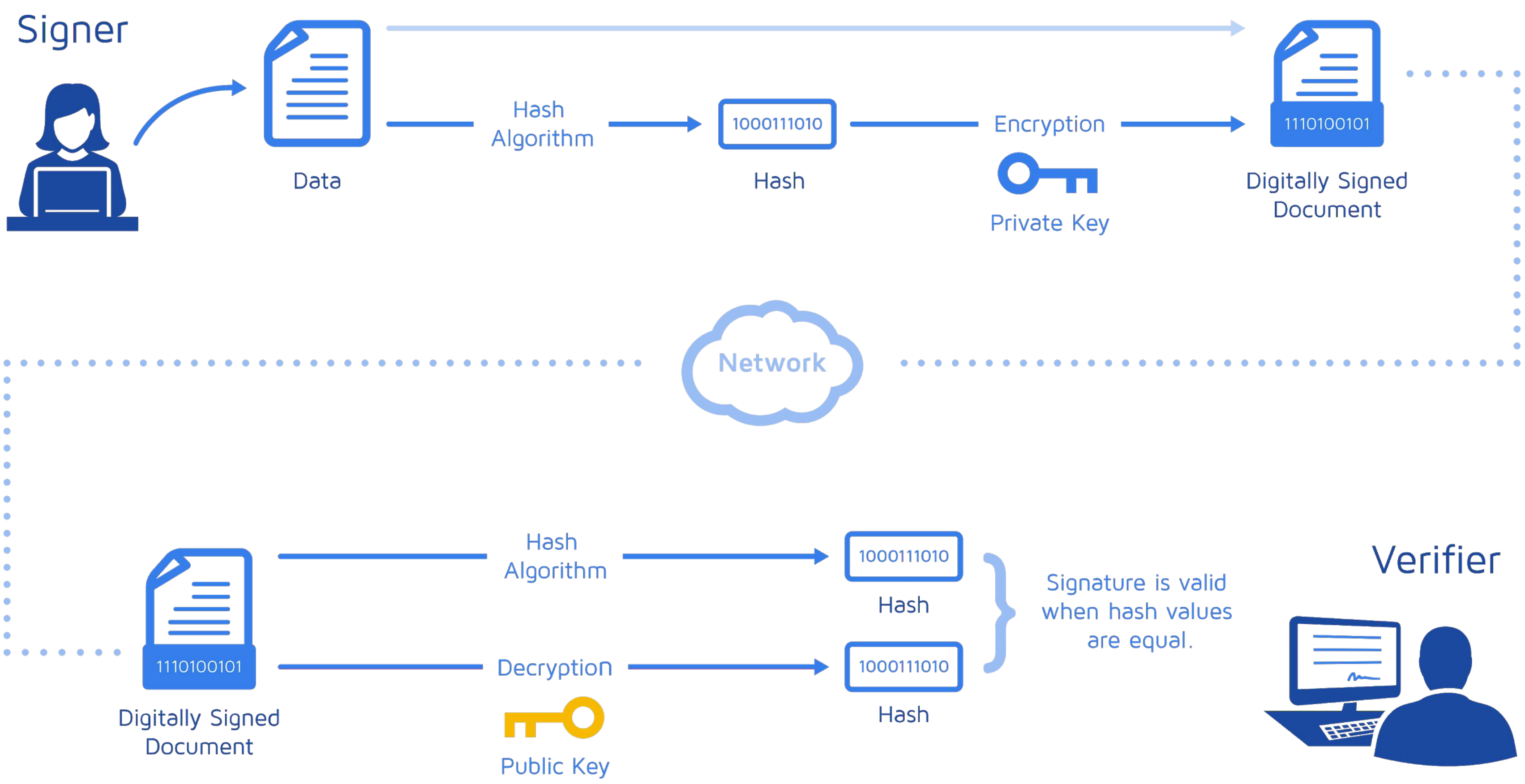
Example

- Bob wants to know for sure that the message he just received actually came from Alice. That is, perform **message authentication**.
- Alice can keep her encryption key private to encrypt the messages she sends.
- If Bob can decrypt a message using Alice’s public key the message must have come from Alice, because the decryption key is uniquely tied to the encryption key.

Digital signatures

- Bob agreed to sell an item to Alice for $500 via e-mail.
- Two issues that need to be considered regarding the integrity of the message are:
  - Alice needs to be assured that Bob will not change the $500 mentioned in her message into something higher.
  - Bob needs to be assured that Alice cannot deny ever having sent the message, for example, because she had second thoughts.
- Both issues can be dealt with if Alice **digitally signs** the message such that her signature is uniquely tied to its content.
- **Digital signatures** may be implemented using a public key cryptosystems.

- A hash function $H(.)$ reduces a variable length input to a fixed length output.
- A hash function will have the following properties:
    - Computed $m = H(M)$ is easy but computing the inverse is impossible.
    - If $M \neq M'$ then $H(M) \neq H(M')$ (strong collision resistance).



https://www.docusign.co.uk/how-it-works/electronic-signature/digital-signature/digital-signature-faq

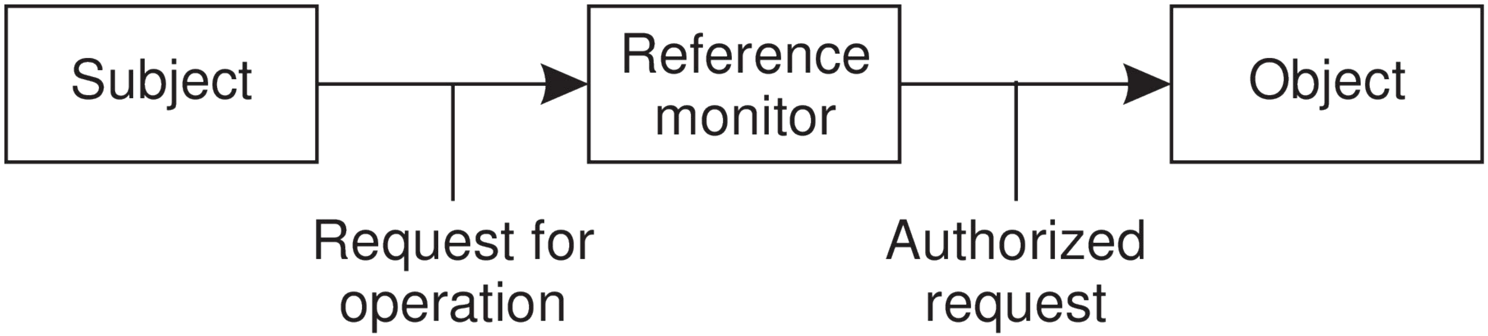
Theorem - If the two hashes are equal the message was sent by the person corresponding to the private-public key pair in question.

Proof - Someone encrypted the hash using their private. The private key in question belongs to that person. Therefore they are the only person who could have sent the message.

Access control

- Consider a client requesting that a remote object method is invoked.
- Such a request should be carried out only if the client has sufficient **access rights** for that invocation.
- Verifying access rights is known as **access control**.
- **Authorisation** concerns granting access rights.

- **Access control** concerns protecting an object against invocations by subjects that are not allowed to have specific (or even any) of the methods carried out.
- Commonly enforced using a **reference monitor** which
  - Records which subject may do what.
  - Decides whether a subject is allowed to have a specific operation carried out.



General model of controlling access to objects.

Access control matrix

- Access rights of subjects with respect to objects may be modelled using an **access control matrix**.
- Each subject is represented by a row in this matrix; each object is represented by a column.
- Given a matrix M, the entry M[s,o] lists operations subject s can request to be carried out on object o.

- Subjects: *p*, *q*
- Objects: *f*, *g*, *h*
- Operations: *r*, *w*, *x*, *a*, *o*

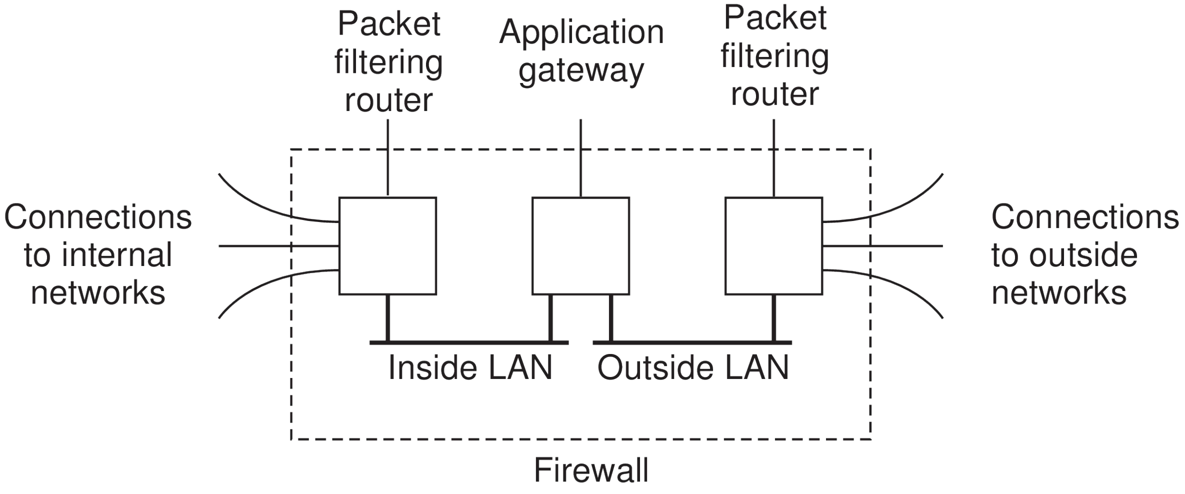
	f	*g*	*h*
p	*rwo*	*r*	*rwxo*
q	*a*		

Access control matrix

- Many entries in the matrix will be empty; a single subject will have access to relatively few objects.

Firewall

- A **firewall** acts as a **reference monitor** between a distributed system and other distributed systems.
- All messages are routed through the reference monitor and inspected before they are passed through.
- Unauthorized traffic is discarded and not allowed to continue.



A common implementation of a firewall.

- Two main types of firewalls - **packet-filtering gateway** and **application-level gateway**.
- Packet-filtering gateway - makes decisions based on the source and destination address in a packet header.
- Example usage - protect an internal server against requests from external clients.
- Application-level gateway - inspects the content of an incoming or outgoing message.
- Example usage - filtering spam e-mail.

Denial of service (DoS) attack

- An attack related to access control is preventing authorized processes from accessing resources.
- Usually implemented by an attacker sending a larger number of fake requests to a server.
- DoS attacks that come from a single process can often be handled effectively.
- In a **distributed denial of service** (**DDoS**) a set of processes jointly attempt to bring down a service.

- In many cases the attackers hijack a large group of machines which unknowingly participate in the attack.
- Such a set/network of machines is known as a **botnet**.

Secure mobile code

- Mobile code introduces a number of security threats; malicious code may corrupt its host.
- Represents an **access control** problem; program should not be allowed unauthorised access to host's resources.
- Security policies require we can allow access to local resources in a flexible, yet fully controlled manner.

- Security mechanisms to deal with threats and enforce policies:
  1. Execute code in a restricted environment called a **sandbox**.
  2. Ensure code **authenticity** and **integrity**; enforce a **security policy** based on where the program came from. Only trusted code is accepted.
- Authenticity - assures users that they know where the code came from.
- Integrity - verifies that the code hasn't been tampered with since its publication.
- Note, **code signing** may be used to ensure code authenticity and integrity.

Secure naming

- When a client retrieves an object based on a name, how does it know that it got back the correct object?
- We have three issues to worry about:
  - Validity: is the object returned a complete, unaltered copy of what was stored at the server?
  - Provenance: can the server that returned the object be trusted as a genuine supplier?
  - Relevance: is what was returned relevant considering what was asked?

- One solution to secure naming is to securely bind the name of an object to its content through hashing.
- That is, take the hash function H(O) as the name of the object O.
- Given this, one can check that the object has not be altered by comparing its name to its hash.
- This is a form of a **self-certifying name**.