Teórica – Prova 2

Certificados digitais

Um certificado enviado por uma **Autoridade de Certificados** (CA), autêntica a tua **chave pública**. Simplificando, um certificado é a tua chave pública assinada por uma chave privada de uma CA

Mecanismo de atestado:

- $\text{Sign}_{\text{Private key CA}}\ (\text{hash}(\text{User}_{\text{Public key}}))$

→ Para verificar utilizamos a chave pública da CA certificada.

Dados mínimos assinados pela CA

$$C_A = E(PR_{CA}, [T, ID_A, PU_A])$$

T – (Time stamp) Expiration date
ID – Identifier of the key owner
PU – User's Public key

} Hash is encrypted

The whole block encrypted with the CA's private key

Outros dados que são assinados por uma CA

- Chave pública do dono
- Números de série
- Organização que enviou o certificado e a sua assinatura

Verificação de certificados

Fazemo-lo ao decifrar o certificado com a chave pública da CA

Now we have the **same problem**: how to guarantee the Authenticity of CA's public key ?

Burned in your computer's Operating System…

(If the CA happens to be a root CA)

- When *A* presents his/her certificate to *B*
  - The latter can **verify** the **legitimacy** of the **certificate**
  - By, decrypting it with the **CA**'s public key

Successful decryption **authenticates** the certificate **issuing authority**

Authentication	This also provides *B* with **authentication** for *A*'s identity since only the real *A* could have provided a legitimate certificate with *A*'s identifier in it

Two parties *A* and *B* acquires the other party's public key **not directly** but through the other party's certificate

For greater security, *B* can ask **CA** to verify that the certificate received from *A* **is currently valid** — that is, it has not been **revoked**

Outras maneiras de confiar

Web of Trust (WoT)

Based on friends & friends of friends (for small communities)

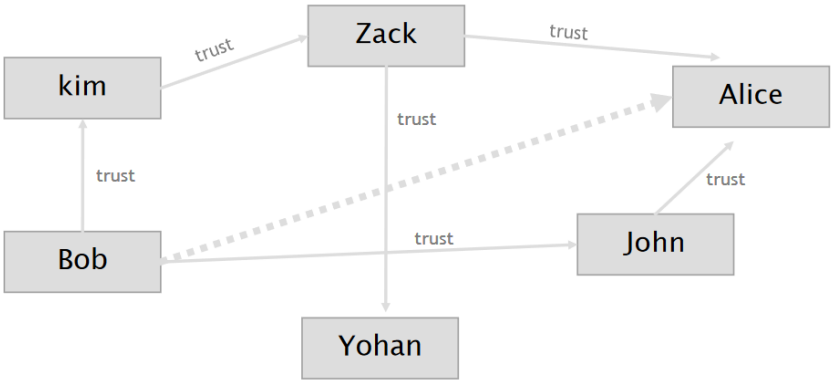
- Adopted in PGP and GnuPG communities (IETF OpenPGP compliant)

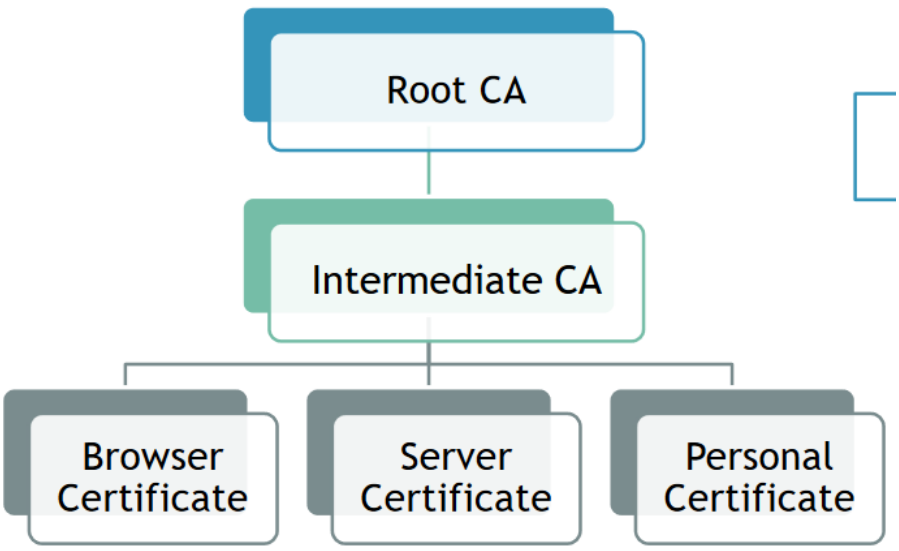
Public Key passed from friend to friend

- PGP/GPG parties at conferences & workshops (key signed parties)

- Key signing parties are common in PGP & GNU community
- PGP **public key infrastructure** (PKI) does not depend on a **central key certifying authority,** but in a **distributed web of trust** approach
- Key signing parties strengthen the web of trust
- Participants are expected to present adequate identity documents

Web of Trust which uses **self-signed certificates** and third-party attestations of those certificates





Outro acrónimo que está relacionado com a CA é **RA (Registration Autority)**

- Funcionam como revendedores de um certificado pelas CA
- Não devem ser confundidos como CA de nível intermédio

Uma **CA de nível intermédio** é apenas a CA que não é a root, apenas assina por baixo o seu certificado

Certificados X.509

O padrão de formato de **certificado X.509** para a **chave pública de infraestruturas (PKI)**

Public Key Infrastructure (PKI) = The set of standards related to the **creation**, **distribution**, **use**, and **revocation** of digital certificates

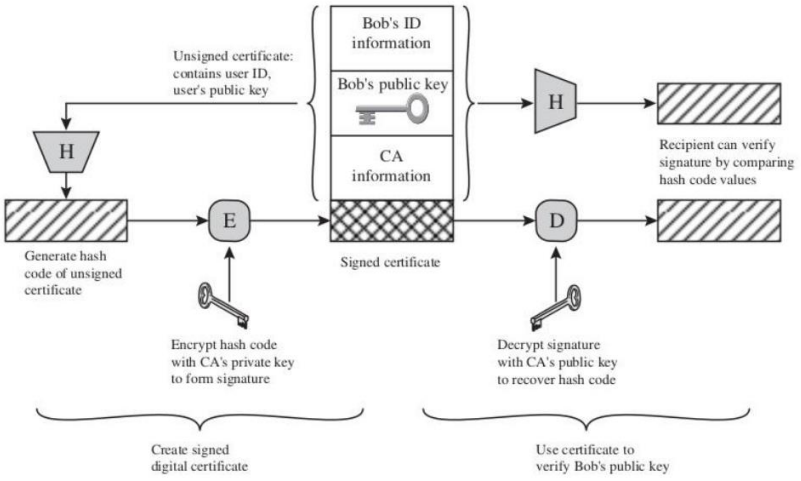
In cryptography, **X.509** is a standard that defines the format of **public key certificates**

X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web

The public keys of the root CAs, of which VeriSign, Comodo, and so on, are examples, are incorporated in your **browser software** and **other applications** that require networking

So that the root-level verification is not subject to network-based man-in-the-middle attacks

- O X.509 define uma framework para a prestação de serviços de autenticação pela diretoria X.500 para os seus users
- A diretoria pode servir como **repositório de certificados de chaves públicas**
- O certificado **contém** a **chave pública do user e também é assinado pela CA confiada**.



Owner's **public** key
Owner's name or alias
Expiration date
Serial number
Organization that issued the certificate and its signature
Etc.

- **X.509 formats (PKCS - *Public-Key Cryptography Standards*)**
  - PKCS#7/P7B format – RSA Labs (**Cryptographic Message Syntax Standard**)
    - Envelope to store multiple PEM or DER certificates
    - Used to sign and/or encrypt messages under a PKI
    - Used also for certificate dissemination
    - All the chain but no private keys
    - Base for S/MIME

- PKCS#12/PFX format – (**Personal Information Exchange Syntax Standard**)
  - Defines a file format used to **store certificates with private keys**
  - **Store private keys** with accompanying X.509 **public key certificates**
  - **Protected with a password-based symmetric key**
  - Also used to **bundle all the members of a chain** of trust
  - PFX is a predecessor to PKCS#12

Quantas mais CA o nosso browser confia, mais comprometemos em segurança. Devido a essas razões muita gente critica o modelo das CA

- Suppose CA issue certificates with errors
  - How to invalidate the certificate?

Revocation

Add certificate **serial number** to revocation lists…

You must stop using revoked certificates

- **Can a certificate issued by a CA be "forged" ?**
  - Yes!
- **Can a certificate issued by a CA be "forget" ?**
  - Yes!

Assinaturas digitais

- **Hash verification**
  - If the **Hash received** is equal to the **computed Hash**
    - Message is authentic (authentication)
    - Message was not changed (integrity)
  - Else
    - Message must not be "processed", **authentication and/or integrity failed**

Autenticação de Mensagens

Autenticação de mensagem ou assinatura digital tem dois níveis de funcionalidade

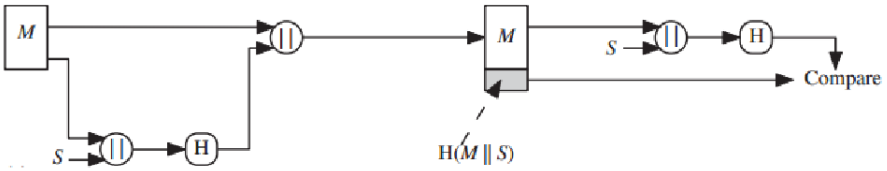
- At the lower level, there must be some sort of function that produces an authenticator:
  - ✓ **a value to be used to authenticate a message**
- This lower-level function is then **used as a primitive in a higher-level authentication protocol** that enables a receiver to verify the authenticity of a message

Tipos de funções que podem ser utilizadas para produzir autenticadores

Hash function	• Map a message of any length into a fixed length hash value, which serves as the authenticator
Message encryption	• The cipher text of the entire message serves as its authenticator
Message authentication code (MAC)	• A function of the message and a secret key that produces a fixed-length value **- the authenticator**

Using Hash function

- Types of functions that may be used to produce an authenticator:

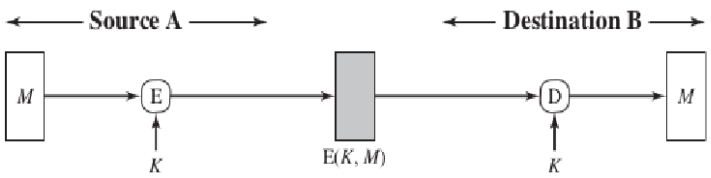


Message authentication, integrity and no confidentiality

- Reduces processing burden
- The two communicating parties share a common secret value that is concatenated to the message

Using Encryption function

- Till now we have said that symmetric encryption basically provides **confidentiality**. Is this true?
- It also provides sender **authentication** (among those who share the secret key)



Symmetric encryption:	confidentiality and authentication

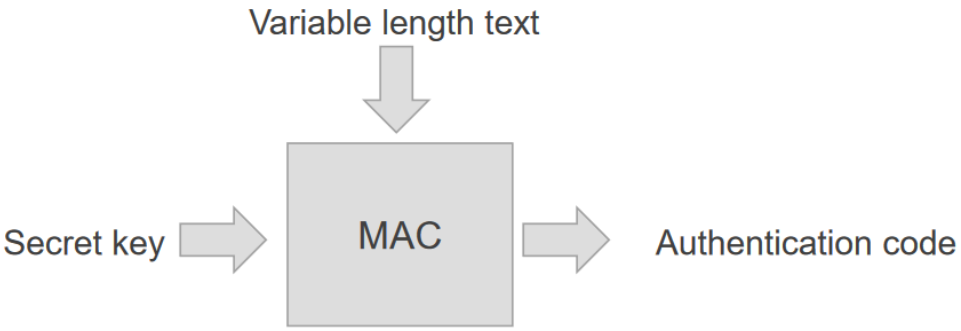
- What about **integrity**?
  Message is not always readable/verifiable by humans (e.g., binary, image files)

Códigos de autenticação de mensagens (MAC)

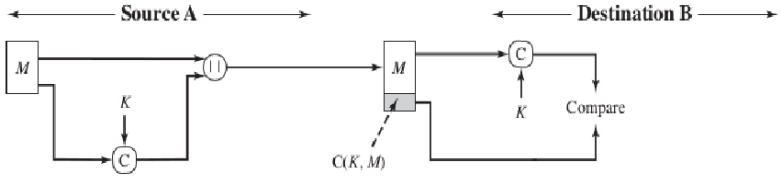
Algoritmo MAC é um algoritmo simétrico de criptografia que providencia autenticação de mensagem

Para utilizar o MAC, o remetente e destinatário partilham a mesma chave simétrica.

Essentially, a MAC is an **encrypted checksum** generated on the underlying message that is **sent along with a message** to ensure message authentication



- How to make a Message Authentication Code? Prime approach:



(a) Message authentication

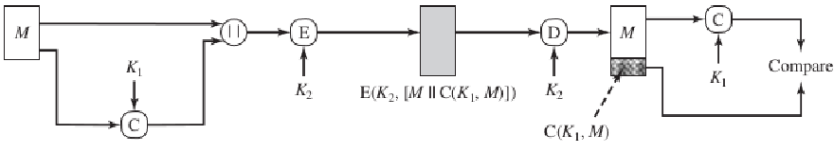
- **Message authentication**, involving the use of a **secret key to generate a small fixed-size block of data** (e.g., using the last block of block ciphers), known as a **cryptographic checksum** or **MAC**, that is appended to the message (assumes that communicating parties, share a secret key)
- A MAC function is **similar to encryption**
- One **difference is that the MAC algorithm need not be reversible**, as it must be for decryption

A principal diferença entre uma hash e o algoritmo MAC é que o MAC usa a chave secreta durante a compressão

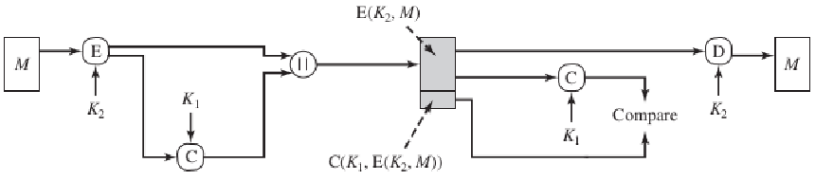
Se quisermos usar confidencialidade, temos de encriptar a mensagem.

- On receipt of the message and the MAC
  - The receiver feeds the received message and the shared secret key *K* into the MAC algorithm and re-computes the MAC value
- The receiver now **checks equality** of freshly computed MAC with the MAC received from the sender:
  - If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender
- If the **computed MAC does not match the MAC sent** by the sender:
  - The receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified
  - As a bottom-line, a receiver safely assumes that the message is not the genuine

- How to make a Message Authentication Code? Prime approach:



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Two separate keys are needed, shared by sender/receiver

As limitações do MAC

Estão apenas associadas a natureza da parte simétrica

Establishment of Shared Secret

- It can provide message authentication among pre-decided legitimate users who have **shared key**
- This requires establishment of shared secret prior to use of MAC

Inability to Provide Non-Repudiation

Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions

- MAC technique **does not provide a non-repudiation service**
  - If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender
- Though no third party can compute the MAC
  - Still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC

Ambas estas limitações, podem ser ultrapassadas utilizando uma **assinatura digital**

(Confiança Mútua) - Gerenciamento e Distribuição de Chaves

Cryptographic key **management**

Cryptographic key **distribution**

⇨ Are complex!

Involving **cryptographic**, **protocol**, and **management** considerations

O ponto forte que qualquer sistema de criptografia, está na técnica de distribuição das chaves.

Symmetric encryption

Big challenge
- Secret key sharing!
- Protected from access by others

Asymmetric encryption

Encrypt with target public key...

Certificates (man-in-the-middle)

- **Using symmetric encryption**
  - Dilemma
    - Symmetric key must be shared among players
    - **Frequent key changes** are desirable, to limit the amount of data compromised if an attacker learns the key

Modos de distribuição de chaves:

#1	A selects a key and delivers it physically to B
#2	A third party delivers it physically to A and B
#3	If A and B previously used a secret key, one transmits the new key to the other encrypted with old key
#4	If A and B have an encrypted connection to third party C, C delivers keys through the encrypted link (tunnel)

- For **end-to-end encryption** over a network, manual delivery is awkward
- In a distributed system, **any given host or terminal** may need to engage in exchanges with many other hosts and terminals over time
- Thus, each device needs a number of **keys supplied dynamically**
- The problem is especially difficult in a **wide-area distributed system**

- Option #4

  If A and *B* have an encrypted connection to third party *C*, *C* delivers keys through the encrypted link (tunnel)

  ↓

- For end-to-end encryption, some variations on option #4 has been widely adopted
- A **key distribution center** is responsible for distributing keys to pairs of users (hosts, processes, applications)
- Each user must share a **unique key** with **the key distribution center**

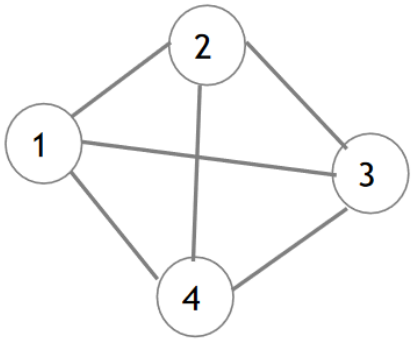
- The concept of a **key hierarchy** and the use of **automated key distribution** techniques

  ↓

  Greatly **reduce the number of keys** that must be **manually managed and distributed**

- It also may be desirable to **impose some control** on the way in which **automatically distributed** keys are used

- In addition to **separating master keys** from **session keys**
  - Defining **different types of session keys** based on usage might be good idea:
    - Data-encrypting key, for communication across a network
    - PIN-encrypting key, for personal identification
    - File-encrypting key, encrypting files in publicly accessible locations
- Communication between end systems is encrypted with **temporary key** (**session key**) for the duration of the connection & then discarded
- **Session key** is obtained from the KDC over the same networking facilities used for end-user communication
- Session keys are transmitted in encrypted form, using a **master key** that is **shared** by the KDC and an **end system or user**
- **Each end system/user** has **a unique master key shared with KDC**, distributed in some fashion (scale of the problem is vastly reduced)



`4(3)/2 = 6 keys`

`3 keys per node`

- **Key distribution centers (KDC):**
  - Responsible for **distributing keys** to pairs of users (hosts, processes, applications) as needed
  - Each user **shares a unique key** (master key) with KDC for key distribution (tunnel)
  - KDC delivers **session keys** through tunnel

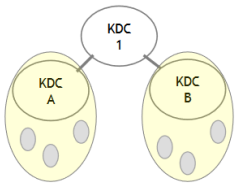
Only N (master) keys are now necessary (one per node)

KDC (Centros de Distribuição de Chaves)

A wishes to establish a logical connection with *B* and requires a **one-time session key** to protect the data transmitted over the connection

Hierarchical key distribution centers

- For big networks, one KDC per domain, building, etc.
  - Minimizes effort in **master key** (MK) **distribution** - most MK are those shared by a local KDC with its local entities
- If *A* (domain 1) needs to talk to *B* (domain 2)
  - Each local KDC interacts with next level KDC



Advantages:
- Easier to share master keys, only local KDCs need to know its population master keys
- Limits damage, limits the damage of a faulty or subverted KDC to its local area only

Session key lifetime

Frequently exchanged = more **secure**

The opponent has less cipher text to work with for any given session key, but more **overhead** (distribution of session keys delays)

Session key lifetime

- Example: **Connection-oriented session**
  - The **same while connection is opened** (or **change every time** the protocol data unit **sequence number cycles**)
- Example: **Connectionless (transaction-oriented) protocol**
  - No explicit connection initiation or termination
  - New session **key for each exchange**, however this **contradicts the main benefit of connectionless protocols** (minimum overhead, no session setup)
  - Perhaps a **session key for** a certain fixed **period/number of transactions**
- A **decentralized key distribution scheme**
  - **Centralized approach requires** KDC to be **trustable**, **secure**, **protected** from subversion (not required for decentralized approaches)
  - In decentralized approaches **end node** does the job
    - But **full decentralization is not practical for larger networks using symmetric encryption** only (may be useful in local contexts)
  - A decentralized approach requires that **each end system be able to communicate in a secure manner with all potential partner end systems** for purposes of session key distribution
  - Each node "knows" the others: $n-1$ master keys per node!
- **Using asymmetric encryption**
  - Simple secret key distribution

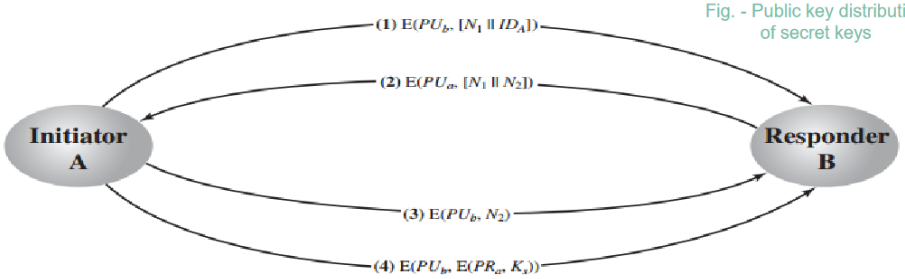
Because of the **inefficiency of public key cryptosystems**
- They are **almost never used for the direct encryption** of sizable block of data
- but **are limited to relatively small blocks**

One of the most important uses of a public-key cryptosystem is to **encrypt secret keys for distribution**

- **Using asymmetric encryption**
  - Secret key distribution with **secrecy** and **authentication**
  - ***A*** has the ***B*'** **certificate** and vice-versa

A and *B* have exchanged public keys by one of the schemes described subsequently

Fig. - Public key distribution of secret keys

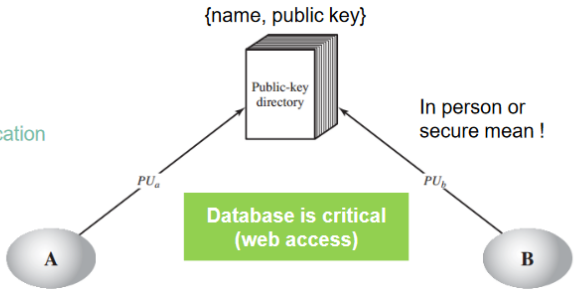


Provides protection against both active and passive attacks

Distribuição de chaves públicas

- **Public available directory**
  - Maintain a **dynamic public directory** of public keys
    - Responsibility of some trusted entity
    - Safer than the previous scenario, but single point of failure (all keys may be compromised) !

Fig. - Public key publication



- **Public-key authority**
  - Now, **public-key is signed by the trusted authority**
  - **Single point of failure** (all keys may be compromised and performance bottleneck) !

Request with timestamp T1

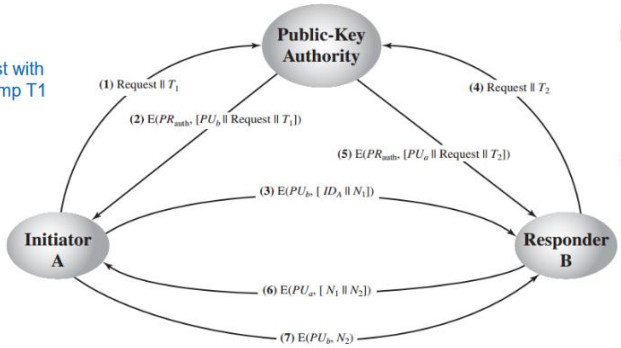
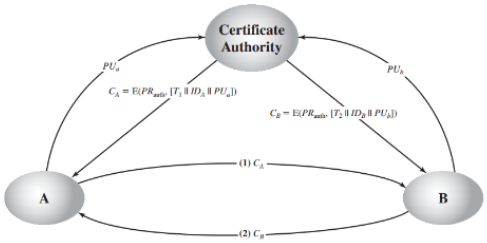


Fig. - Public key distribution

Caching avoids the first 4 steps. Refresh from time to time

[Scheme to be analyzed in the next slide]

Public-key certificates

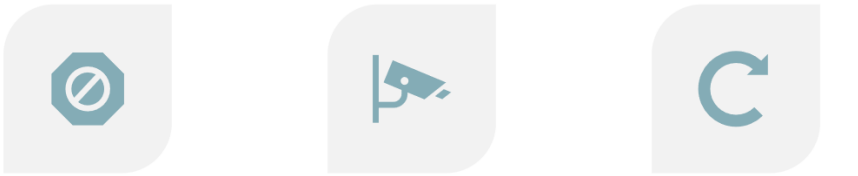


- **Requirements on this scheme:**
  1. Any participant can **read a certificate** to determine the **name and public key** of **the certificate's owner**
  2. Any participant can **verify** that the **certificate originated** from the certificate authority and is not counterfeit
  3. Only the **certificate authority can create** and update certificates
  4. Any participant can verify the currency of the certificate

Disadvantages

- Nodes must belong to the KDC domain
- Central authority!
  - A single breach, it compromises the entire system

Cyber segurança



Prevenir

- Medidas de segurança devem ser tomadas para proteger a informação
- Durante a fase de prevenção
  - Políticas de segurança
  - Controlos (físicos, técnicos, legais, etc.)
  - Processos devem de ser desenhados e implementados
- Proteger de:
  - Desastres naturais, fogos, terramotos…
  - Acidentes
  - Erros
  - Mau funcionamento
  - Ataques

Deteção

- A deteção de um sistema que está comprometido é extremamente crítico
- Uma defesa em layers deve de ser implementada
    - Para sempre que um layer seja comprometido, avisa logo o a seguir e faz suar um alarme de aviso
- O elemento mais importante desta estratégia é deteção a tempo e notificação do compromisso

Concept	Description	Example
Threat	Any activity that may endanger the information	Attackers routinely perform surveys (scanning) in systems accessible via the Internet, the discovery of open ports
Vulnerability	A security hole that can be exploited by an attacker	Misconfigured firewall with a TCP/UDP port open
Attack	Deliberate attempt to circumvent security systems	Forward beyond the firewall and access the internal network

Hacker → Geek que explora a vulnerabilidades do sistema, White Hacker

Cracker → Usa as vulnerabilidades do sistema como sua vantagem para fazer dinheiro, Black Hacker

A maioria dos hackers são, Gray Hackers

Motivos dos ataques

- De dentro da empresa
- Intencional
- Acidental
- Erros

▶ **Most common threats**

Targets	Examples	Threats
Networks	Internet, intranet, extranets	• Intrusion • Denial of Service (DoS)
Data	Credit cards data, Business confidential information	• Theft • Destruction • Interception
Physical components	Computers, switches, routers	• Theft • Destruction • Misconfiguration

Origin	Examples
Users	• Share or use weak passwords • Ignoring or misunderstanding of security policies • Open e-mail, visit Web sites or use software with malicious code • Be nudged to break security policies
Services	• Services misconfiguration, do not update software security patch's • Inadequate protection of network access accounts • Inadequate protection of physical access to the hardware • Bypass security policies
Software	• Use of operating system and applications with know vulnerabilities that allow exploitation by potential attackers

Attacks classification

- Depending on the **changes operated in the data**
  - Passive
  - Active
- Depending on how the **attack is operated**
  - Listening
  - Block
  - Deviation
  - Modification

Passive attacks

Passive attacks do not involve changing the information

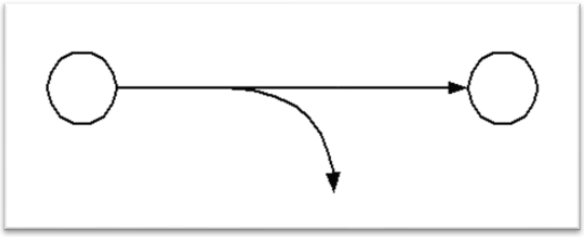
- Listening

Active attacks

Active attacks always involve at least one of the following:

- Insertion, removal or alteration of data
- Deviation or fabrication (Replay)
- Blocking (Denial of Service)
- Modification (Man-in-the-Middle)

- Passive attacks/Listening/Snooping
  - Require interfaces in promiscuous mode
    - Best known attack - Sniffing
    - Defense is possible - Encryption (and regular control/auditing)



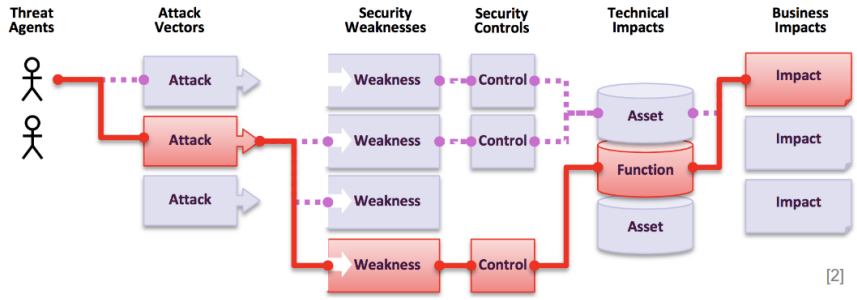
Ataques ativos

- Ataques na HW/SW (cortar os cabos de rede ou energia, tudo o que for ativo e no momento)
- Bombardear o email
- SYN Flood, para fazer o request denial
- Spoofing, do endereço IP
- Vírus
- Software malicioso
- DoS

Vulnerabilidade

Application Security Risks

- Attackers can potentially use **many different paths** through an application to do **harm a business** or **organization**
- Sometimes, these paths are **trivial to find** and **exploit** (and sometimes they are extremely difficult)



- Ataques de vetores, dão a oportunidade de hackers encontrarem vulnerabilidades no sistema

SQL Injection

- Cracker pode apagar todos os dados dos utilizadores ou mesmo, ter acesso a todos os dados dos utilizadores

Cross-site scripting (XSS) ataque

- Conteúdo que um website pode correr HTML ou JS sem o utilizador saber

An XSS attack rewrites the structure of a Web page or executes arbitrary JavaScript within the victim's Web browser

- The **visitor uses characters reserved for HTML markup as part of the search query**
- If the **Web site echoes** anything typed in a search box
- For example, this **reflected content can be used to rewrite & convey a very different message to the Web browser** than the Web site's developers intended

- O cracker consegue inserir JS dentro de uma textbox, que depois vai afetar o cliente

Cross-Site Request Forgery (CRSF) attack

In simplest terms, a **CSRF attack forces the victim's browser to make a request without the victim's knowledge** or agency

```
<html><body>This is an empty page!
     <iframe src=http://search.yahoo.com/search?p=maltese+falcon
          height=0 width=0 style=visibility:hidden>
     <img src=http://search.yahoo.com/search?p=maltese+falcon alt="">
</body></html>
```

- **When anyone visits this page, his/her Web browser will make two search requests**
  - What if the link is to an advertising banner?
  - Attacker would get money from **click fraud** from a variety of browsers, IP addresses, geographic locations **difficult for fraud detection**

Cybersecurity – Solutions?

- Use **encryption and authentication** in the various OSI layers
- Use **secure settings**
- **Reuse, don't reimplement** (like in cryptographic algorithms)
  - Frameworks code reuse is better than writing from scratch (of course, reusing insecure code is no better than writing insecure code from scratch)
- **Apply patches from manufacturers** when available
- **Be attentive** to the disclosure of vulnerabilities and attacks
- Use **monitoring** and **auditing** tools
- **Adopt cybersecurity standards** and **official sources of data**