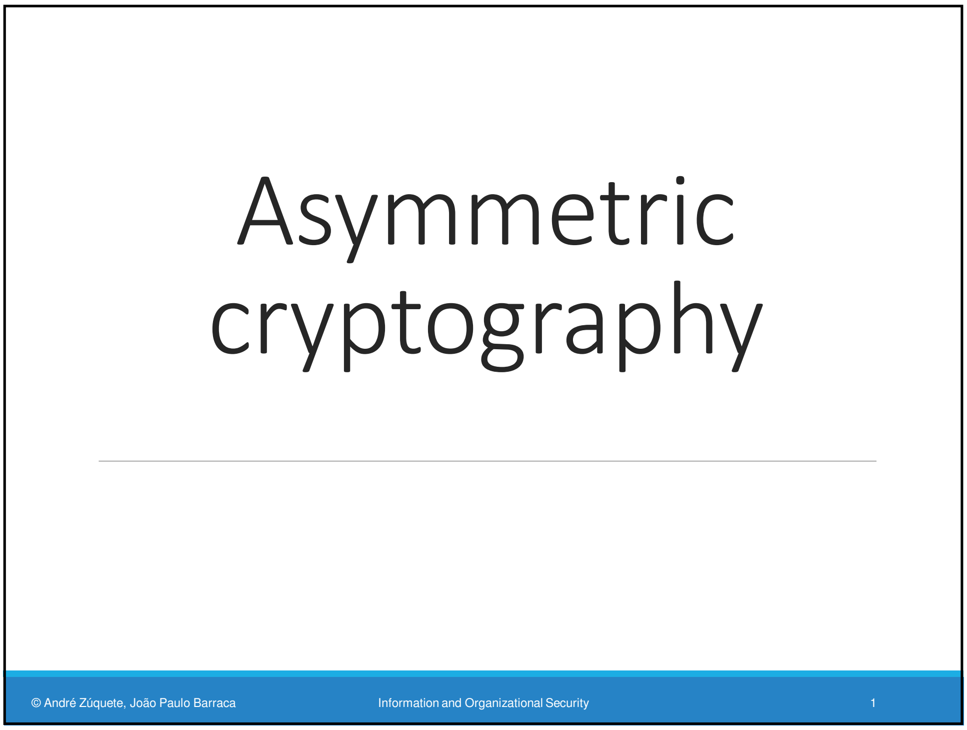
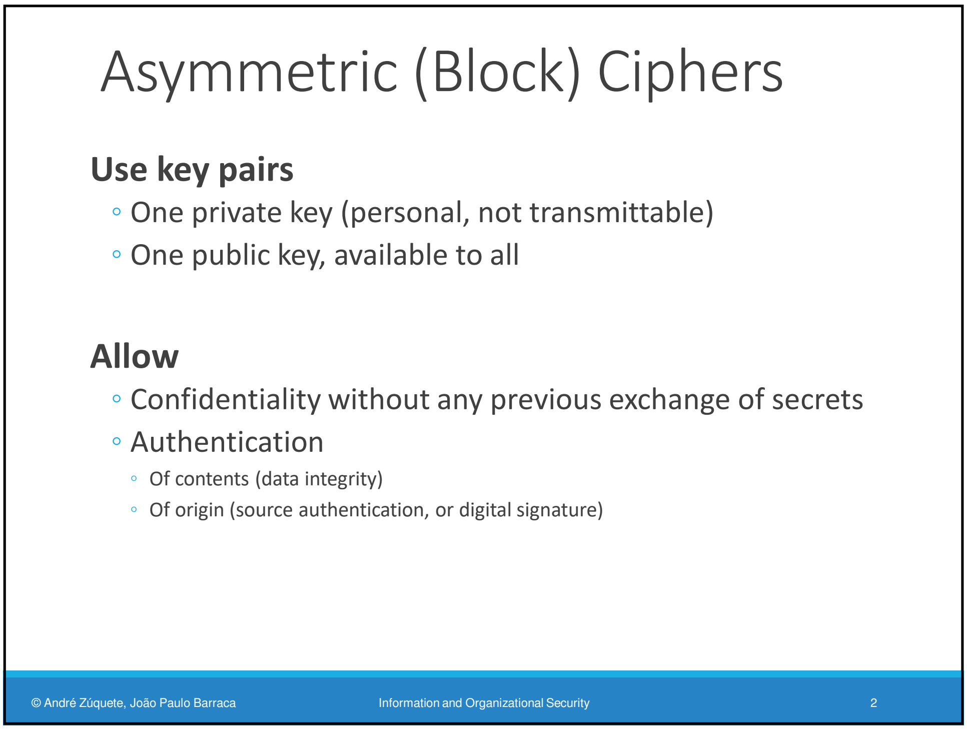


# Asymmetric cryptography



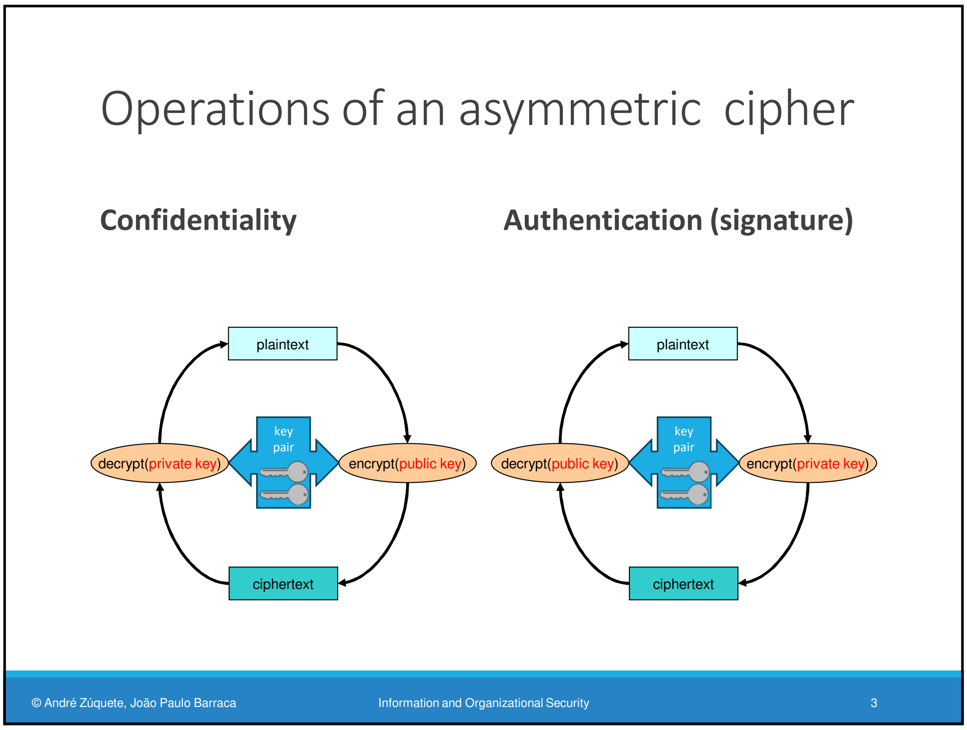
## Asymmetric (Block) Ciphers

**Use key pairs**

- One private key (personal, not transmittable)
- One public key, available to all

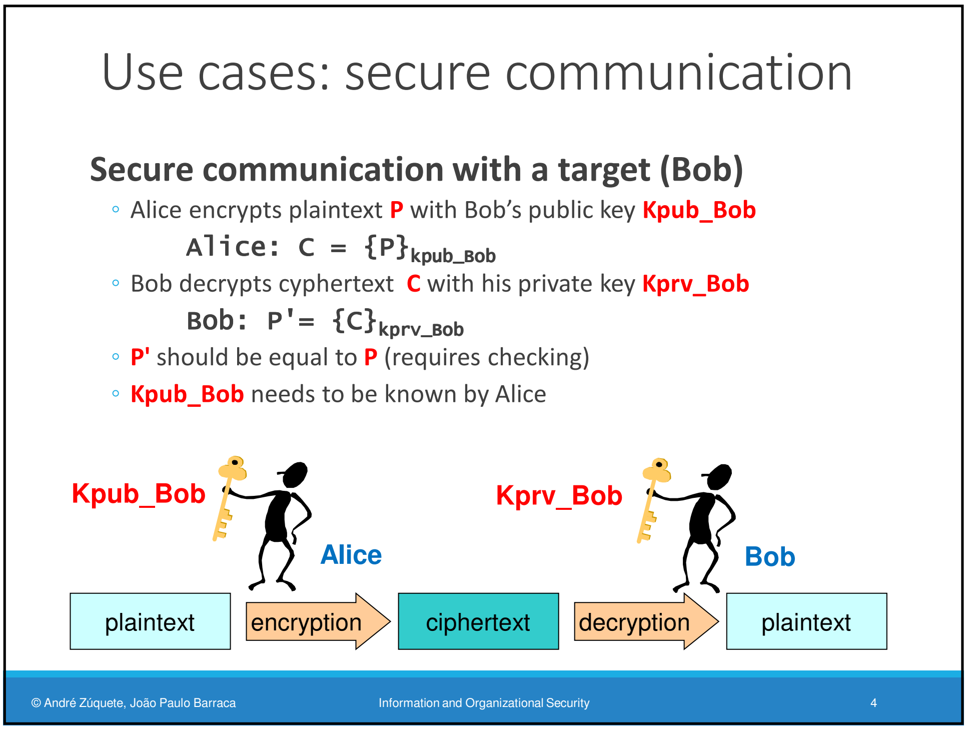
**Allow**

- Confidentiality without any previous exchange of secrets
- Authentication
  - Of contents (data integrity)
  - Of origin (source authentication, or digital signature)



# Operations of an asymmetric cipher

**Confidentiality** **Authentication (signature)**



# Use cases: secure communication

## Secure communication with a target (Bob)

- Alice encrypts plaintext **P** with Bob's public key **Kpub_Bob**

  `Alice: C = {P}`$_{\text{kpub\_Bob}}$

- Bob decrypts cyphertext **C** with his private key **Kprv_Bob**

  `Bob: P'= {C}`$_{\text{kprv\_Bob}}$

- **P'** should be equal to **P** (requires checking)
- **Kpub_Bob** needs to be known by Alice

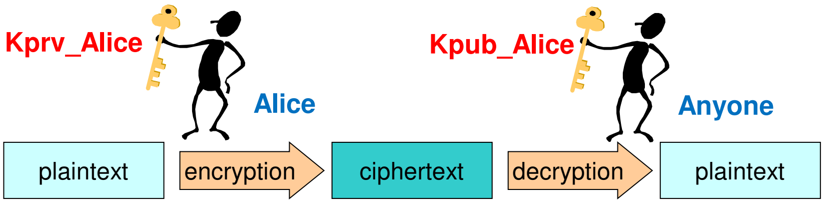
# Use cases: signature

## Data signature by Alice

- Alice encrypts plaintext **P** with her private key **Kprv_Alice**

  Alice: C = $\{P\}_{kprv\_Alice}$
- Anyone can decrypt cyphertext **C** with Alice's public key **Kpub_Alice**

  Anyone: P'= $\{C\}_{kpub\_Bob}$
- If **P'** = **P**, then **C** is Alice's signature of **P**
- **Kpub_Alice** needs to be known by signature verifiers



# Asymmetric ciphers

## Advantages

- They are a fundamental authentication mechanism
- They allow to explore features that are not possible with asymmetric ciphers

## Disadvantages

- Performance
- Usually are very inefficient and memory consuming

## Problems

- Trustworthy distribution of public keys
- Lifetime of key pairs

# Asymmetric ciphers

## Approaches: complex mathematic problems

- Discrete logarithms of large numbers
- Integer factorization of large numbers

## Most common algorithms

- RSA
- ElGamal
- Elliptic curves (ECC)

## Other techniques with asymmetric key pairs

- Diffie-Hellman (key agreement)

# RSA (Rivest, Shamir, Adelman, 1978)

## Keys

- Private: (d, n)
- Public: (e, n)

## Public key encryption (confidentiality)

- $C = P^e \bmod n$
- $P = C^d \bmod n$

P, C are numbers

$0 \le P, C < n$

## Private key encryption (signature)

- $C = P^d \bmod n$
- $P = C^e \bmod n$

# RSA (Rivest, Shamir, Adelman, 1978)

## Computational complexity

- Discrete logarithm
- Integer factoring

coprime → $\gcd(a, b) = 1$
× → multiplication
mod → modulo operation
≡ → modular congruence

## Key selection

- Large $n$ (hundreds or thousands of bits)
- $n = p \times q$ with $p$ and $q$ being large (secret) prime numbers
- Chose an $e$ co-prime with $(p-1) \times (q-1)$
- Compute $d$ such that $e \times d \equiv 1 \pmod{(p-1) \times (q-1)}$
- Discard $p$ and $q$
- The value of $d$ cannot be computed out of $e$ and $n$
  - Only from $p$ and $q$

# RSA example

## $p = 5$ $q = 11$ (prime numbers)

- $n = p \times q = 55$
- $(p-1) \times (q-1) = 40$

## $e = 3$ (public key = e, n)

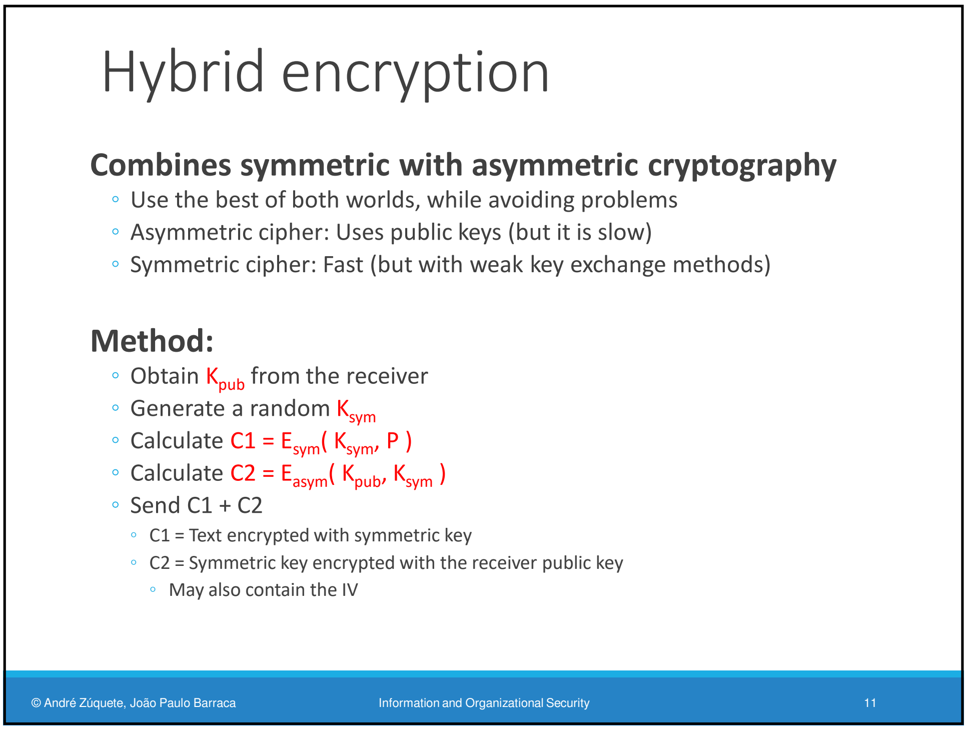
- Coprime of 40

## $d = 27$ (private key = d, n)

- $e \times d \equiv 1 \pmod{40}$ → $d \times e \bmod 40 = 1$, $(27 \times 3) \bmod 40 = 1$

## For $P = 26$ (notice that $P, C \in [0, n-1]$)

- $C = P^e \bmod n = 26^3 \bmod 55 = 31$
- $P = C^d \bmod n = 31^{27} \bmod 55 = 26$



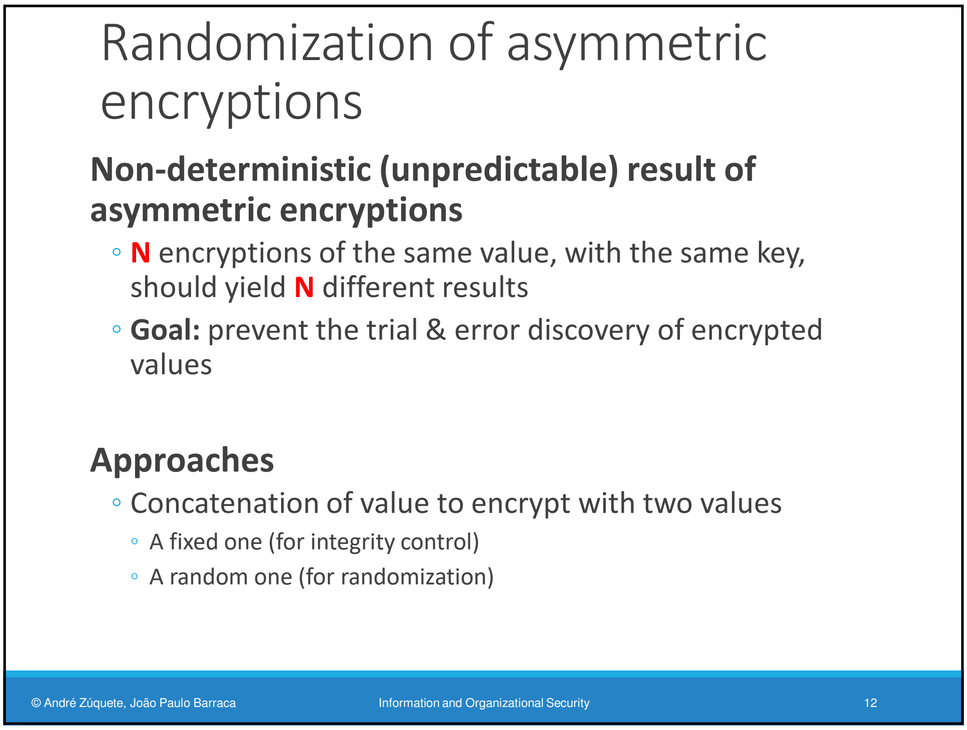
# Hybrid encryption

## Combines symmetric with asymmetric cryptography

- Use the best of both worlds, while avoiding problems
- Asymmetric cipher: Uses public keys (but it is slow)
- Symmetric cipher: Fast (but with weak key exchange methods)

## Method:

- Obtain $K_{pub}$ from the receiver
- Generate a random $K_{sym}$
- Calculate $C1 = E_{sym}( K_{sym}, P )$
- Calculate $C2 = E_{asym}( K_{pub}, K_{sym} )$
- Send C1 + C2
  - C1 = Text encrypted with symmetric key
  - C2 = Symmetric key encrypted with the receiver public key
    - May also contain the IV



# Randomization of asymmetric encryptions

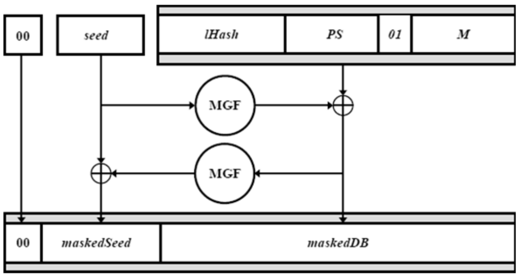
## Non-deterministic (unpredictable) result of asymmetric encryptions

- **N** encryptions of the same value, with the same key, should yield **N** different results
- **Goal:** prevent the trial & error discovery of encrypted values

## Approaches

- Concatenation of value to encrypt with two values
  - A fixed one (for integrity control)
  - A random one (for randomization)

## Randomization of asymmetric encryptions: OAEP (Optimal Asymmetric Encryption Padding)



**lHash:** digest over **Label**

**seed:** random

**PS:** zeros

**M:** plaintext

**MGF: Mask Generation Function**

- Similar to Hash, but with variable size

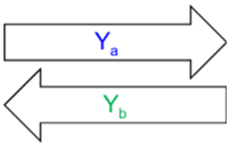
## Diffie-Hellman Key Agreement (1976)

q (large prime)
α (primitive root mod q)

**a = random**

$Y_a = \alpha^a \bmod q$

$K_{ab} = Y_b^a \bmod q$



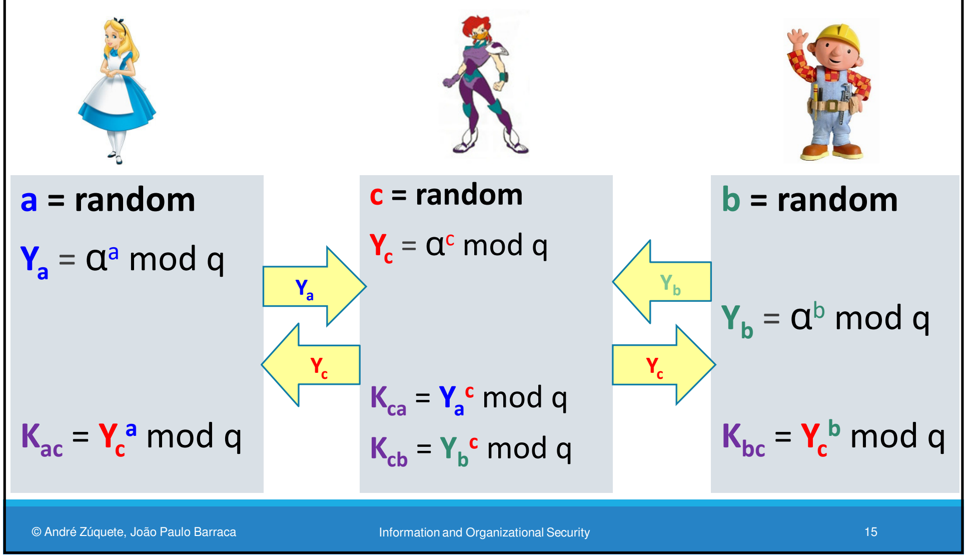
**b = random**

$Y_b = \alpha^b \bmod q$

$K_{ba} = Y_a^b \bmod q$

$K_{ab} = K_{ba}$

# DH Key Agreement: MitM attack



$a$ = random

$Y_a = \alpha^a \bmod q$

$K_{ac} = Y_c^{\ a} \bmod q$

$c$ = random

$Y_c = \alpha^c \bmod q$

$K_{ca} = Y_a^{\ c} \bmod q$

$K_{cb} = Y_b^{\ c} \bmod q$

$b$ = random

$Y_b = \alpha^b \bmod q$

$K_{bc} = Y_c^{\ b} \bmod q$

# Elliptic Curve Cryptography (ECC)

## Elliptic curves are specific functions

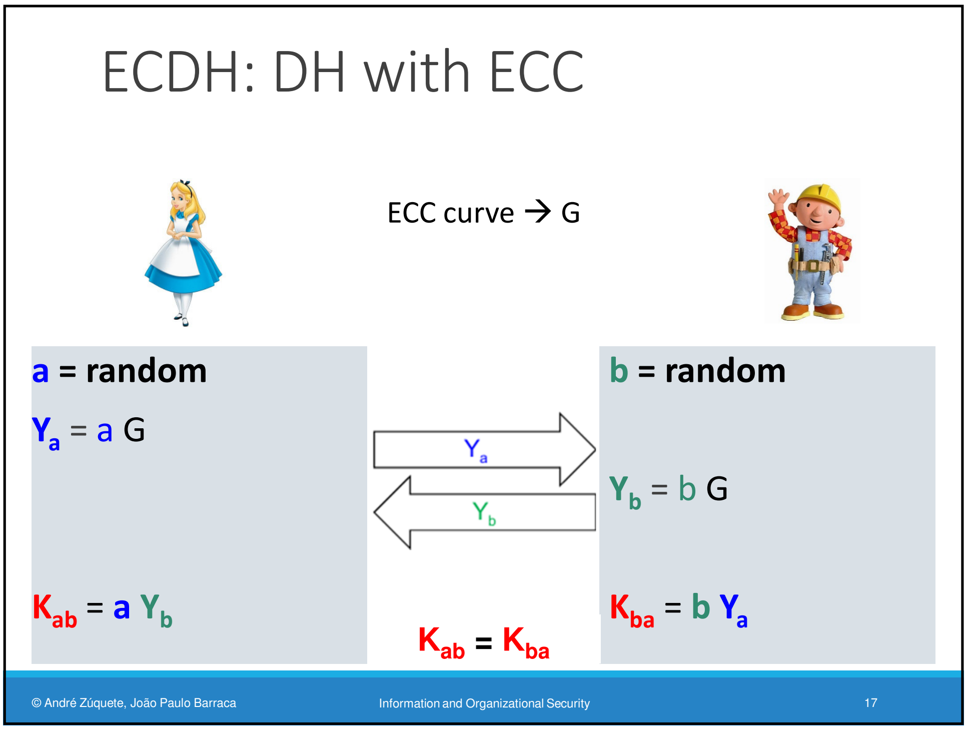
- They have a generator ($G$)
- A private key $K_{prv}$ is an integer with a maximum of bits allowed by the curve
- A public key $K_{pub}$ is a point $(x,y) = K_{prv} \times G$
- Given $K_{pub}$, it should be hard to guess $K_{prv}$

## Curves

- NIST curves (15)
  - P-192, P-224, P-256, P-384, P-521
  - B-163, B-233, B-283, B-409, B-571
  - K-163, K-233, K-283, K-409, K-571

## Other curves

- Curve25519 (256 bits)
- Curve448 (448 bits)



# ECDH: DH with ECC

ECC curve → G

$a$ = random

$Y_a = a\,G$

$Y_a$

$Y_b$

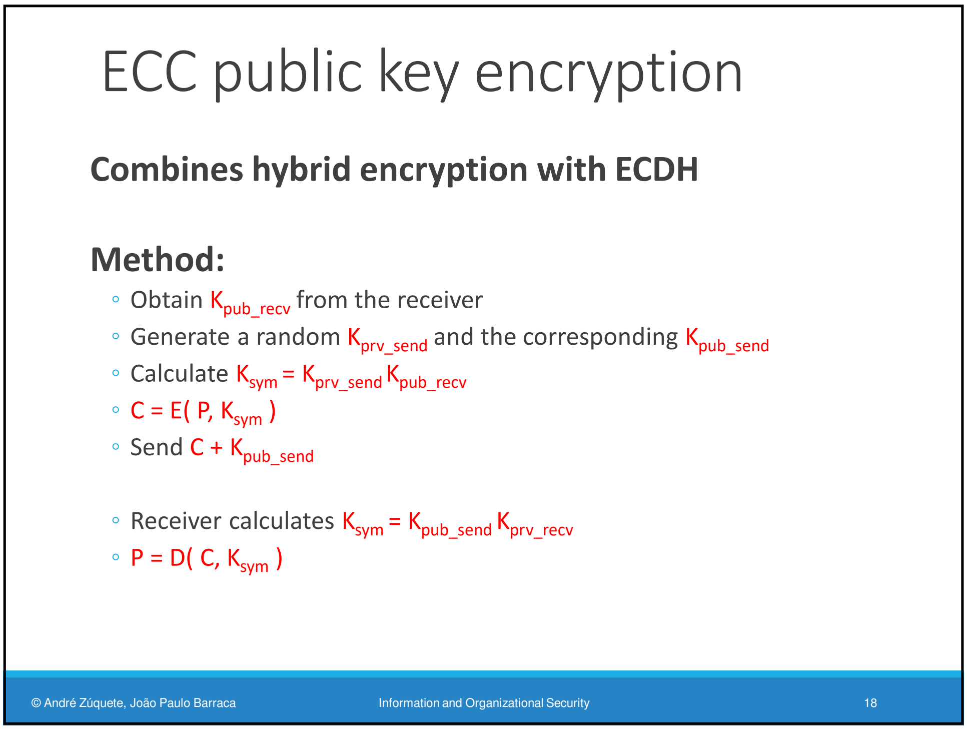
$b$ = random

$Y_b = b\,G$

$K_{ab} = a\,Y_b$

$K_{ba} = b\,Y_a$

$K_{ab} = K_{ba}$



# ECC public key encryption

**Combines hybrid encryption with ECDH**

**Method:**

- Obtain $K_{pub\_recv}$ from the receiver
- Generate a random $K_{prv\_send}$ and the corresponding $K_{pub\_send}$
- Calculate $K_{sym} = K_{prv\_send}\,K_{pub\_recv}$
- $C = E(\,P, K_{sym}\,)$
- Send $C + K_{pub\_send}$

- Receiver calculates $K_{sym} = K_{pub\_send}\,K_{prv\_recv}$
- $P = D(\,C, K_{sym}\,)$

# Digital signatures

# Asymmetric (Block) Ciphers

## Use key pairs

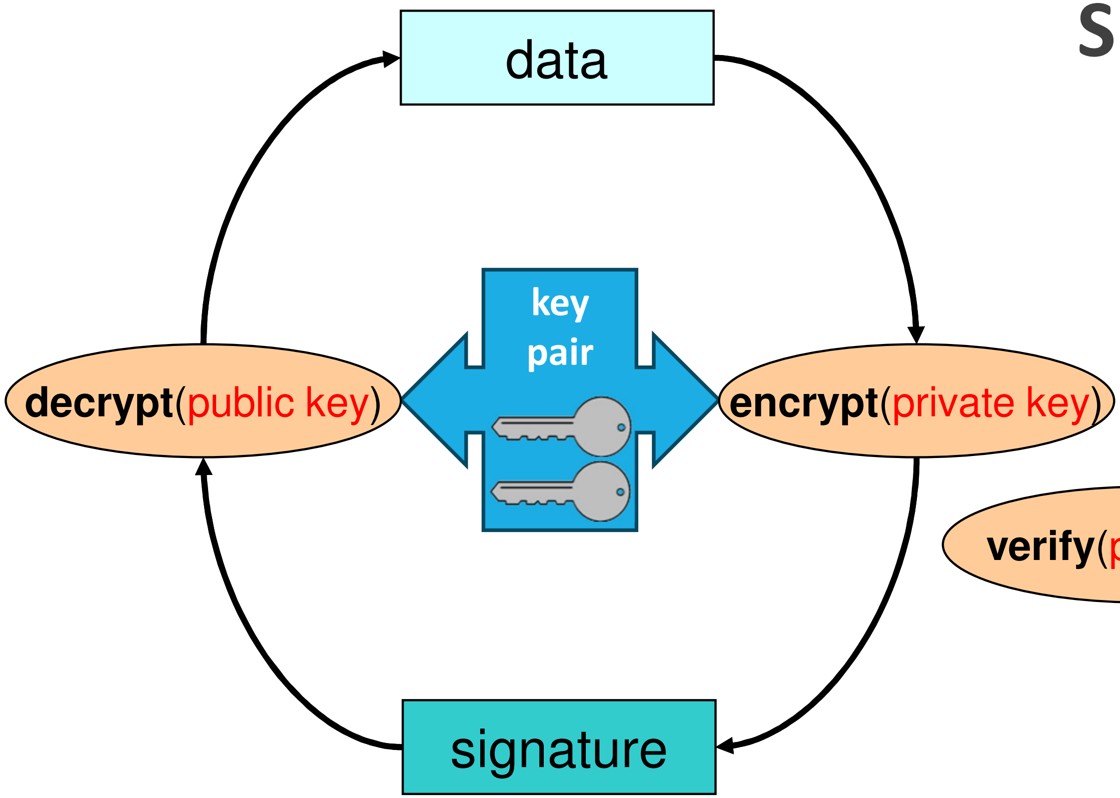
- One private key (personal, not transmittable)
- One public key, available to all

## Allow

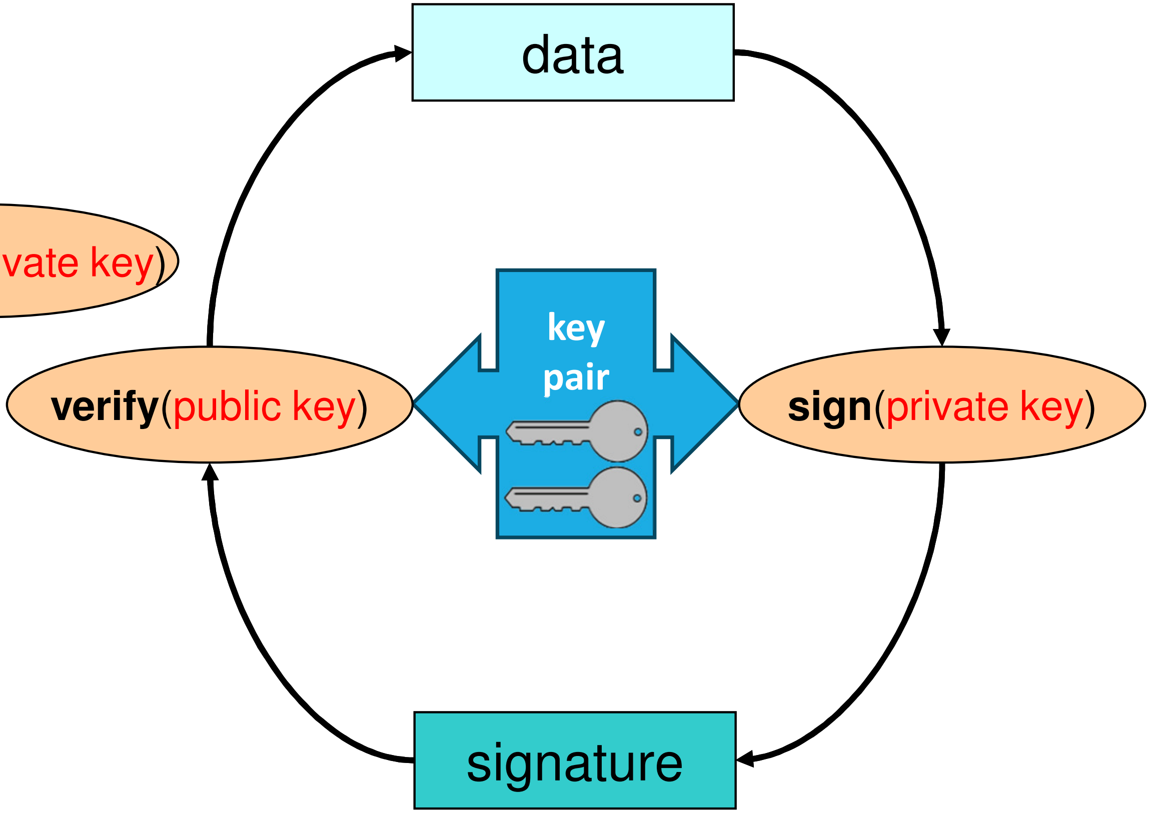
- Confidentiality without any previous exchange of secrets
- Authentication
  - Of contents (data integrity)
  - Of origin (source authentication, or digital signature)

# Digital signatures

## Encrypt / decrypt (RSA)



## Sign / verify (ElGamal, EC)



# Digital signatures

**Authenticate the contents of a document**

- Ensure its integrity (it was not changed)

**Authenticate its author**

- Ensure the identity of the creator/originator

**Prevent repudiation of signatures**

- Non-repudiation
- Genuine authors cannot deny authorship
  - Only the identified author could have generated a given signature

# Digital Signatures

**Approaches**

- Asymmetric encryption/decryption or signature/verification
- Digest functions (only for performance)

**Signing:** $A_x(doc) = info + E(K_x^{-1}, digest(doc + info))$

$A_x(doc) = info + S(K_x^{-1}, digest(doc + info))$

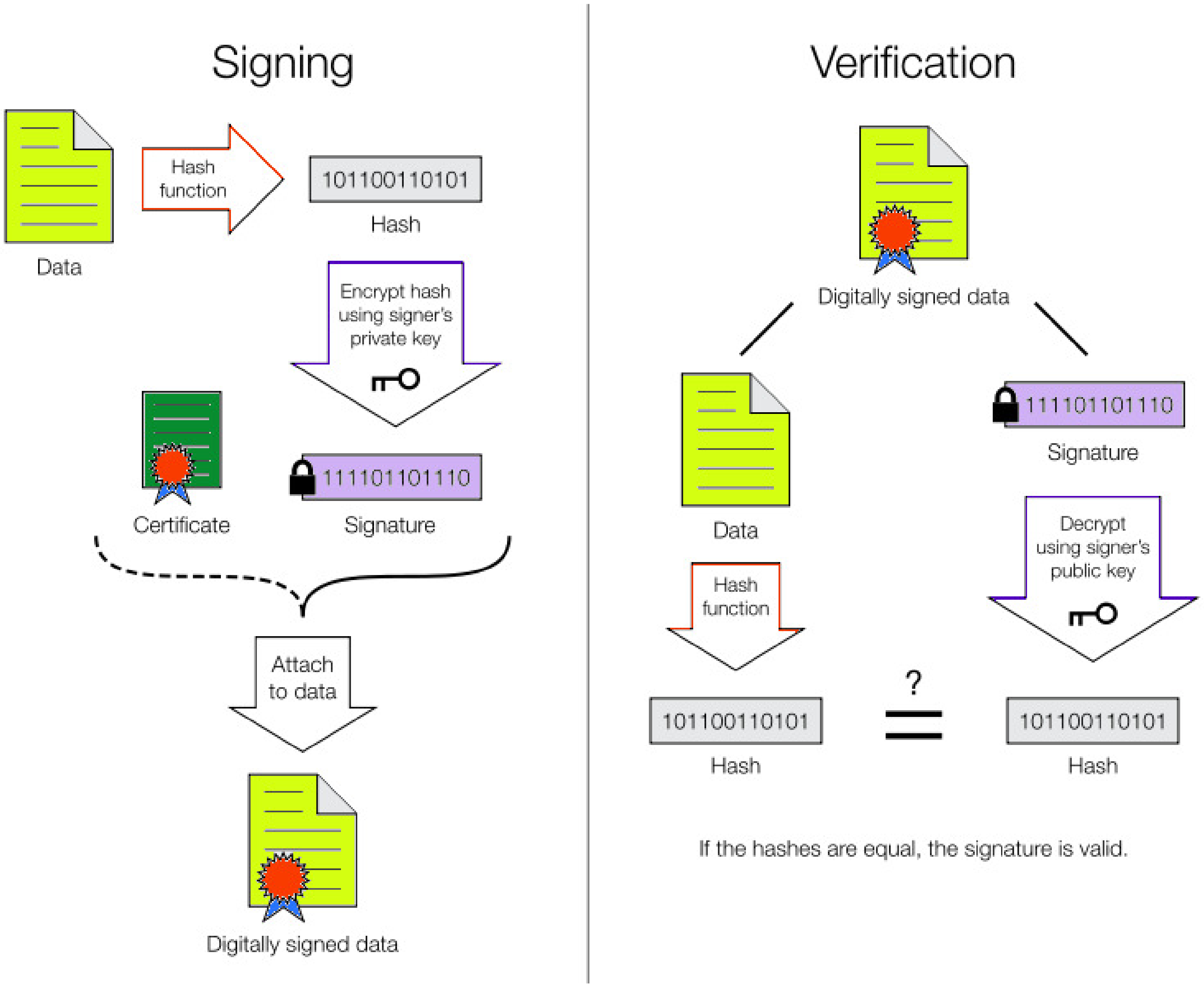
$info$ = signing context, signer identity, $K_x$

**Verification:**

$D(K_x, A_x(doc)) \equiv digest(doc + info)$

$V(K_x, A_x(doc), doc, info) \rightarrow$ True / False wikipedia, http://en.wikipedia.org/wiki/Digital_signature

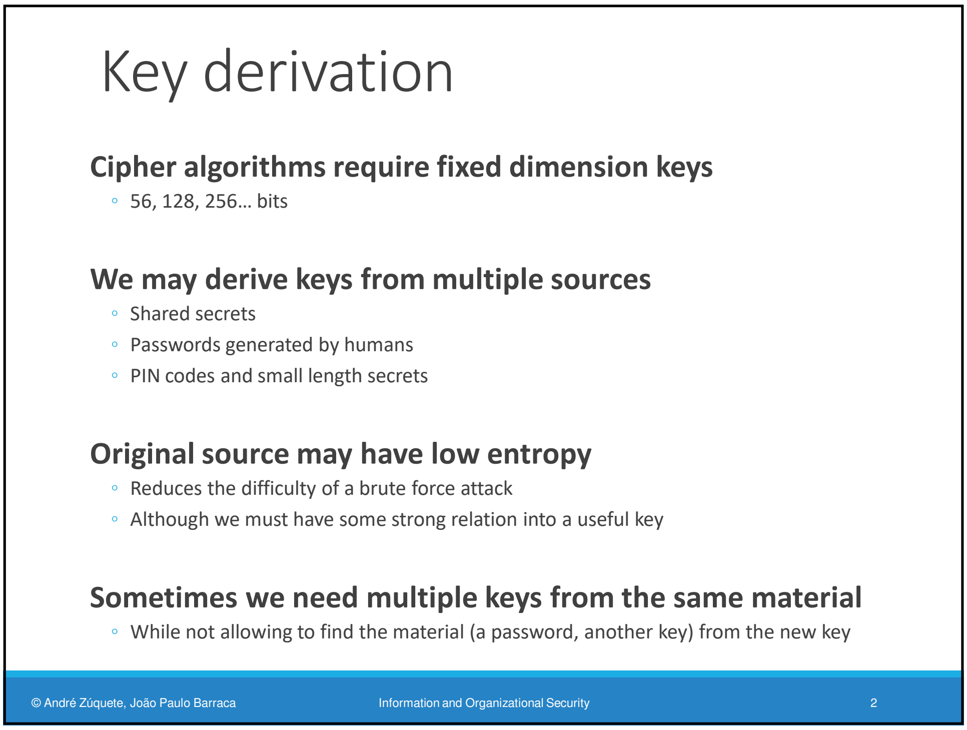
# Encryption / decryption signatures



# Digital signature on a mail: Multipart content, signature w/ certificate

```
From - Fri Oct 02 15:37:14 2009
[…]
Date: Fri, 02 Oct 2009 15:35:55 +0100
From: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Reply-To: andre.zuquete@ua.pt
Organization: IEETA / UA
MIME-Version: 1.0
To: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Subject: Teste
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="------------ms050405070101010502050101"

This is a cryptographically signed message in MIME format.

--------------ms050405070101010502050101
Content-Type: multipart/mixed;
 boundary="------------060802050708070409030504"

This is a multi-part message in MIME format.
--------------060802050708070409030504
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Corpo do mail

--------------060802050708070409030504—
--------------ms050405070101010502050101
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIamTCC
BUkwggSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTELMAkGA1UEBhMCVVMxGDAWBgNV
[…]
KoZIhvcNAQEBBQAEgYCofks852BV77NVuww53vSxO1XtI2JhC1CDlu+tcTPoMD1wq5dc5v40
Tgsaw0N8dqgVLk8aC/CdGMbRBu+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbqvpd2QO2
vxB3iPTlU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAAAAA==
--------------ms050405070101010502050101--
```

# Key derivation



## Key derivation

**Cipher algorithms require fixed dimension keys**

- 56, 128, 256… bits

**We may derive keys from multiple sources**

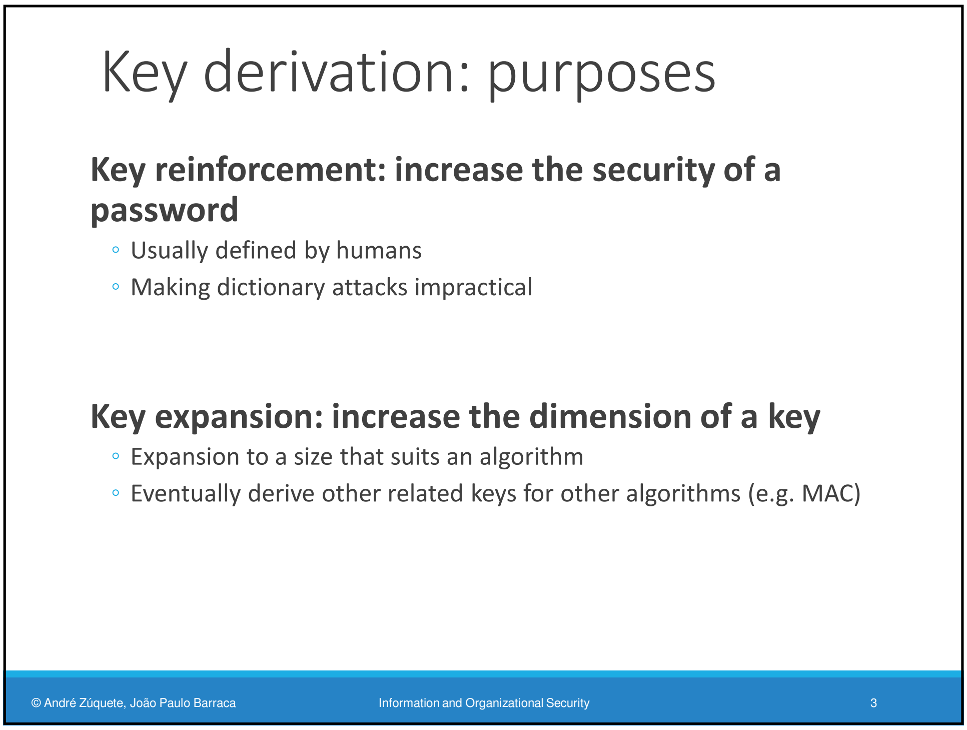
- Shared secrets
- Passwords generated by humans
- PIN codes and small length secrets

**Original source may have low entropy**

- Reduces the difficulty of a brute force attack
- Although we must have some strong relation into a useful key

**Sometimes we need multiple keys from the same material**

- While not allowing to find the material (a password, another key) from the new key



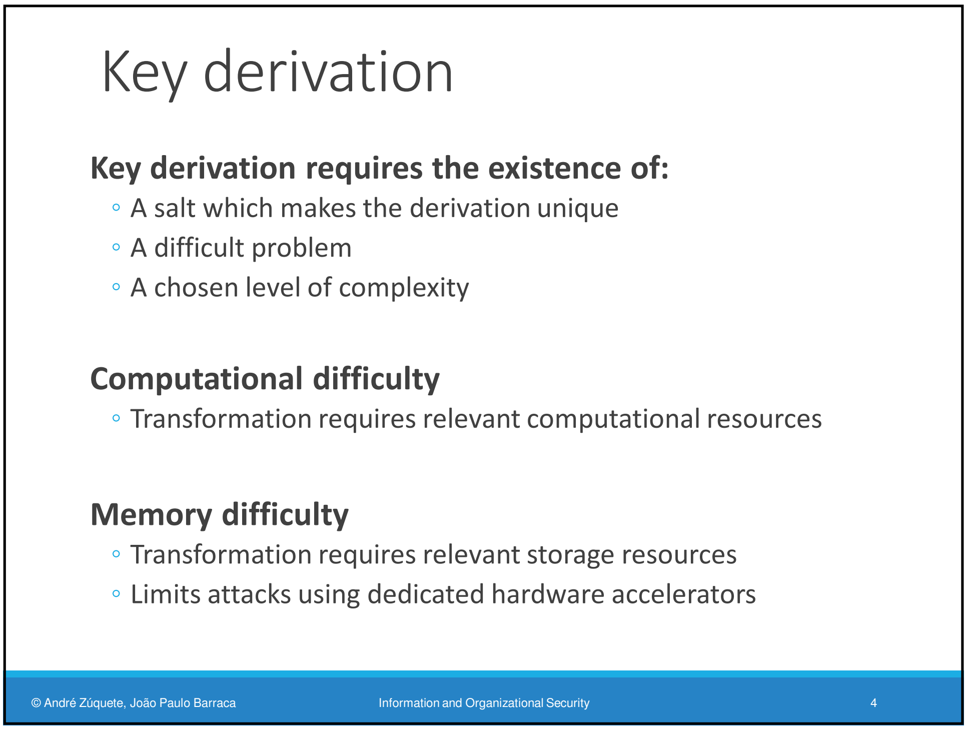
# Key derivation: purposes

**Key reinforcement: increase the security of a password**

- Usually defined by humans
- Making dictionary attacks impractical

**Key expansion: increase the dimension of a key**

- Expansion to a size that suits an algorithm
- Eventually derive other related keys for other algorithms (e.g. MAC)



# Key derivation

**Key derivation requires the existence of:**

- A salt which makes the derivation unique
- A difficult problem
- A chosen level of complexity

**Computational difficulty**

- Transformation requires relevant computational resources

**Memory difficulty**

- Transformation requires relevant storage resources
- Limits attacks using dedicated hardware accelerators

# Key derivation: PKBDF2

**Password Based Key Derivation Function 2**

**Produces a key from a password, with a chosen difficulty**

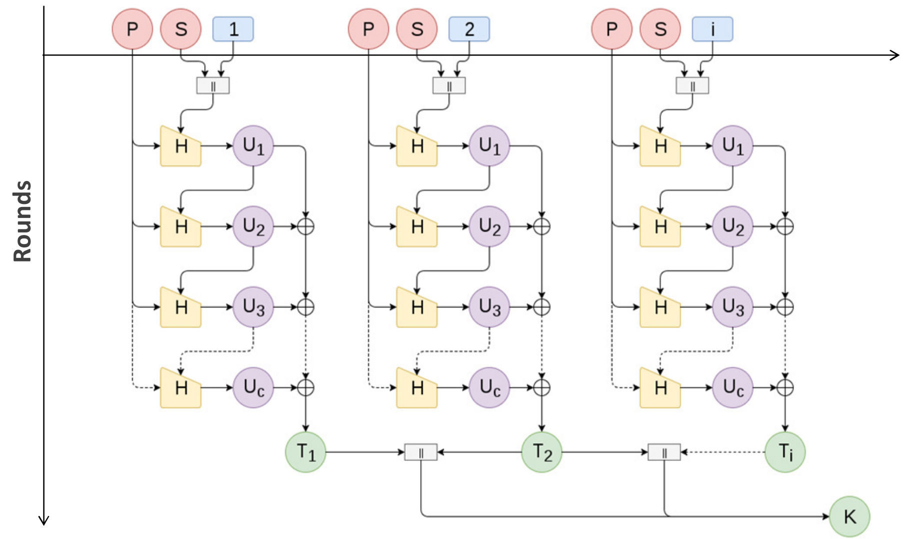
**K = PBKDF2(PRF, Salt, rounds, dim, password)**

- PRF: Pseudo-Random-Function: a digest function
- Salt: a random value
- Rounds: the computational cost (tens or hundreds of thousands)
- Dim: the size of the result required

**Operation: calculates ROUNDS x DIM operations from the PRF using the SALT and PASSWORD**

- Larger number of rounds will increase the cost

# Key Derivation: PBKDF2



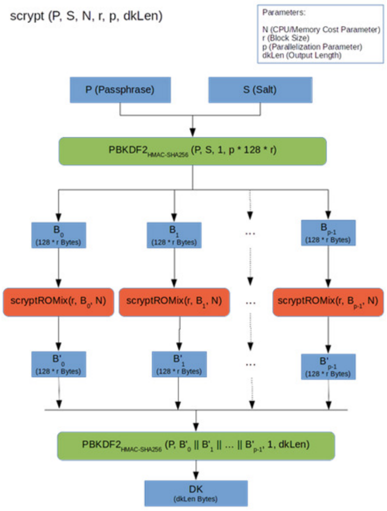
# Key Derivation: scrypt

**Produces a key with a chosen storage cost**

**K = scrypt(password, salt, n, p, dim, r, hLen, Mflen)**

- Password: a secret
- Salt: a random value
- N: the cost parameter
- P: the parallelization parameter. p ≤ ($2^{32}$– 1) * hLen / MFLen
- Dim: the size of the result
- R: the size of the blocks to use (default is 8)
- hLen: the size of the digest function (32 for SHA256)
- Mflen: bytes in the internal mix (default is 8 x R)

# Key Derivation: scrypt


scrypt (P, S, N, r, p, dkLen)
Parameters:
N (CPU/Memory Cost Parameter)
r (Block Size)
p (Parallelization Parameter)
dkLen (Output Length)
P (Passphrase)
S (Salt)
$\text{PBKDF2}_{\text{HMAC-SHA256}}$ (P, S, 1, p * 128 * r)
$B_0$ (128 * r Bytes)
$B_1$ (128 * r Bytes)
...
$B_{p-1}$ (128 * r Bytes)
scryptROMix(r, $B_0$, N)
scryptROMix(r, $B_1$, N)
...
scryptROMix(r, $B_{p-1}$, N)
$B'_0$ (128 * r Bytes)
$B'_1$ (128 * r Bytes)
...
$B'_{p-1}$ (128 * r Bytes)
$\text{PBKDF2}_{\text{HMAC-SHA256}}$ (P, $B'_0$ || $B'_1$ || ... || $B'_{p-1}$, 1, dkLen)
DK (dkLen Bytes)

# Management of Asymmetric keys

## Problems to solve

**Ensure proper and correct use of asymmetric key pairs**

**Privacy of private keys**

- To ensure authenticity
- To prevent the repudiation of digital signatures

**Correct distribution of public keys**

- To ensure confidentiality
- To ensure the correct validation of digital signatures

# Problems to solve

## Temporal evolution of entity <-> key pair mappings

**To tackle catastrophic occurrences**

- e.g. loss of private keys

**To tackle normal exploitation requirements**

- e.g. refresh of key pairs for reducing impersonation risks

# Problems to solve

## Ensure a proper generation of key pairs

**Random generation of secret values**

- So that they cannot be easily predicted

**Increase efficiency without reducing security**

- Make security mechanisms more useful
- Increase performance

# Goals

### Key pair generation

- When and how should they be generated

### Handling of private keys

- How do I maintain them private

### Distribution of public keys

- How are they correctly distributed worldwide

### Lifetime of key pairs

- When will they expire
- Until when should they be used
- How can I check the obsolesce of a key pair

# Generation of key pairs: Design principles

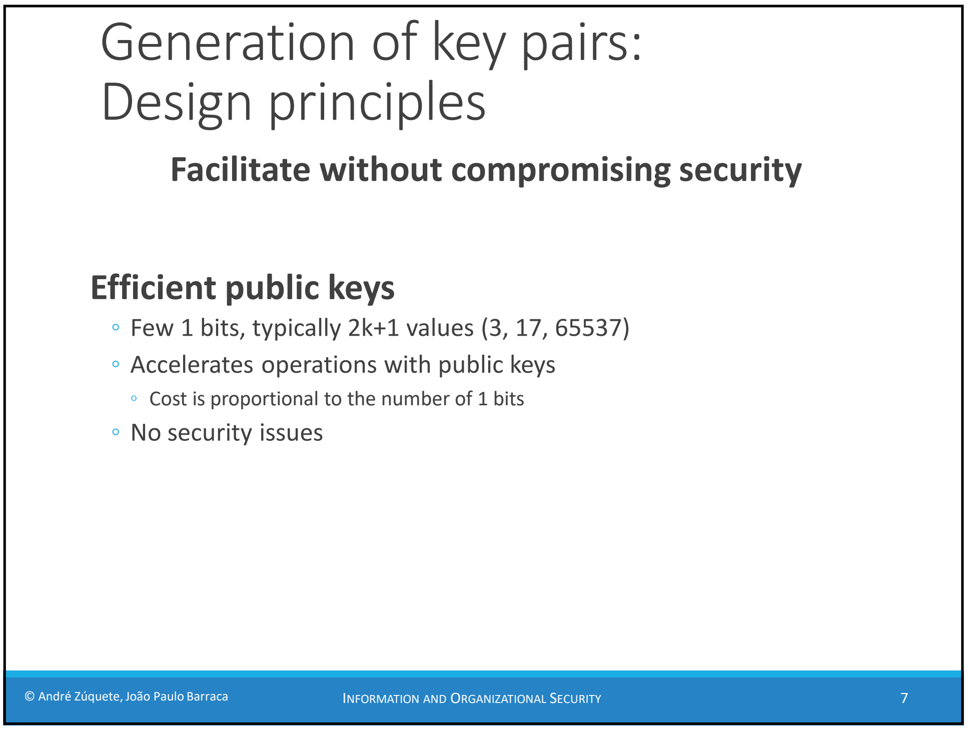
### Good random generators for producing secrets

### Result is indistinguishable from noise

- All values have equal probability
- No patterns resulting from the iteration number or previous values

### Example: Bernoulli ½ generator

- Memoryless generator
- P(b=1) = P(b=0) = ½
- Coin toss

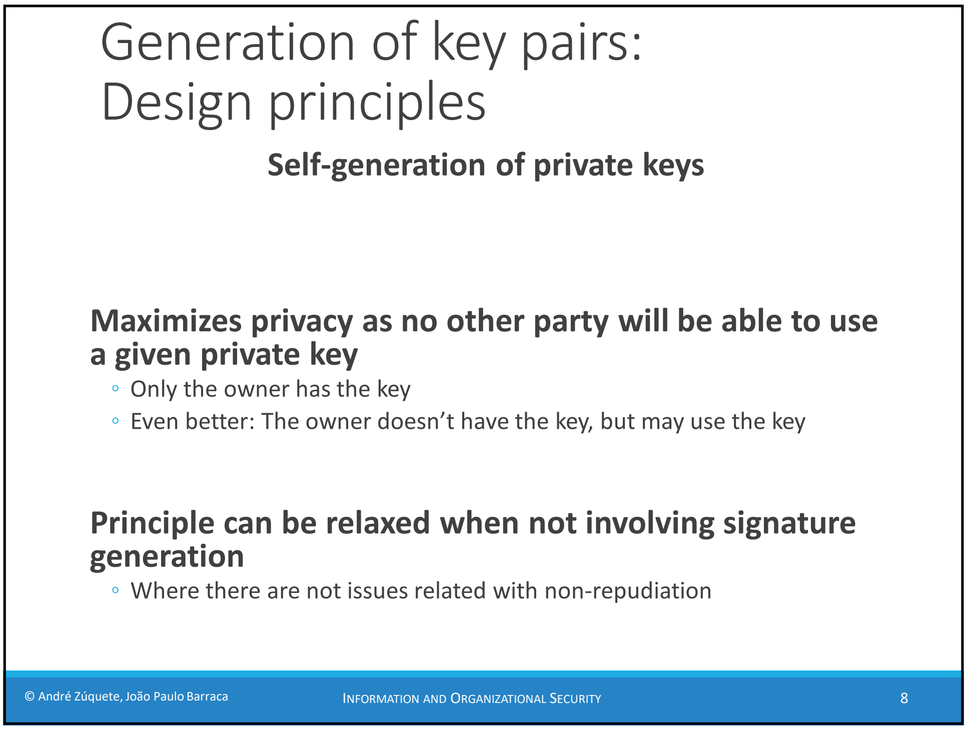


# Generation of key pairs: Design principles

**Facilitate without compromising security**

## Efficient public keys

- Few 1 bits, typically 2k+1 values (3, 17, 65537)
- Accelerates operations with public keys
  - Cost is proportional to the number of 1 bits
- No security issues



# Generation of key pairs: Design principles

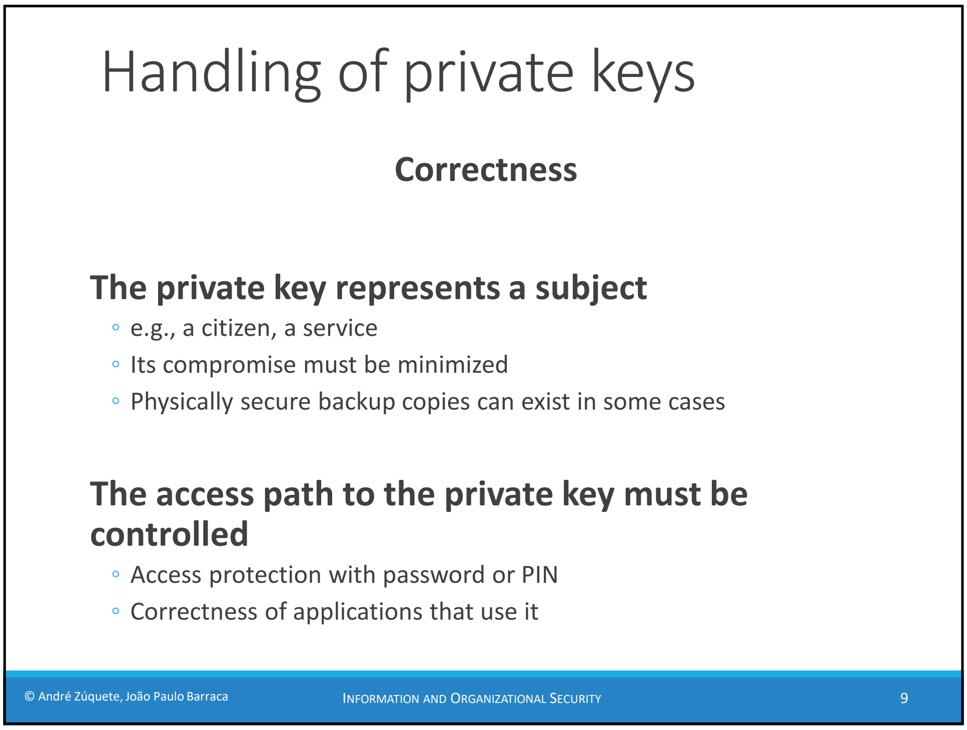
**Self-generation of private keys**

## Maximizes privacy as no other party will be able to use a given private key

- Only the owner has the key
- Even better: The owner doesn’t have the key, but may use the key

## Principle can be relaxed when not involving signature generation

- Where there are not issues related with non-repudiation



# Handling of private keys

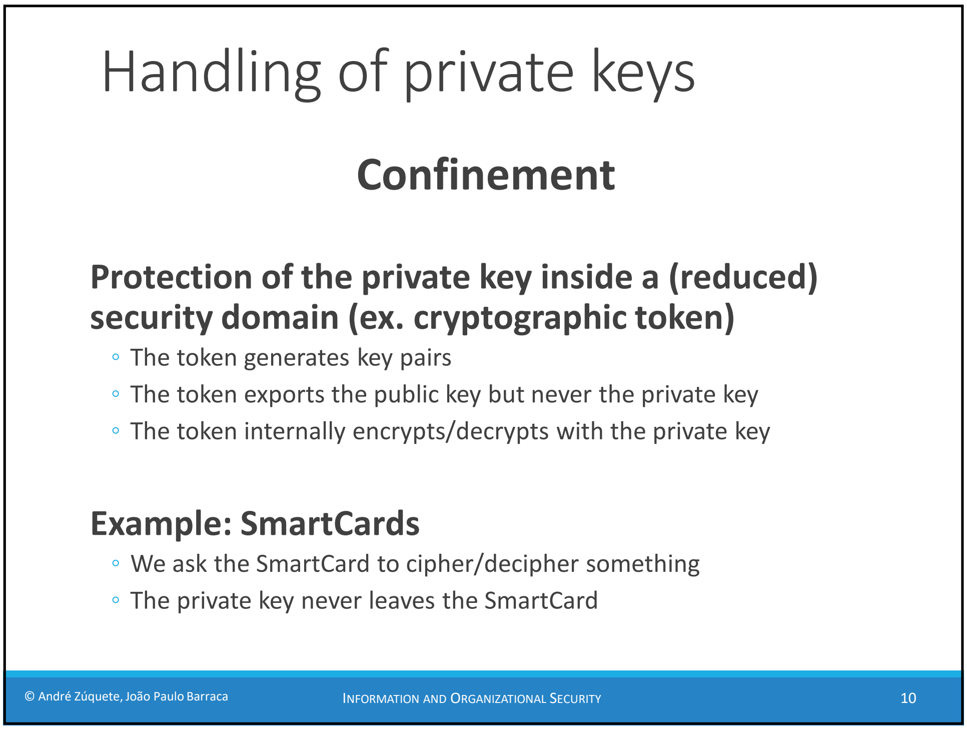
## Correctness

### The private key represents a subject

- e.g., a citizen, a service
- Its compromise must be minimized
- Physically secure backup copies can exist in some cases

### The access path to the private key must be controlled

- Access protection with password or PIN
- Correctness of applications that use it



# Handling of private keys

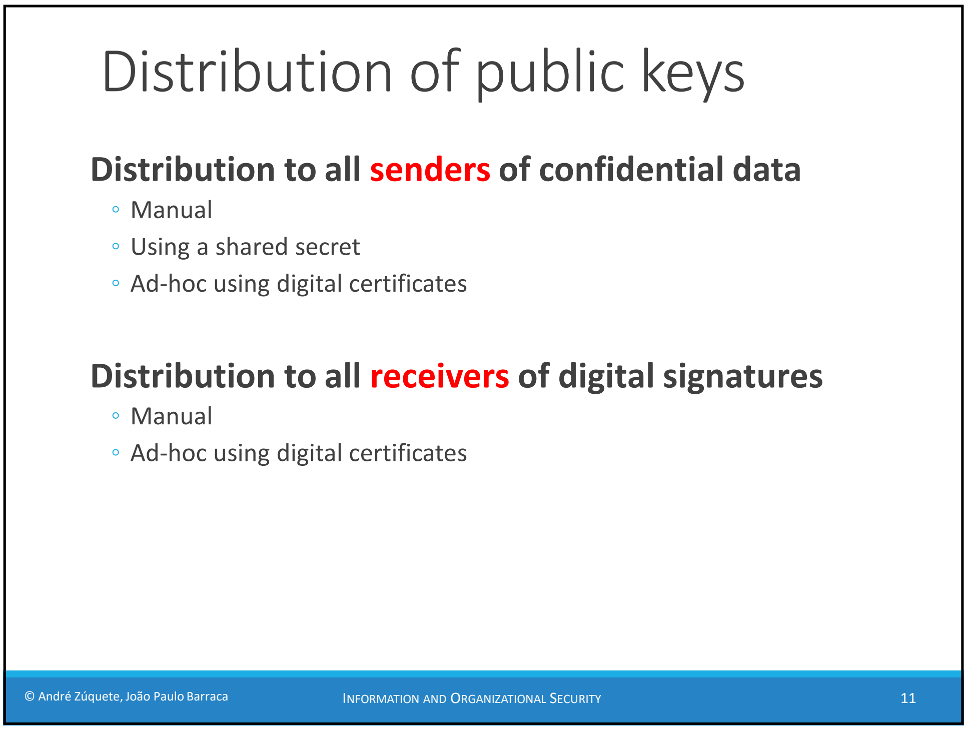
## Confinement

### Protection of the private key inside a (reduced) security domain (ex. cryptographic token)

- The token generates key pairs
- The token exports the public key but never the private key
- The token internally encrypts/decrypts with the private key

### Example: SmartCards

- We ask the SmartCard to cipher/decipher something
- The private key never leaves the SmartCard



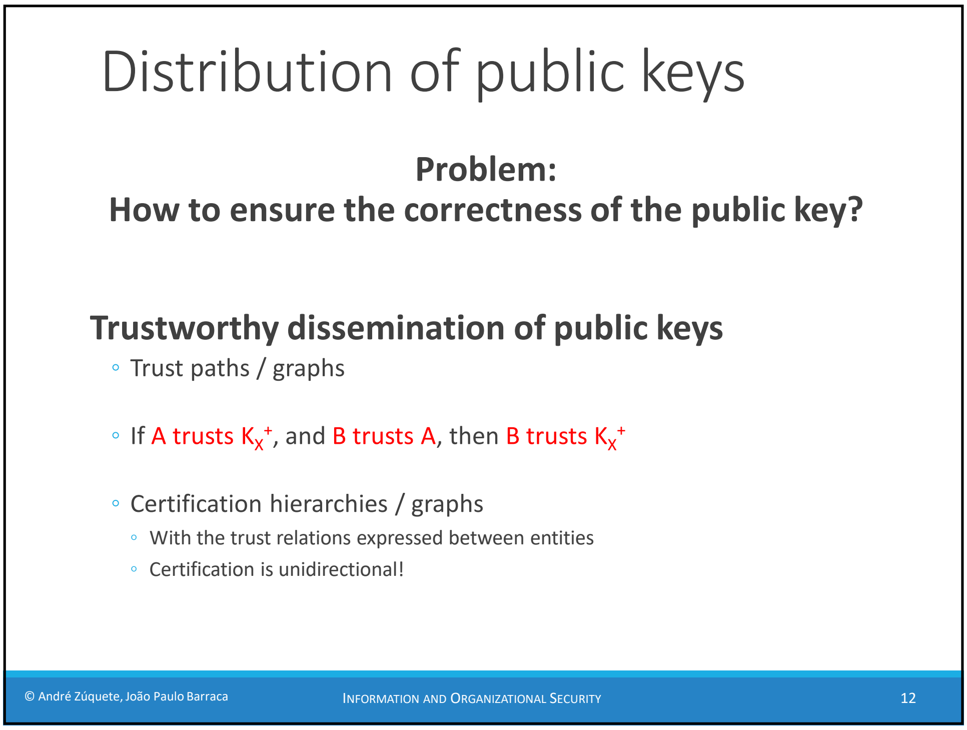
## Distribution of public keys

### Distribution to all senders of confidential data

- Manual
- Using a shared secret
- Ad-hoc using digital certificates

### Distribution to all receivers of digital signatures

- Manual
- Ad-hoc using digital certificates



## Distribution of public keys

**Problem:**
**How to ensure the correctness of the public key?**

### Trustworthy dissemination of public keys

- Trust paths / graphs
- If A trusts $K_X^+$, and B trusts A, then B trusts $K_X^+$
- Certification hierarchies / graphs
  - With the trust relations expressed between entities
  - Certification is unidirectional!

# Public key (digital) certificates

**Digital Document issued by a Certification Authority (CA)**

## Binds a public key to an entity

- Person, server or service

## Are public documents

- Do not contain private information, only public one
- Can have additional binding information (URL, Name, email, etc.)

## Are cryptographically secure

- Digitally signed by the issuer, cannot be changed

# Public key (digital) certificates

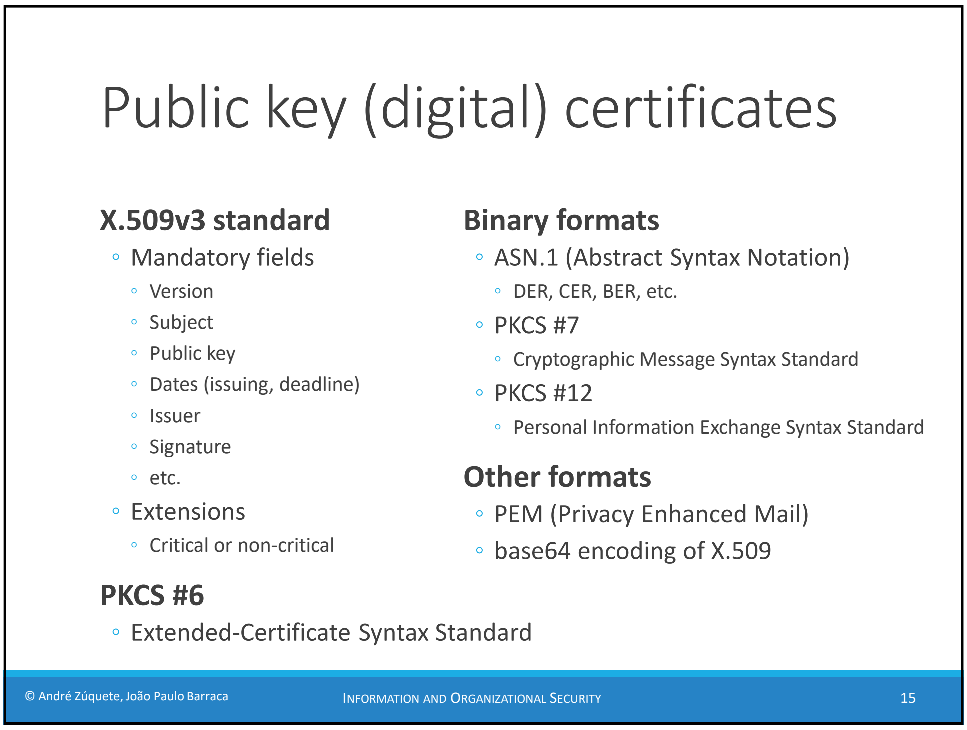
**Can be used to distribute public keys in a trustworthy way**

## A certificate receiver can validate it in many ways

- With the CA's public key
- Can also validate the identification
- Validate the validity
- Validate is the key is being properly used

## A certificate receiver trusts the behavior of the CA

- Therefore, will trust the documents they sign
- When a CA associates a certificate to A
  - If the receiver trusts the CA
  - Then it will trust that the association of A is correct



# Public key (digital) certificates

**X.509v3 standard**

- Mandatory fields
  - Version
  - Subject
  - Public key
  - Dates (issuing, deadline)
  - Issuer
  - Signature
  - etc.
- Extensions
  - Critical or non-critical

**PKCS #6**

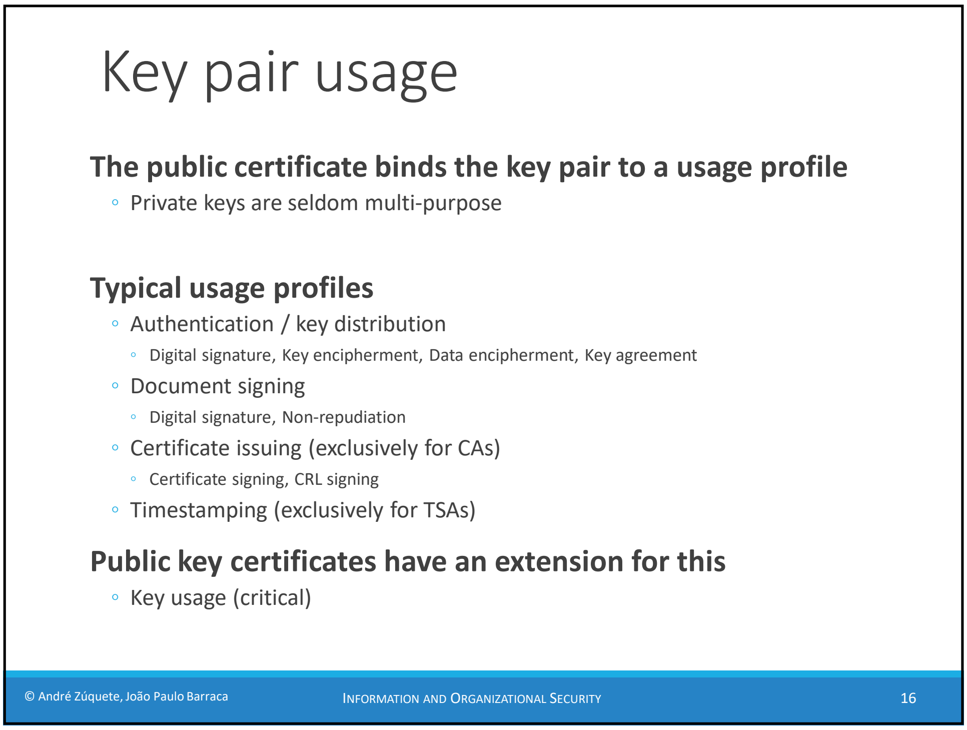
- Extended-Certificate Syntax Standard

**Binary formats**

- ASN.1 (Abstract Syntax Notation)
  - DER, CER, BER, etc.
- PKCS #7
  - Cryptographic Message Syntax Standard
- PKCS #12
  - Personal Information Exchange Syntax Standard

**Other formats**

- PEM (Privacy Enhanced Mail)
- base64 encoding of X.509



# Key pair usage

**The public certificate binds the key pair to a usage profile**

- Private keys are seldom multi-purpose

**Typical usage profiles**

- Authentication / key distribution
  - Digital signature, Key encipherment, Data encipherment, Key agreement
- Document signing
  - Digital signature, Non-repudiation
- Certificate issuing (exclusively for CAs)
  - Certificate signing, CRL signing
- Timestamping (exclusively for TSAs)

**Public key certificates have an extension for this**

- Key usage (critical)

# Certification Authorities (CA)

## Organizations that manage public key certificates

- Companies, not for profit organizations or governmental
- Have the task of validating the relation between key and identity

## Define policies and mechanisms for:

- Issuing certificates
- Revoking certificates
- Distributing certificates
- Issuing and distributing the corresponding private keys

## Manage certificate revocation lists

- Lists of revoked certificates
- Programmatic interfaces to verify the current state of a certificate

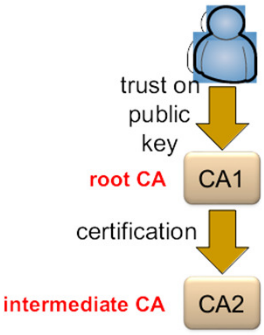
# Trusted Certification Authorities

## Intermediate CAs: CAs certified by other trusted CAs

- Using a certificate
- Enable the creation of certification hierarchies

## Trusted anchor (or certification root)

- One that has a trusted public key
- Usually implemented by self-certified certificates
  - Issuer = Subject
- Manual distribution
  - e.g., within browsers code (Firefox, Chrome, etc.), OS, distribution…



trust on public key

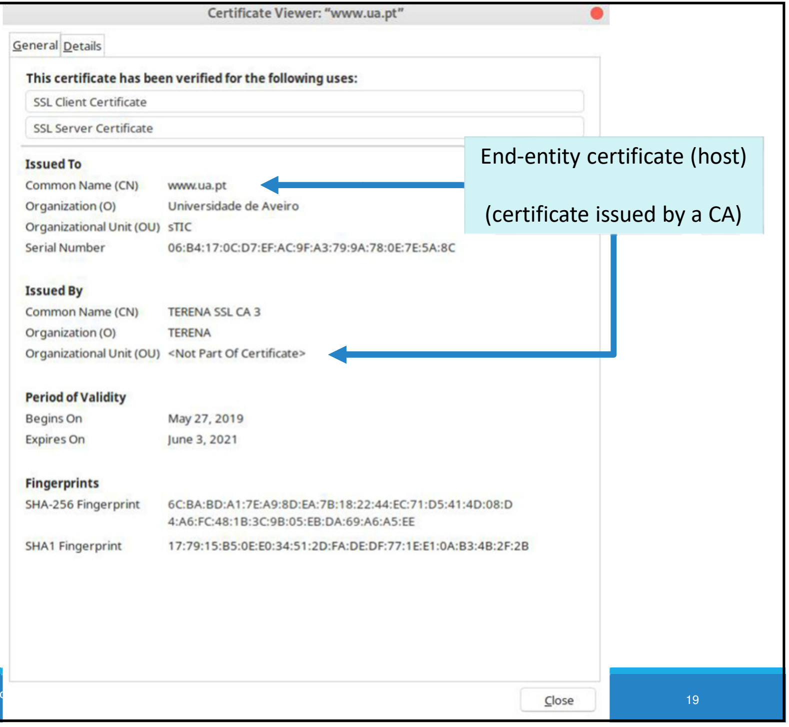
root CA

CA1

certification

intermediate CA

CA2



Certificate Viewer: "www.ua.pt"

**This certificate has been verified for the following uses:**

SSL Client Certificate

SSL Server Certificate

**Issued To**

| | |
|---|---|
| Common Name (CN) | www.ua.pt |
| Organization (O) | Universidade de Aveiro |
| Organizational Unit (OU) | sTIC |
| Serial Number | 06:B4:17:0C:D7:EF:AC:9F:A3:79:9A:78:0E:7E:5A:8C |

**Issued By**

| | |
|---|---|
| Common Name (CN) | TERENA SSL CA 3 |
| Organization (O) | TERENA |
| Organizational Unit (OU) | <Not Part Of Certificate> |

**Period of Validity**

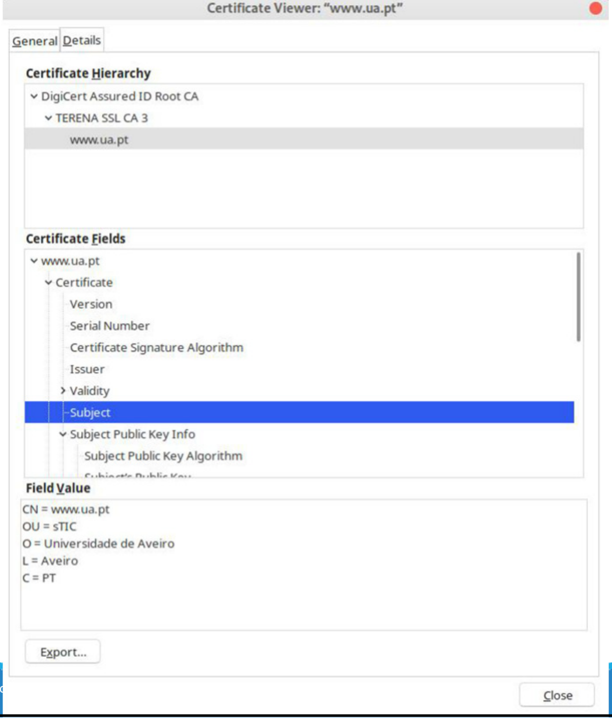
| | |
|---|---|
| Begins On | May 27, 2019 |
| Expires On | June 3, 2021 |

**Fingerprints**

| | |
|---|---|
| SHA-256 Fingerprint | 6C:BA:BD:A1:7E:A9:8D:EA:7B:18:22:44:EC:71:D5:41:4D:08:D4:A6:FC:48:1B:3C:9B:05:EB:DA:69:A6:A5:EE |
| SHA1 Fingerprint | 17:79:15:B5:0E:E0:34:51:2D:FA:DE:DF:77:1E:E1:0A:B3:4B:2F:2B |

End-entity certificate (host)

(certificate issued by a CA)



Certificate Viewer: "www.ua.pt"

**Certificate Hierarchy**

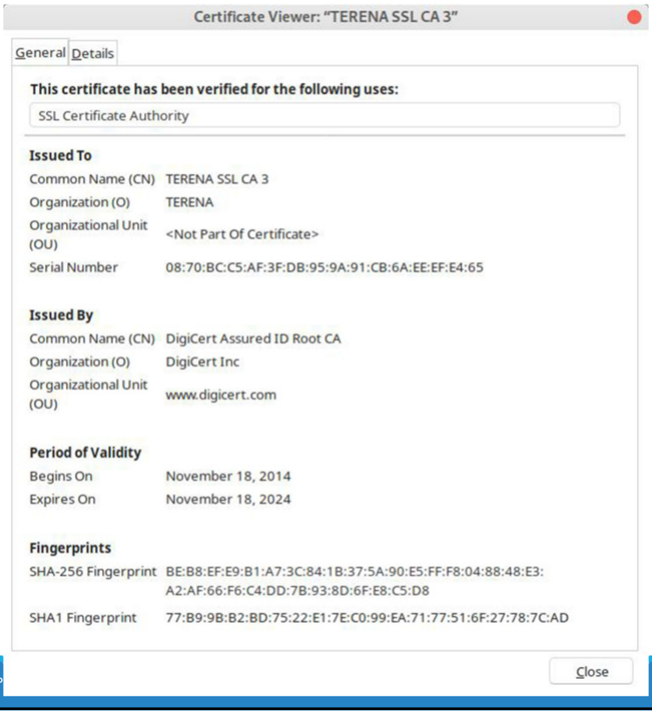
- DigiCert Assured ID Root CA
  - TERENA SSL CA 3
    - www.ua.pt

**Certificate Fields**

- www.ua.pt
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
    - Subject
    - Subject Public Key Info
      - Subject Public Key Algorithm
      - Subject's Public Key

**Field Value**

CN = www.ua.pt
OU = sTIC
O = Universidade de Aveiro
L = Aveiro
C = PT



**This certificate has been verified for the following uses:**

SSL Certificate Authority

**Issued To**

| | |
|---|---|
| Common Name (CN) | TERENA SSL CA 3 |
| Organization (O) | TERENA |
| Organizational Unit (OU) | <Not Part Of Certificate> |
| Serial Number | 08:70:BC:C5:AF:3F:DB:95:9A:91:CB:6A:EE:EF:E4:65 |

**Issued By**

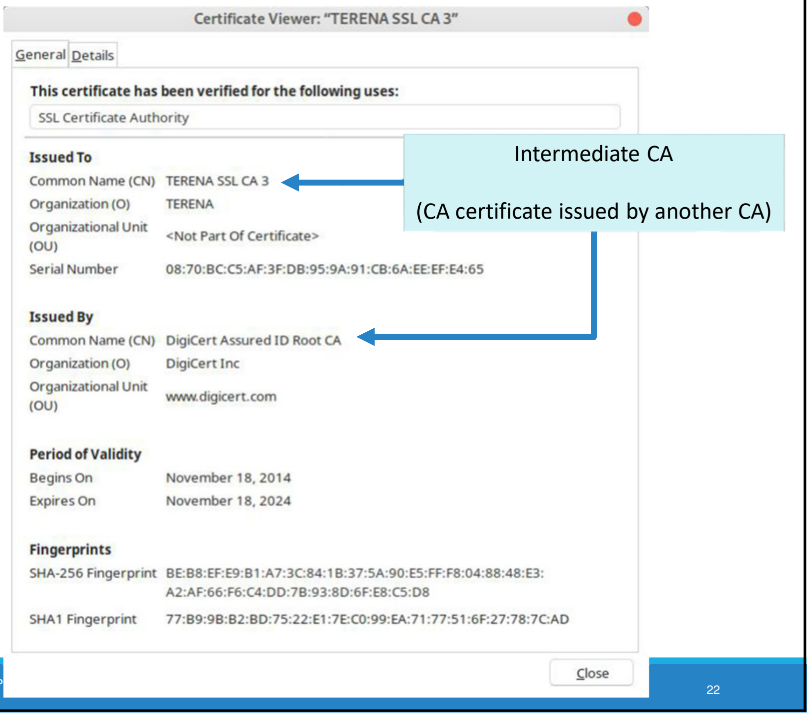
| | |
|---|---|
| Common Name (CN) | DigiCert Assured ID Root CA |
| Organization (O) | DigiCert Inc |
| Organizational Unit (OU) | www.digicert.com |

**Period of Validity**

| | |
|---|---|
| Begins On | November 18, 2014 |
| Expires On | November 18, 2024 |

**Fingerprints**

| | |
|---|---|
| SHA-256 Fingerprint | BE:B8:EF:E9:B1:A7:3C:84:1B:37:5A:90:E5:FF:F8:04:88:48:E3:<br>A2:AF:66:F6:C4:DD:7B:93:8D:6F:E8:C5:D8 |
| SHA1 Fingerprint | 77:B9:9B:B2:BD:75:22:E1:7E:C0:99:EA:71:77:51:6F:27:78:7C:AD |



**This certificate has been verified for the following uses:**

SSL Certificate Authority

Intermediate CA

(CA certificate issued by another CA)

**Issued To**

| | |
|---|---|
| Common Name (CN) | TERENA SSL CA 3 |
| Organization (O) | TERENA |
| Organizational Unit (OU) | <Not Part Of Certificate> |
| Serial Number | 08:70:BC:C5:AF:3F:DB:95:9A:91:CB:6A:EE:EF:E4:65 |

**Issued By**

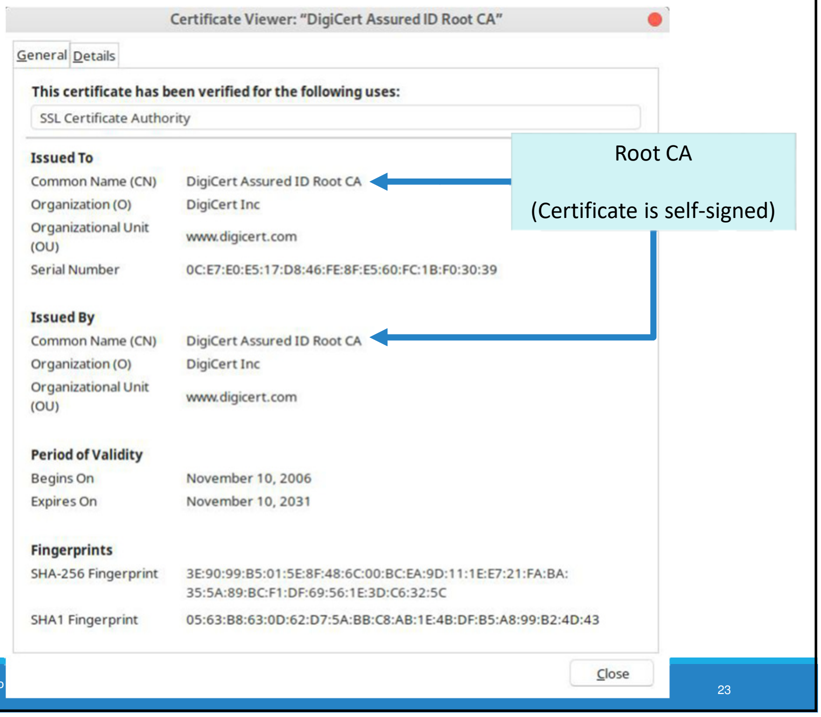
| | |
|---|---|
| Common Name (CN) | DigiCert Assured ID Root CA |
| Organization (O) | DigiCert Inc |
| Organizational Unit (OU) | www.digicert.com |

**Period of Validity**

| | |
|---|---|
| Begins On | November 18, 2014 |
| Expires On | November 18, 2024 |

**Fingerprints**

| | |
|---|---|
| SHA-256 Fingerprint | BE:B8:EF:E9:B1:A7:3C:84:1B:37:5A:90:E5:FF:F8:04:88:48:E3:<br>A2:AF:66:F6:C4:DD:7B:93:8D:6F:E8:C5:D8 |
| SHA1 Fingerprint | 77:B9:9B:B2:BD:75:22:E1:7E:C0:99:EA:71:77:51:6F:27:78:7C:AD |



**This certificate has been verified for the following uses:**

SSL Certificate Authority

**Issued To**

| | |
|---|---|
| Common Name (CN) | DigiCert Assured ID Root CA |
| Organization (O) | DigiCert Inc |
| Organizational Unit (OU) | www.digicert.com |
| Serial Number | 0C:E7:E0:E5:17:D8:46:FE:8F:E5:60:FC:1B:F0:30:39 |

**Issued By**

| | |
|---|---|
| Common Name (CN) | DigiCert Assured ID Root CA |
| Organization (O) | DigiCert Inc |
| Organizational Unit (OU) | www.digicert.com |

**Period of Validity**

| | |
|---|---|
| Begins On | November 10, 2006 |
| Expires On | November 10, 2031 |

**Fingerprints**

| | |
|---|---|
| SHA-256 Fingerprint | 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:<br>35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5C |
| SHA1 Fingerprint | 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43 |

Root CA

(Certificate is self-signed)

# Refreshing of asymmetric key pairs

## Key pairs should have a limited lifetime

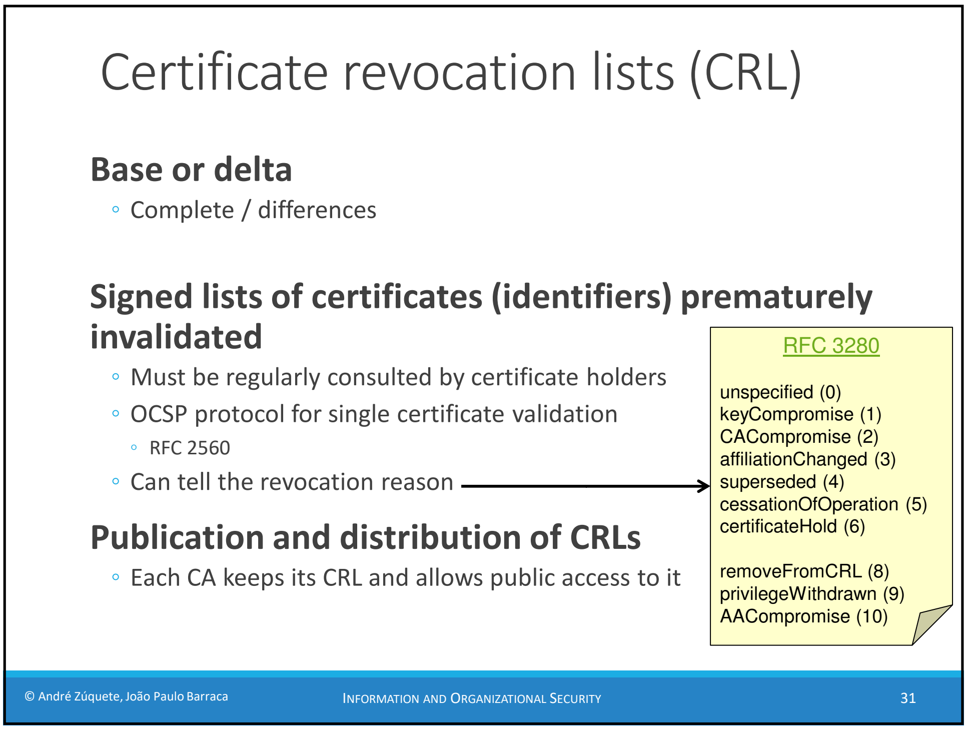
- Because private keys can be lost or discovered
- To implement a regular update policy

## Problem

- Certificates can be freely copied and distributed
- The universe of holders of certificates is unknown
  - Therefore, we cannot contact them to eliminate specific certificates

## Solutions

- Certificates with a validity period (not before, not after)
- Certificate revocation lists
  - To revoke certificates before expiring their validity



# Certificate revocation lists (CRL)

## Base or delta

- Complete / differences

## Signed lists of certificates (identifiers) prematurely invalidated

- Must be regularly consulted by certificate holders
- OCSP protocol for single certificate validation
  - RFC 2560
- Can tell the revocation reason →

## Publication and distribution of CRLs

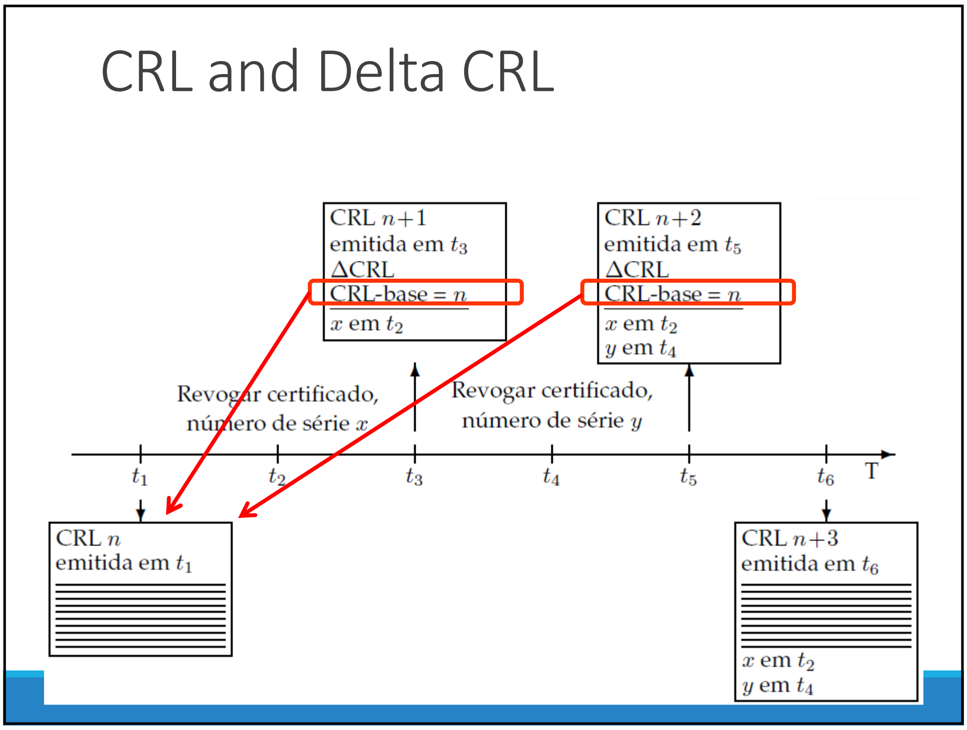
- Each CA keeps its CRL and allows public access to it

RFC 3280

unspecified (0)
keyCompromise (1)
CACompromise (2)
affiliationChanged (3)
superseded (4)
cessationOfOperation (5)
certificateHold (6)

removeFromCRL (8)
privilegeWithdrawn (9)
AACompromise (10)

# CRL and Delta CRL



CRL $n+1$
emitida em $t_3$
ΔCRL
CRL-base = $n$
$x$ em $t_2$

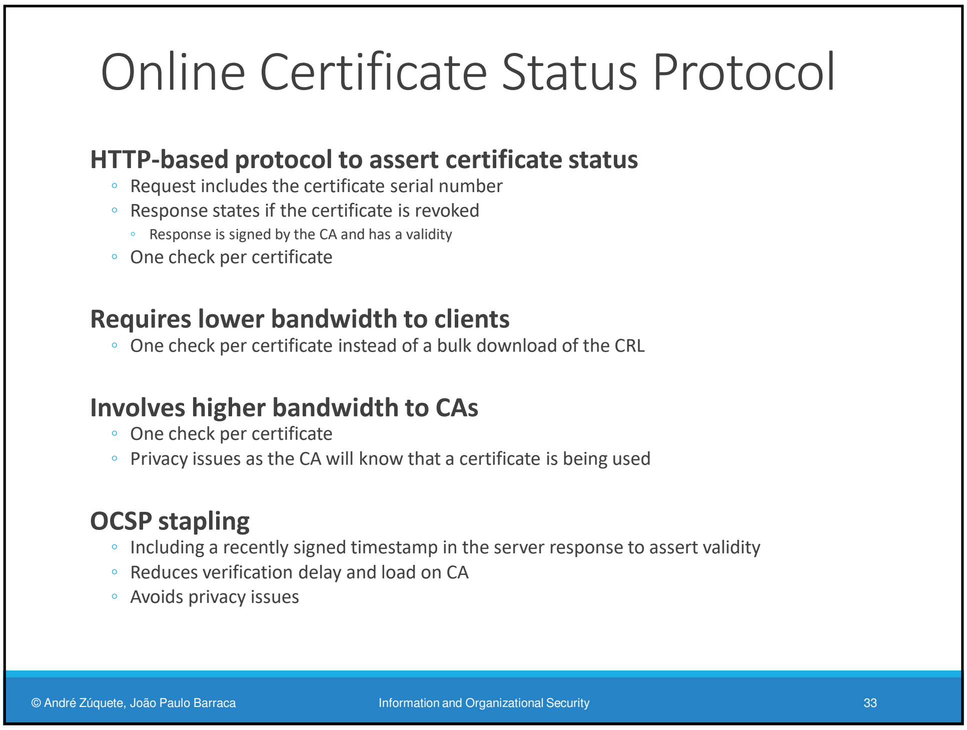
CRL $n+2$
emitida em $t_5$
ΔCRL
CRL-base = $n$
$x$ em $t_2$
$y$ em $t_4$

Revogar certificado,
número de série $x$

Revogar certificado,
número de série $y$

CRL $n$
emitida em $t_1$

CRL $n+3$
emitida em $t_6$
$x$ em $t_2$
$y$ em $t_4$



# Online Certificate Status Protocol

## HTTP-based protocol to assert certificate status

- Request includes the certificate serial number
- Response states if the certificate is revoked
  - Response is signed by the CA and has a validity
- One check per certificate

## Requires lower bandwidth to clients

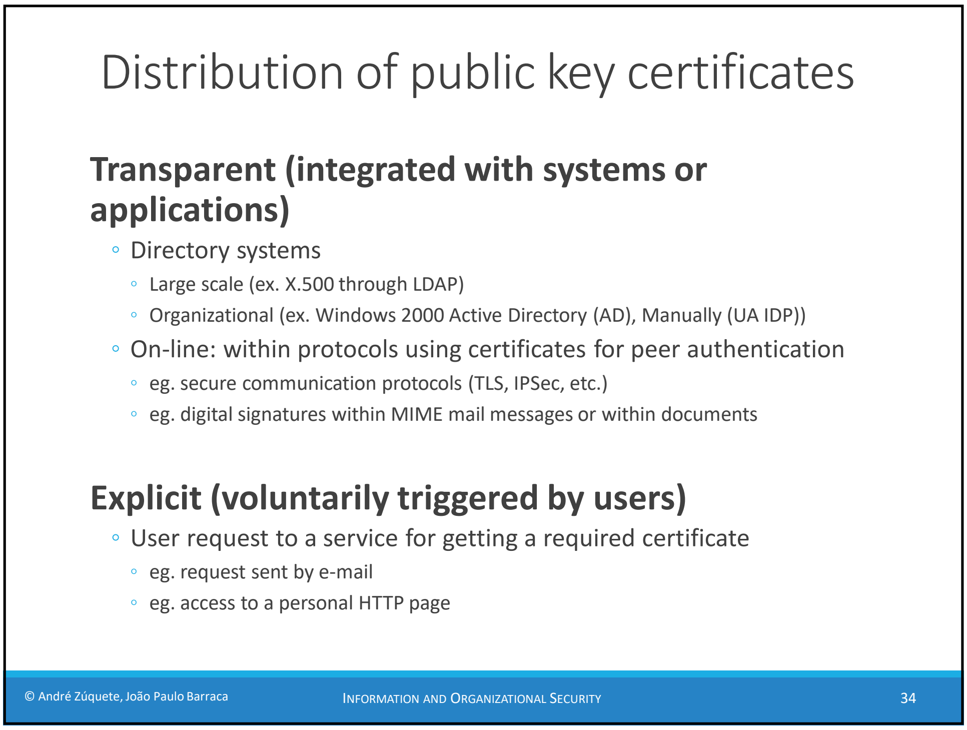
- One check per certificate instead of a bulk download of the CRL

## Involves higher bandwidth to CAs

- One check per certificate
- Privacy issues as the CA will know that a certificate is being used

## OCSP stapling

- Including a recently signed timestamp in the server response to assert validity
- Reduces verification delay and load on CA
- Avoids privacy issues



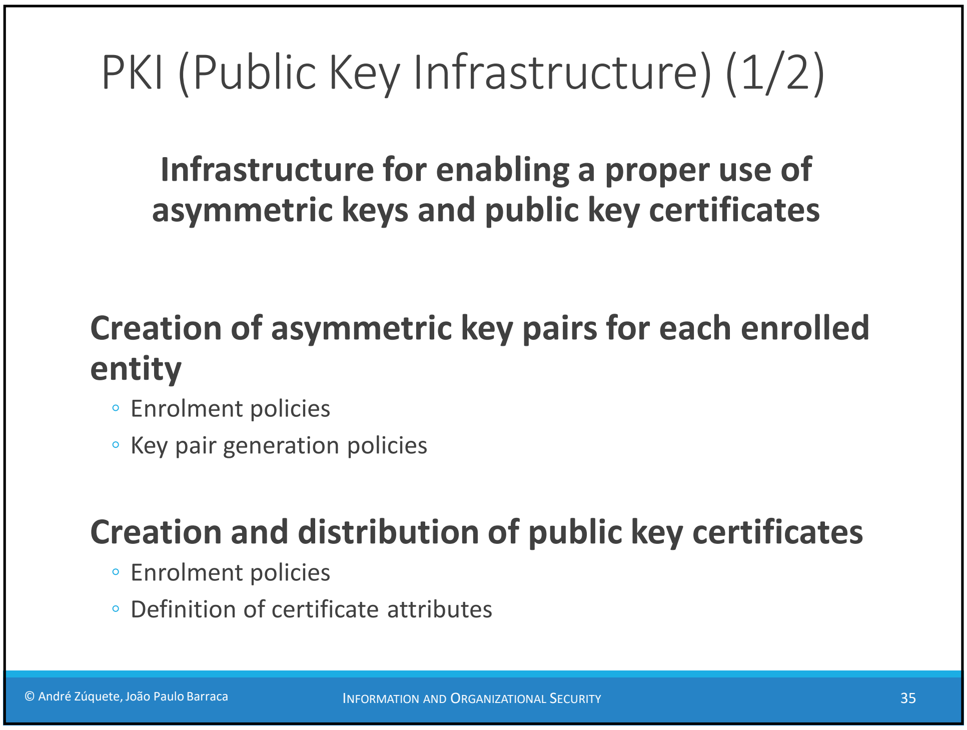
# Distribution of public key certificates

## Transparent (integrated with systems or applications)

- Directory systems
  - Large scale (ex. X.500 through LDAP)
  - Organizational (ex. Windows 2000 Active Directory (AD), Manually (UA IDP))
- On-line: within protocols using certificates for peer authentication
  - eg. secure communication protocols (TLS, IPSec, etc.)
  - eg. digital signatures within MIME mail messages or within documents

## Explicit (voluntarily triggered by users)

- User request to a service for getting a required certificate
  - eg. request sent by e-mail
  - eg. access to a personal HTTP page



# PKI (Public Key Infrastructure) (1/2)

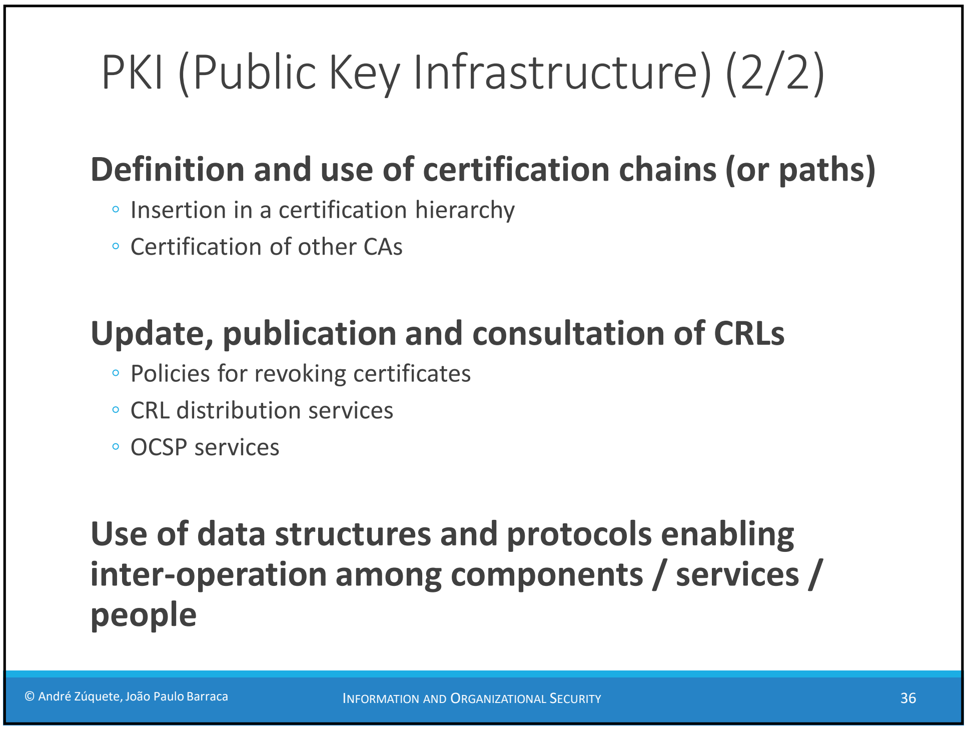
**Infrastructure for enabling a proper use of asymmetric keys and public key certificates**

**Creation of asymmetric key pairs for each enrolled entity**

- Enrolment policies
- Key pair generation policies

**Creation and distribution of public key certificates**

- Enrolment policies
- Definition of certificate attributes



# PKI (Public Key Infrastructure) (2/2)

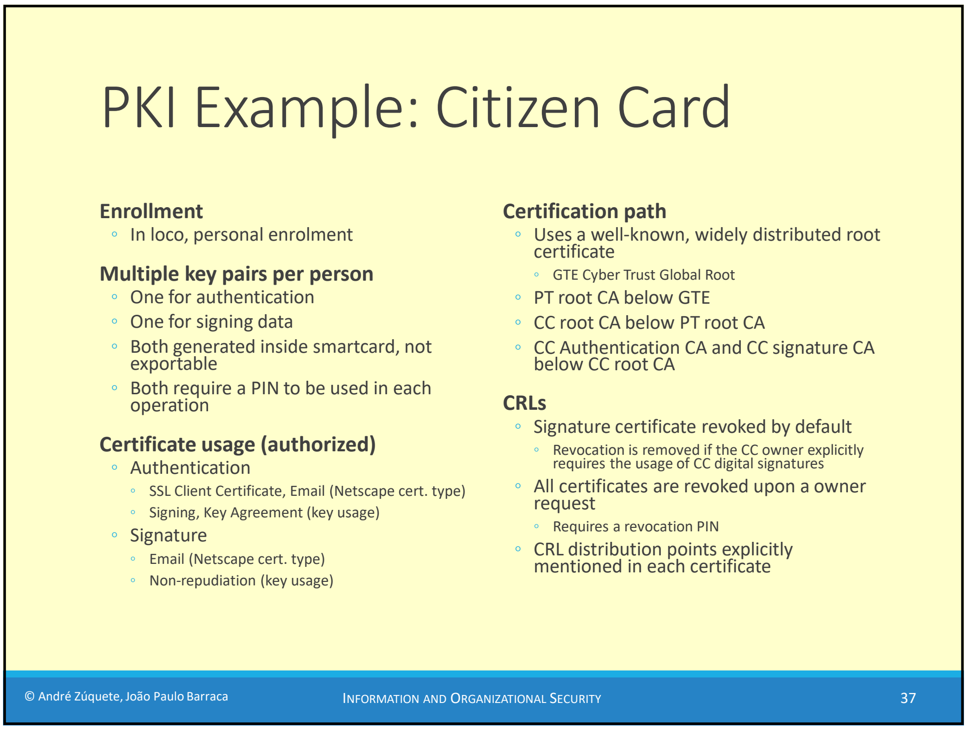
**Definition and use of certification chains (or paths)**

- Insertion in a certification hierarchy
- Certification of other CAs

**Update, publication and consultation of CRLs**

- Policies for revoking certificates
- CRL distribution services
- OCSP services

**Use of data structures and protocols enabling inter-operation among components / services / people**



# PKI Example: Citizen Card

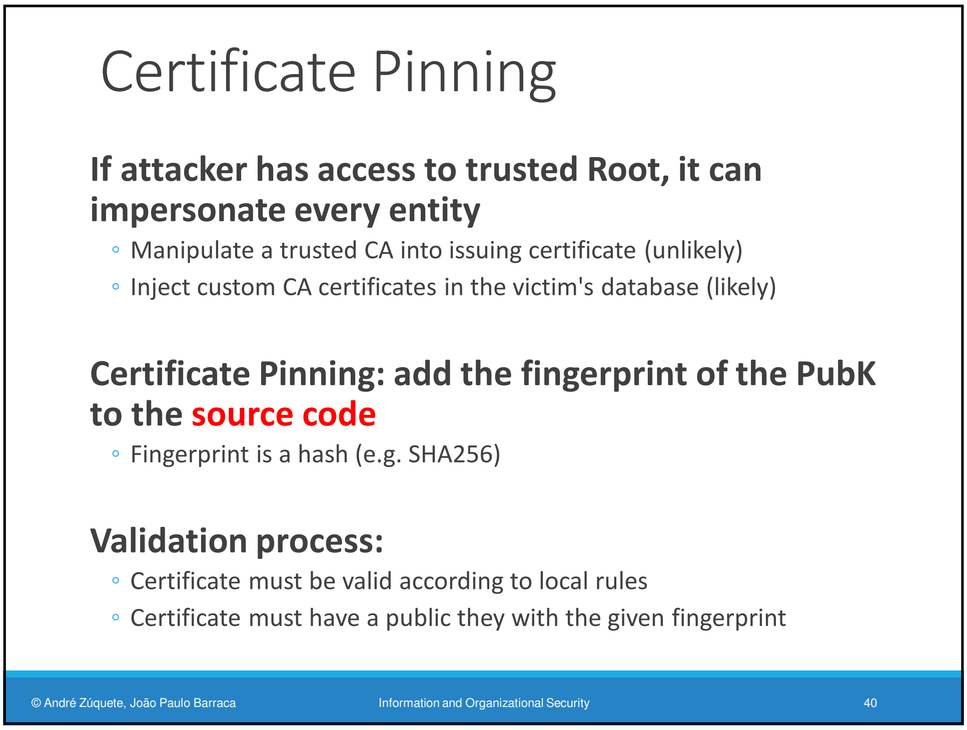
**Enrollment**
- In loco, personal enrolment

**Multiple key pairs per person**
- One for authentication
- One for signing data
- Both generated inside smartcard, not exportable
- Both require a PIN to be used in each operation

**Certificate usage (authorized)**
- Authentication
  - SSL Client Certificate, Email (Netscape cert. type)
  - Signing, Key Agreement (key usage)
- Signature
  - Email (Netscape cert. type)
  - Non-repudiation (key usage)

**Certification path**
- Uses a well-known, widely distributed root certificate
  - GTE Cyber Trust Global Root
- PT root CA below GTE
- CC root CA below PT root CA
- CC Authentication CA and CC signature CA below CC root CA

**CRLs**
- Signature certificate revoked by default
  - Revocation is removed if the CC owner explicitly requires the usage of CC digital signatures
- All certificates are revoked upon a owner request
  - Requires a revocation PIN
- CRL distribution points explicitly mentioned in each certificate

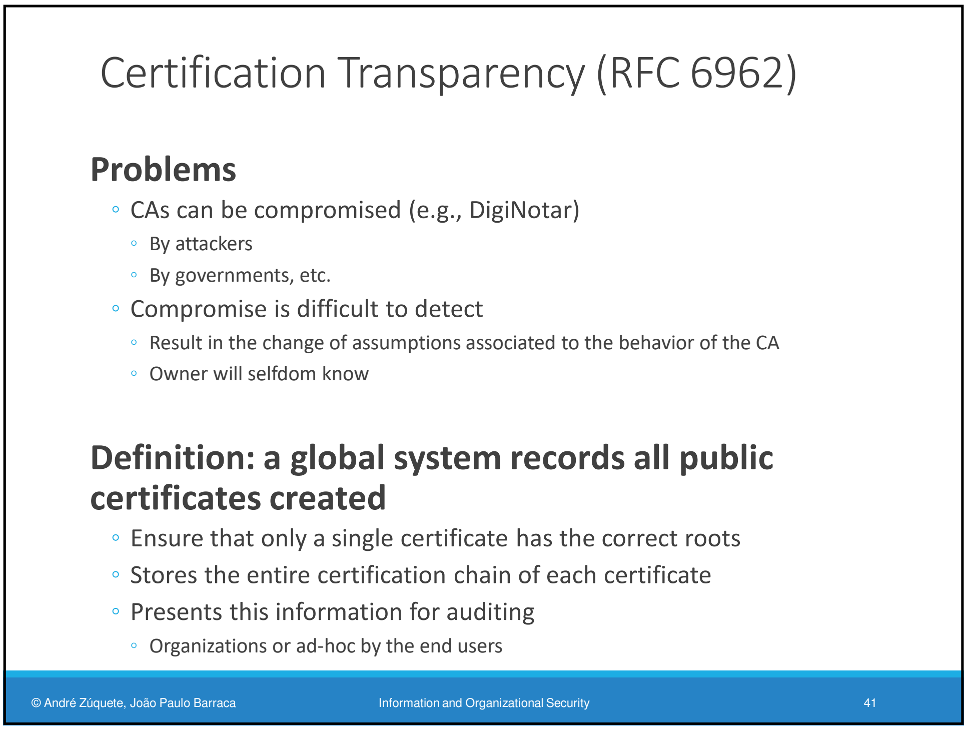


# Certificate Pinning

**If attacker has access to trusted Root, it can impersonate every entity**
- Manipulate a trusted CA into issuing certificate (unlikely)
- Inject custom CA certificates in the victim's database (likely)

**Certificate Pinning: add the fingerprint of the PubK to the source code**
- Fingerprint is a hash (e.g. SHA256)

**Validation process:**
- Certificate must be valid according to local rules
- Certificate must have a public they with the given fingerprint



# Certification Transparency (RFC 6962)

## Problems

- CAs can be compromised (e.g., DigiNotar)
  - By attackers
  - By governments, etc.
- Compromise is difficult to detect
  - Result in the change of assumptions associated to the behavior of the CA
  - Owner will selfdom know

## Definition: a global system records all public certificates created

- Ensure that only a single certificate has the correct roots
- Stores the entire certification chain of each certificate
- Presents this information for auditing
  - Organizations or ad-hoc by the end users

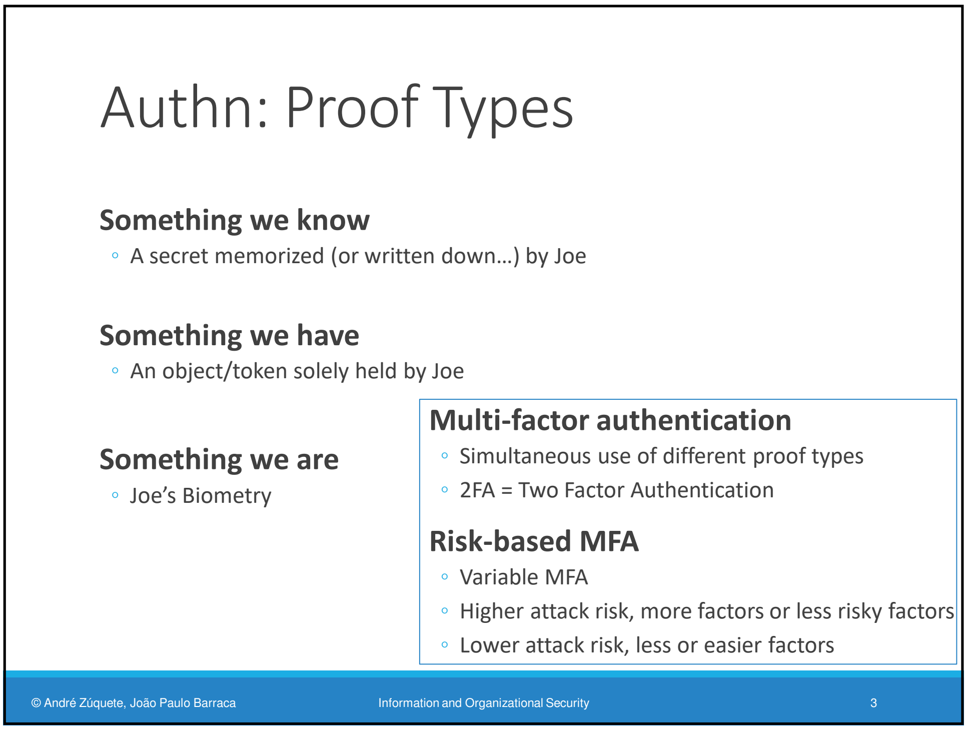
# Authentication Mechanisms and Protocols

## Authentication (Authn)

**Proof that an entity has an attribute it claims to have**

- Hi, I'm Joe
- Prove it!
- Here is my proof,
  calculated with Joe's credentials that I've agreed with you
- Proof accepted/not accepted

- Hi, I'm over 18
- Prove it!
- Here is a claim issued by a competent authority,
  which I can also prove that I'm the owner
- Proof and claim accepted/not accepted



# Authn: Proof Types

**Something we know**

- A secret memorized (or written down…) by Joe

**Something we have**

- An object/token solely held by Joe

**Something we are**

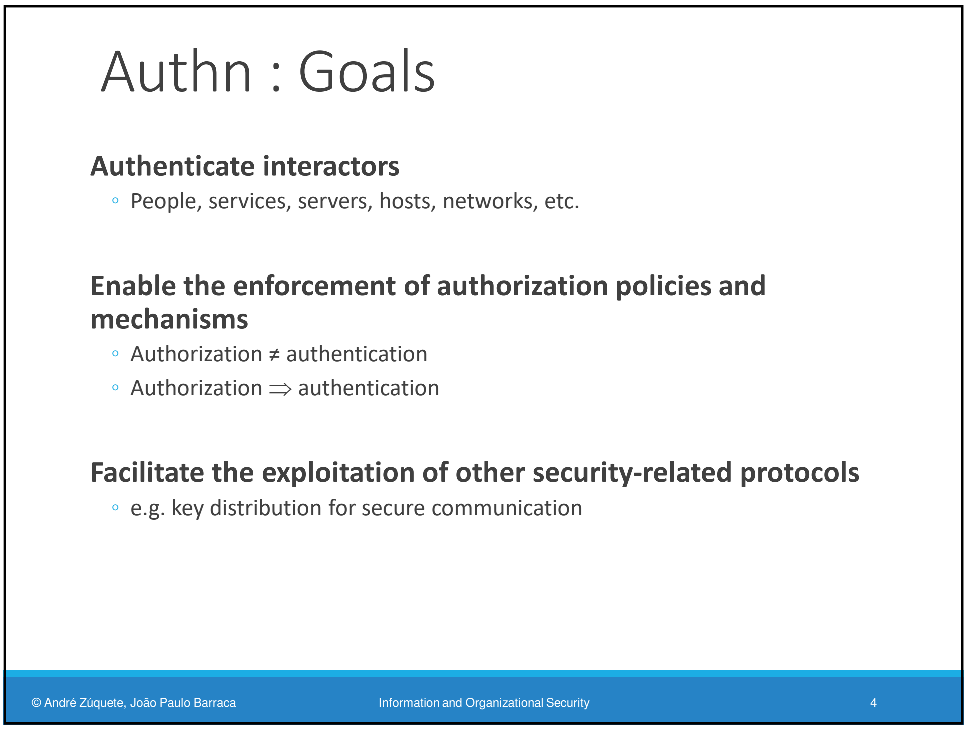
- Joe's Biometry

**Multi-factor authentication**

- Simultaneous use of different proof types
- 2FA = Two Factor Authentication

**Risk-based MFA**

- Variable MFA
- Higher attack risk, more factors or less risky factors
- Lower attack risk, less or easier factors



# Authn : Goals

**Authenticate interactors**

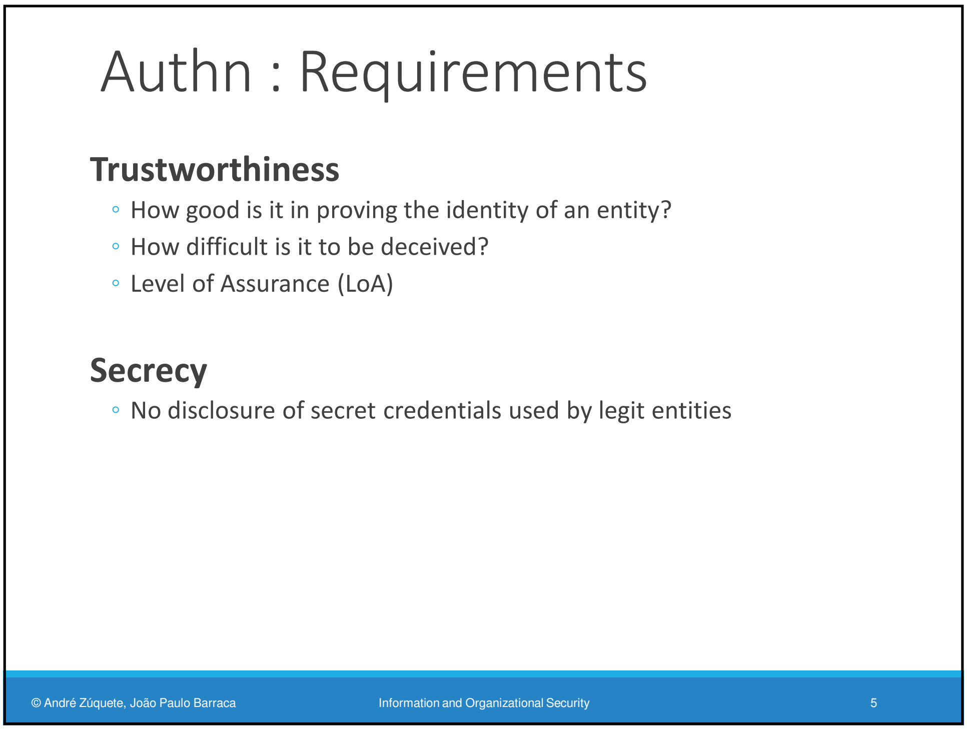
- People, services, servers, hosts, networks, etc.

**Enable the enforcement of authorization policies and mechanisms**

- Authorization ≠ authentication
- Authorization ⇒ authentication

**Facilitate the exploitation of other security-related protocols**

- e.g. key distribution for secure communication



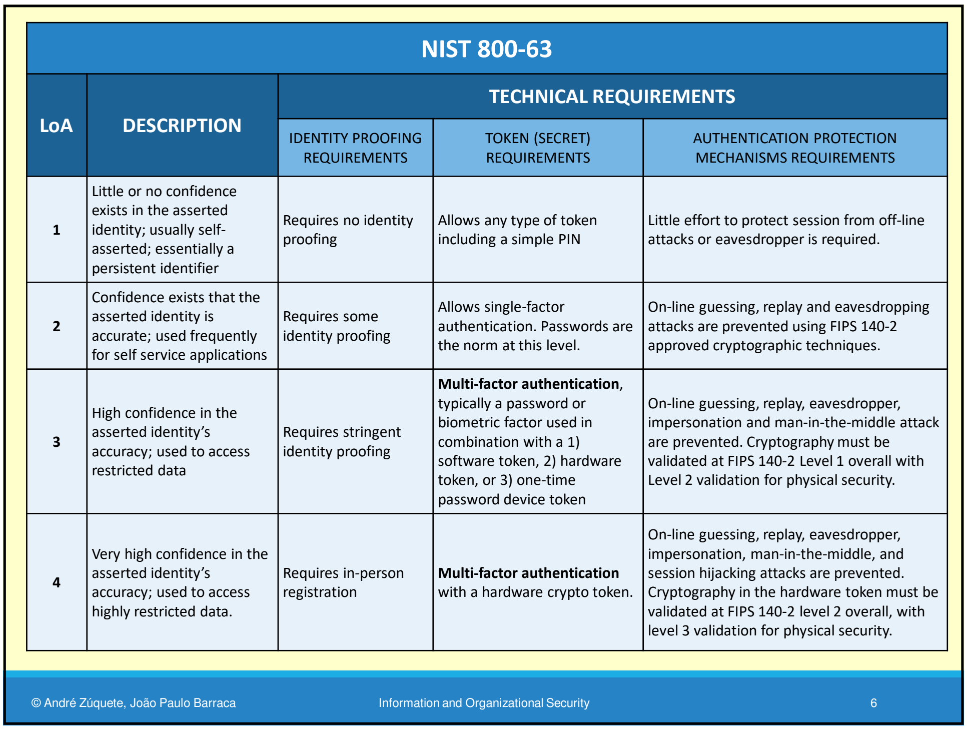
# Authn : Requirements

## Trustworthiness

- How good is it in proving the identity of an entity?
- How difficult is it to be deceived?
- Level of Assurance (LoA)

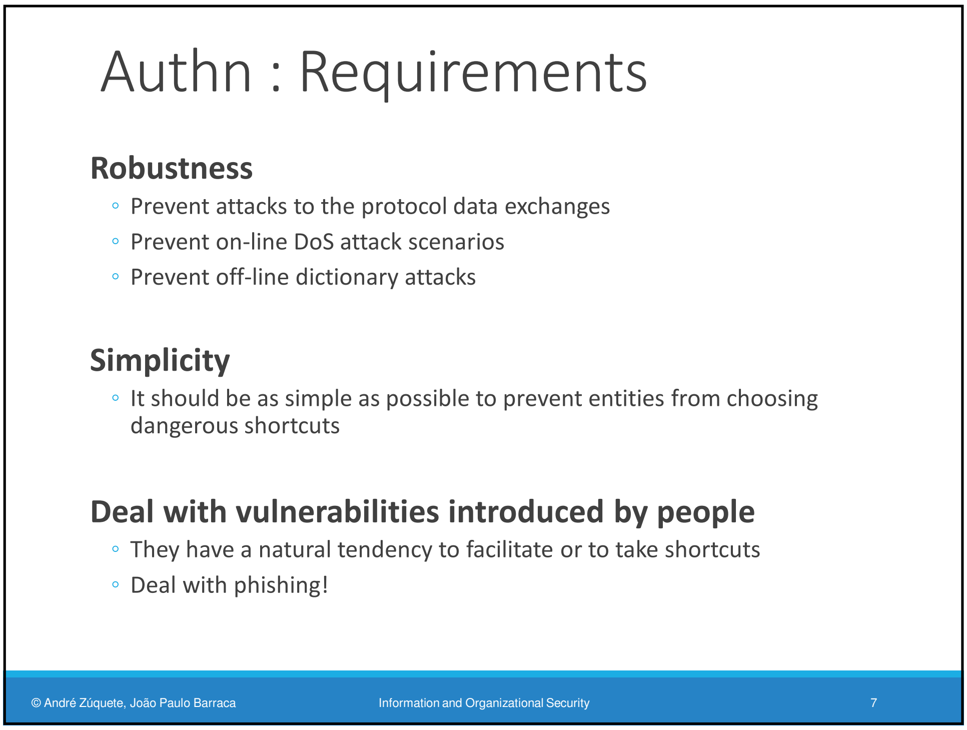
## Secrecy

- No disclosure of secret credentials used by legit entities



## NIST 800-63

| LoA | DESCRIPTION | TECHNICAL REQUIREMENTS | | |
|---|---|---|---|---|
| | | IDENTITY PROOFING REQUIREMENTS | TOKEN (SECRET) REQUIREMENTS | AUTHENTICATION PROTECTION MECHANISMS REQUIREMENTS |
| 1 | Little or no confidence exists in the asserted identity; usually self-asserted; essentially a persistent identifier | Requires no identity proofing | Allows any type of token including a simple PIN | Little effort to protect session from off-line attacks or eavesdropper is required. |
| 2 | Confidence exists that the asserted identity is accurate; used frequently for self service applications | Requires some identity proofing | Allows single-factor authentication. Passwords are the norm at this level. | On-line guessing, replay and eavesdropping attacks are prevented using FIPS 140-2 approved cryptographic techniques. |
| 3 | High confidence in the asserted identity's accuracy; used to access restricted data | Requires stringent identity proofing | **Multi-factor authentication**, typically a password or biometric factor used in combination with a 1) software token, 2) hardware token, or 3) one-time password device token | On-line guessing, replay, eavesdropper, impersonation and man-in-the-middle attack are prevented. Cryptography must be validated at FIPS 140-2 Level 1 overall with Level 2 validation for physical security. |
| 4 | Very high confidence in the asserted identity's accuracy; used to access highly restricted data. | Requires in-person registration | **Multi-factor authentication** with a hardware crypto token. | On-line guessing, replay, eavesdropper, impersonation, man-in-the-middle, and session hijacking attacks are prevented. Cryptography in the hardware token must be validated at FIPS 140-2 level 2 overall, with level 3 validation for physical security. |



# Authn : Requirements

**Robustness**

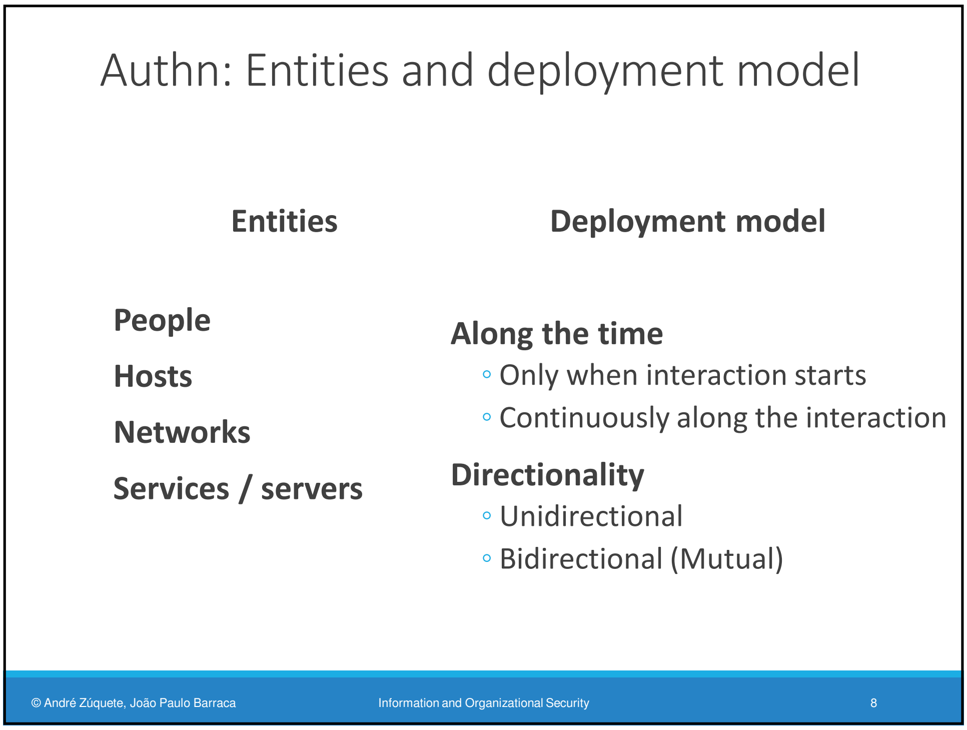
- Prevent attacks to the protocol data exchanges
- Prevent on-line DoS attack scenarios
- Prevent off-line dictionary attacks

**Simplicity**

- It should be as simple as possible to prevent entities from choosing dangerous shortcuts

**Deal with vulnerabilities introduced by people**

- They have a natural tendency to facilitate or to take shortcuts
- Deal with phishing!



# Authn: Entities and deployment model

| **Entities** | **Deployment model** |
|---|---|
| **People**<br>**Hosts**<br>**Networks**<br>**Services / servers** | **Along the time**<br>◦ Only when interaction starts<br>◦ Continuously along the interaction<br>**Directionality**<br>◦ Unidirectional<br>◦ Bidirectional (Mutual) |

# Authn interactions: Basic approaches

**Direct approach**

1. Provide credentials
2. Wait for verdict

- Advantage: no computations by the presenter
- Disadvantage: credentials can be exposed to malicious validators

**Challenge-response approach**

1. Get challenge
2. Provide a response computed from the challenge and the credentials
3. Wait for verdict

- Advantage: credentials are not exposed to malicious validators
- Disadvantage: requires computations by the presenter

# Authn of subjects:
# Direct approach w/ known password

**A password is checked against a value previously stored**

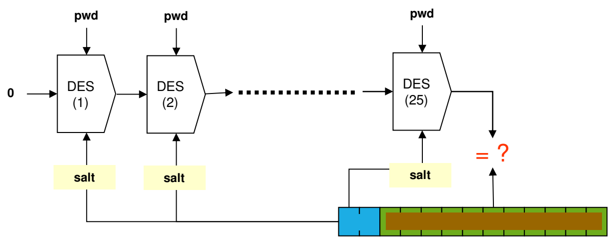
- For a claimed identity (username)

**Personal stored value:**

- Transformed by a unidirectional function
- Windows: digest function
- UNIX: DES hash + salt
- Linux: MD5 + salt
  - hash is configurable

**Optimal: PBKDF2, Script with high complexity**

# Authentication of subjects:
# Direct approach w/ known password



DES hash = $DES_{pwd}^{25}(0)$

$DES_k^n(x) = DES_k(DES_k^{n-1}(x))$

Permutation of 12 subkeys´ bit pairs with salt (12 bits)

# Authn of subjects:
# Direct approach w/ known password

## Advantage

- Simplicity!

## Problems

- Usage of weak passwords
  - Enable dictionary attacks
- Transmission of passwords along insecure communication channels
  - Eavesdroppers can easily learn the password
  - e.g. Unix remote services, PAP

Top Ten 2017 from Splashdata

1. 123456
2. Password
3. 12345678
4. qwerty
5. 12345
6. 123456789
7. letmein
8. 1234567
9. football
10. iloveyou

# Authn of people: Direct approach with biometrics

## People get authenticated using body measures

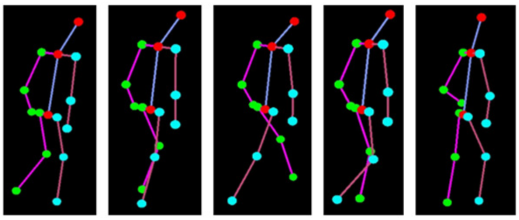
- Biometric samples
- Fingerprint, iris, face geometry, voice timber, manual writing, vein matching, etc.

## Measures are compared with personal records

- Biometric references (or template)
- Registered in the system with a previous enrolment procedure

## Identification vs authentication

- Identification: 1-to-many check for a match
- Authentication: 1-to-1 check for a match



# Authn of people:
# Direct approach with biometrics

## Advantages

- People do not need to use memory, or carry something
  - Just be their self
- People cannot choose weak passwords
  - In fact, they don't choose anything
- Authentication credentials cannot be transferred to others
  - One cannot delegate its own authentication

# Authentication of people:
# Direct approach with biometrics

## Problems

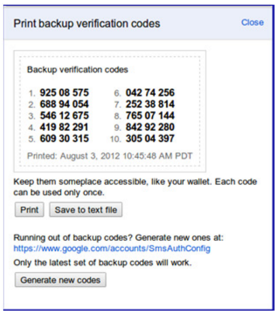
- Biometric methods are still incipient
  - In many cases it can be fooled with ease (Face Recognition, Fingerprint)
- People cannot change credentials
  - If the credentials or templates are stolen
- Credentials cannot be transferred between individuals
  - If it is required in extraordinary scenarios
- Can pose risks to individuals
  - Physical integrity can be compromised by an attacker in order to acquire biometric data
- It is not easy to be implemented in remote systems
  - It is mandatory to have secure and trusted biometric acquisition devices
- Biometrics can reveal other personal secrets
  - Diseases

# Authn of subjects:
# Direct approach with one-time passwords

## One-Time Passwords = Secrets that can be used only once

- Pre-distributed directly, or the result of a generator function

## Example: Bank codes, Google Backup Codes



Print backup verification codes — Close

Backup verification codes

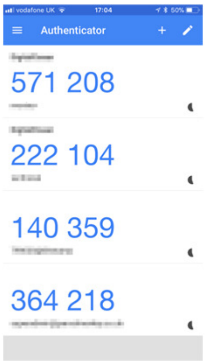
```
 1. 925 08 575     6. 042 74 256
 2. 688 94 054     7. 252 38 814
 3. 546 12 675     8. 765 07 144
 4. 419 82 291     9. 842 92 280
 5. 609 30 315    10. 305 04 397
```

Printed: August 3, 2012 10:45:48 AM PDT

Keep them someplace accessible, like your wallet. Each code can be used only once.

Running out of backup codes? Generate new ones at: https://www.google.com/accounts/SmsAuthConfig

Only the latest set of backup codes will work.



571 208

222 104

140 359

364 218

https://www.montepio.pt/SitePublico/pt_PT/particulares/montepio24/cartao-matriz.page?altcode=10006P

# Authn of subjects:
# Direct approach with one-time passwords

## Advantages

- Can be eavesdropped, allowing its use in channels without encryption
- Can be chosen by the authenticator, which may enforce a given complexity
- Can depend on a shared password

## Problems

- Interacting entities need to know which password to use on each occasion
  - Implies some form of synchronization (e.g., index, coordinates)
- Individuals may require additional resources to store/generate the passwords
  - Sheet of paper, application, additional device, etc.

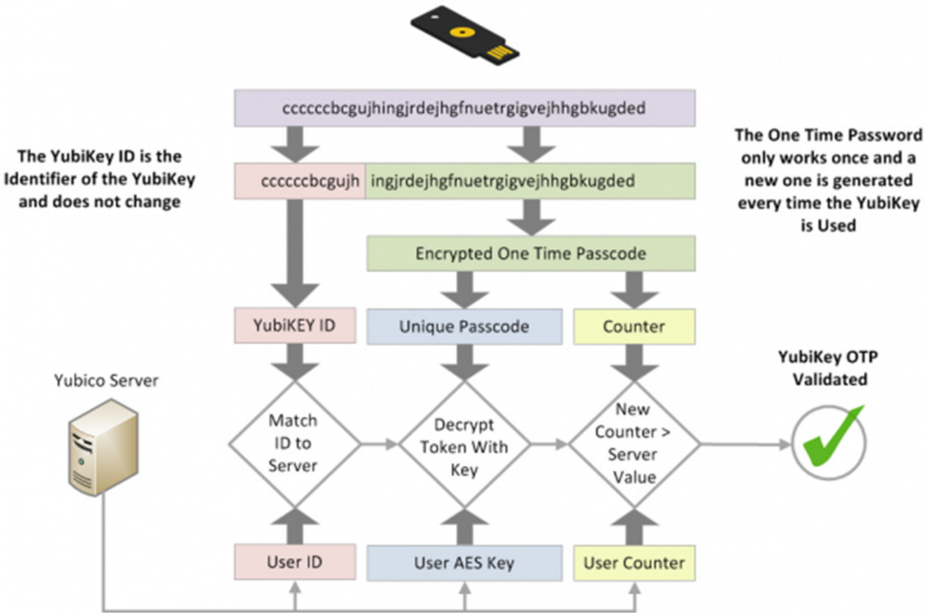
# Yubikey

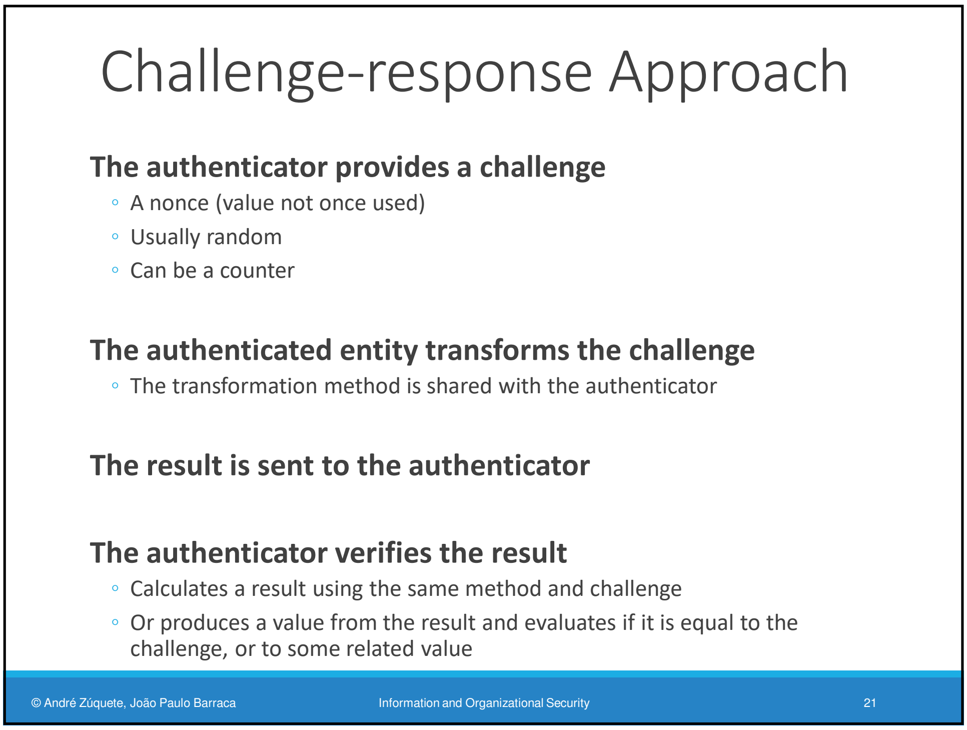
## Personal Authentication Device

- USB, Bluetooth and/or NFC

## Activation generates a 44 characters key

- Emulates a USB keyboard (besides an own API)
- Supports HOTP (events) or TOPT (Temporal)
- If a challenges is provided, user most touch the button to obtain a result
- Several algorithms, including AES 256


cccccbcgujhingjrdejhgfnuetrgigvejhhgbkugded
The YubiKey ID is the Identifier of the YubiKey and does not change
cccccbcgujh
ingjrdejhgfnuetrgigvejhhgbkugded
The One Time Password only works once and a new one is generated every time the YubiKey is Used
Encrypted One Time Passcode
YubiKEY ID
Unique Passcode
Counter
YubiKey OTP Validated
Yubico Server
Match ID to Server
Decrypt Token With Key
New Counter > Server Value
User ID
User AES Key
User Counter



# Challenge-response Approach

**The authenticator provides a challenge**

- A nonce (value not once used)
- Usually random
- Can be a counter

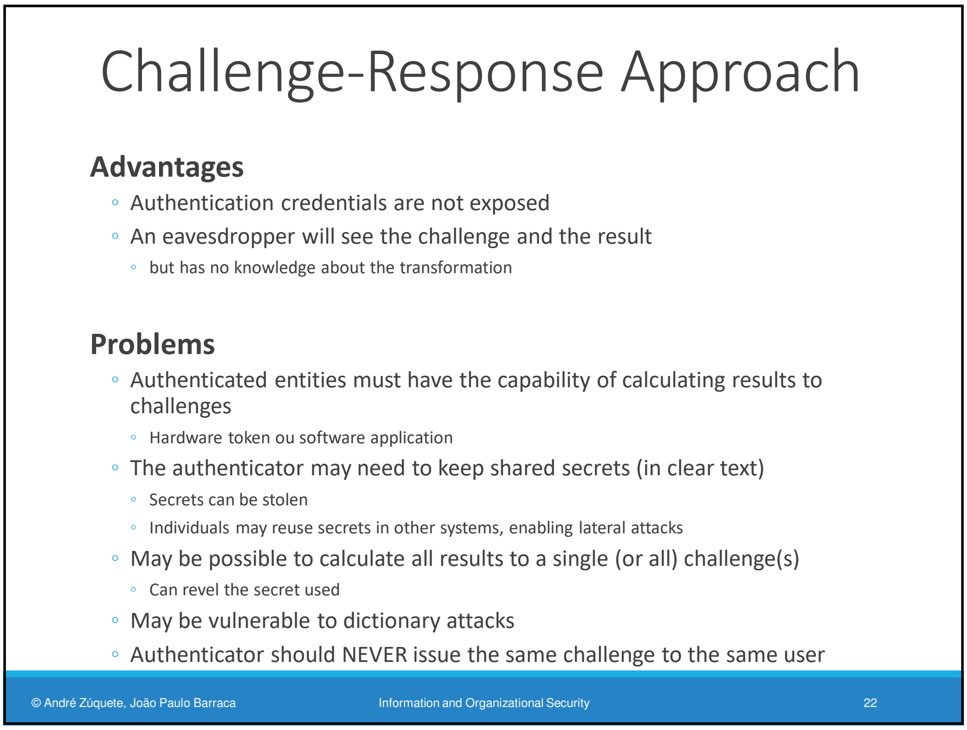
**The authenticated entity transforms the challenge**

- The transformation method is shared with the authenticator

**The result is sent to the authenticator**

**The authenticator verifies the result**

- Calculates a result using the same method and challenge
- Or produces a value from the result and evaluates if it is equal to the challenge, or to some related value



# Challenge-Response Approach

**Advantages**

- Authentication credentials are not exposed
- An eavesdropper will see the challenge and the result
  - but has no knowledge about the transformation

**Problems**

- Authenticated entities must have the capability of calculating results to challenges
  - Hardware token ou software application
- The authenticator may need to keep shared secrets (in clear text)
  - Secrets can be stolen
  - Individuals may reuse secrets in other systems, enabling lateral attacks
- May be possible to calculate all results to a single (or all) challenge(s)
  - Can revel the secret used
- May be vulnerable to dictionary attacks
- Authenticator should NEVER issue the same challenge to the same user

# Authn of Subjects: Challenge-Response with Smartcards

## Authentication Credentials

- Having the smartcard
  - e.g., the Citizen Card
- The private key stored inside the smartcard
- The PIN code to access the key

## The authenticator knows

- The user public key

## Robust against:

- Dictionary attacks
- Offline attacks to the database
- Insecure channels

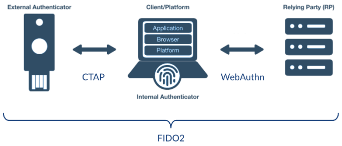
PORTUGAL
SPECIMEN
I<PRT00000000<<0<ZZ4<<<<<<<<<<2
8108108F1201158PRT<<<<<<<<<<<1
AVILA<<PAULA<ANDREIA<CONCEICAO

# Authn of Subjects: Challenge-Response with Smartcards

## Challenge-Response Protocol

- The authenticator generates a challenge
- Smartcard owner ciphers the challenge with their private key
  - Stored in the smartcard, protected by the PIN code
  - In alternative, can sign the challenge
- The authenticator deciphers the result with the public key
  - If the decrypted result matches the challenge, the authentication is successful
  - In alternative, it can verify the signature (which is the same process)

# Authn of Subjects: Challenge-Response with other tokens

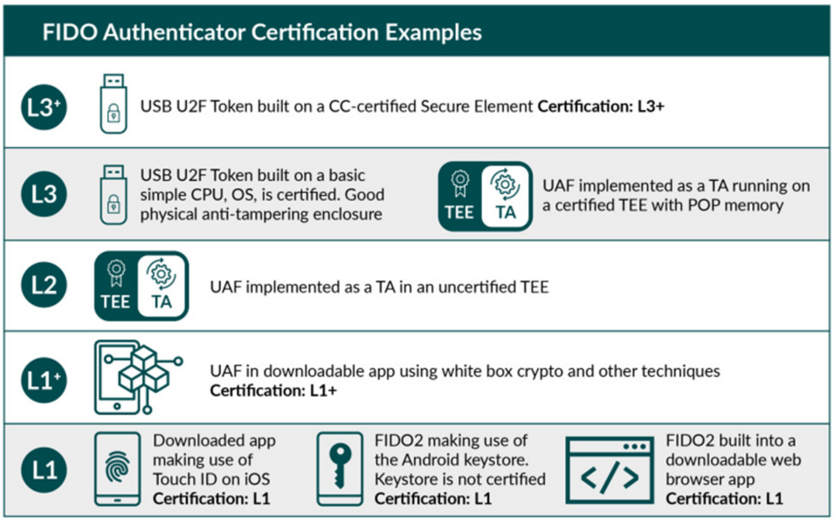


## FIDO2 tokens (FIDO Alliance)

- For both mobile and desktop environments
- Web Authentication (WebAuthn) specification
- Client-to-Authenticator Protocol (CTAP)
- Security
  - Credentials never leave the user's device and are never stored on a server
  - No risks of phishing, no password theft (still, tokens can be stolen)
  - No replay attacks
  - Token certification levels
- Privacy
  - Credentials are unique per website
  - Tracking is not possible (different web sites, different public keys for the same token)
  - Biometric data, when used, never leaves the user's device

https://www.inovex.de/de/blog/fido2-webauthn-in-practice/

# FIDO2 certification



**FIDO Authenticator Certification Examples**

| | | | |
|---|---|---|---|
| L3+ | USB U2F Token built on a CC-certified Secure Element **Certification: L3+** | | |
| L3 | USB U2F Token built on a basic simple CPU, OS, is certified. Good physical anti-tampering enclosure | TEE TA UAF implemented as a TA running on a certified TEE with POP memory | |
| L2 | TEE TA UAF implemented as a TA in an uncertified TEE | | |
| L1+ | UAF in downloadable app using white box crypto and other techniques **Certification: L1+** | | |
| L1 | Downloaded app making use of Touch ID on iOS **Certification: L1** | FIDO2 making use of the Android keystore. Keystore is not certified **Certification: L1** | FIDO2 built into a downloadable web browser app **Certification: L1** |

# Authn of Subjects: Challenge-Response with Shared Secret

## Authentication Credentials

- Password selected by the individual

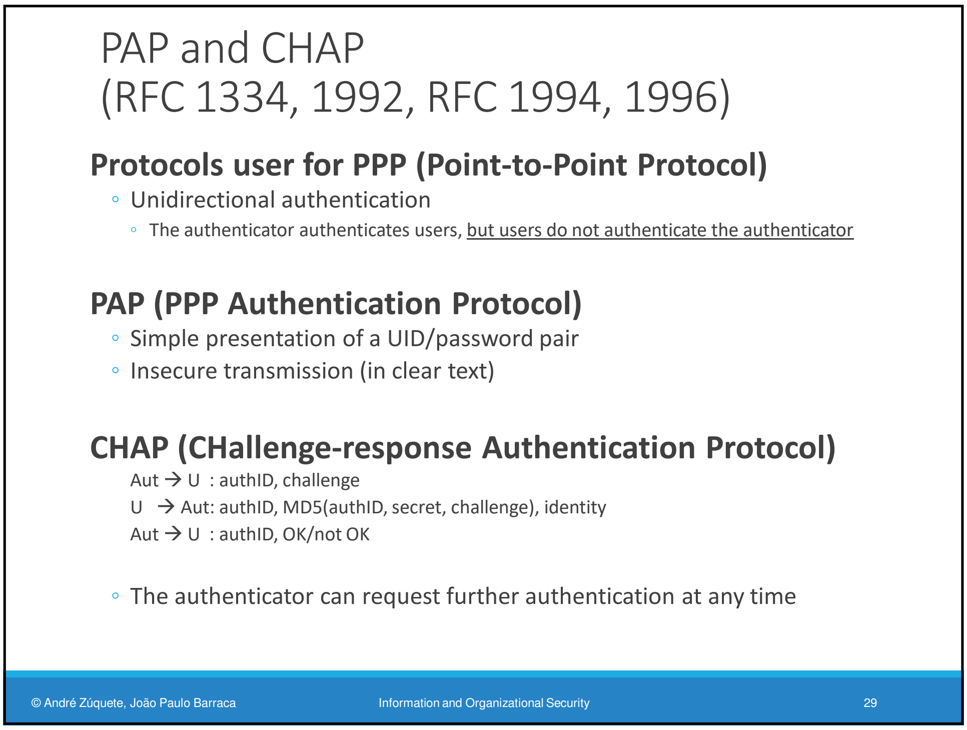
## The authenticator knows:

- Bad approach: the shared password
- Better approach: A transformation of the shared password
  - The transformation should be unidirectional

# Authentication of Subjects: Challenge-Response with Shared Secret

## Basic Challenge-Response Protocol

- The authenticator generates a challenge
- The individual calculates a transformation of the challenge and the password
  - result = hash(challenge || password)
  - or... result = encrypt(challenge, password)
- The authenticator reverts the process and checks if the values match
  - result == hash( challenge || password)
  - or .... challenge == decrypt(result, password)
- Examples with shared passwords: CHAP, MS-CHAP v1/v2, S/Key
- Examples with shared keys: SIM & USIM (celular communications)



# PAP and CHAP
# (RFC 1334, 1992, RFC 1994, 1996)

## Protocols user for PPP (Point-to-Point Protocol)

- Unidirectional authentication
  - The authenticator authenticates users, but users do not authenticate the authenticator

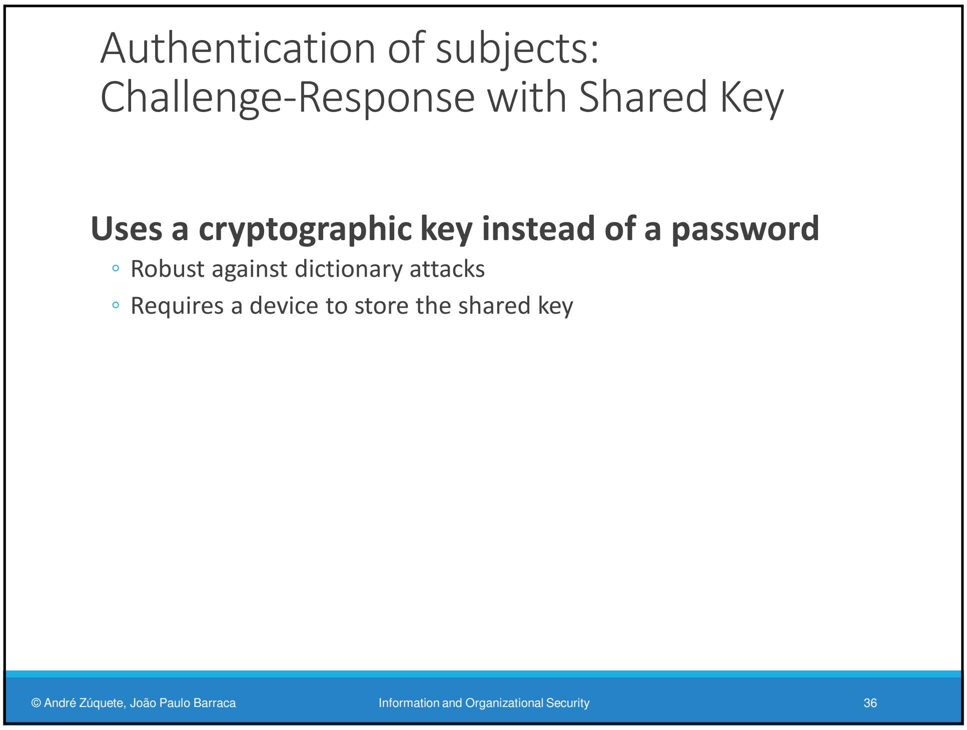
## PAP (PPP Authentication Protocol)

- Simple presentation of a UID/password pair
- Insecure transmission (in clear text)

## CHAP (CHallenge-response Authentication Protocol)

Aut → U : authID, challenge
U → Aut: authID, MD5(authID, secret, challenge), identity
Aut → U : authID, OK/not OK

- The authenticator can request further authentication at any time



# Authentication of subjects:
# Challenge-Response with Shared Key

## Uses a cryptographic key instead of a password

- Robust against dictionary attacks
- Requires a device to store the shared key

# GSM Subscriber authentication

**Uses a secret shared between the HLR and the subscriber phone**

- Uses 128-bit shared key (not an asymmetric key pair)
- Key is stored in the SIM card
- SIM card is unlocked by a user PIN
- SIM card answers challenges using the shared key

**Uses (initially unknown algorithms):**

- A3 for authentication
- A8 to generate the session key
- A5 is a stream cipher for communication

**A3 and A8 executed by the SIM, A5 executed by the baseband**

- A3 and A8 can be chosen by the operator

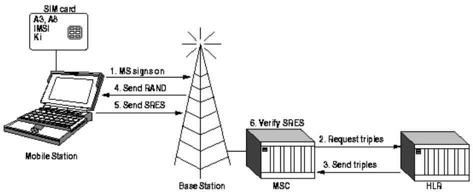
# GSM Subscriber authentication

**MSC requests triples from HLR/AUC**

- RAND, SRES, Kc
- It can ask one or several

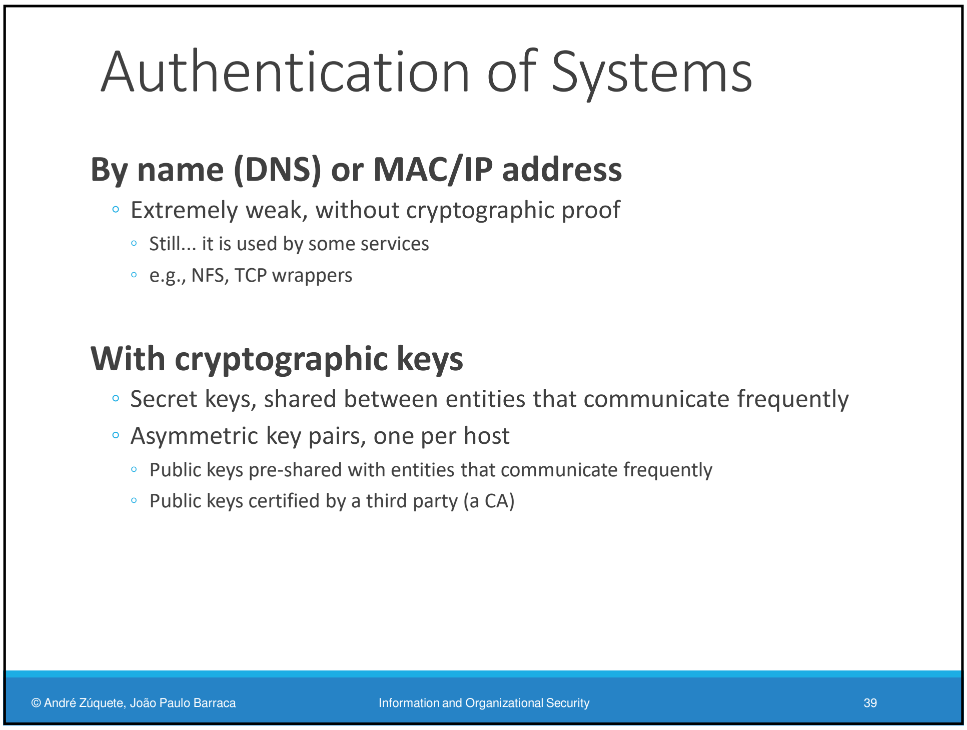
**HLR generates RAND and the triples using the subscriber Ki**

- RAND, random value (128 bits)
- SRES = A3 (Ki, RAND) (32 bits)
- Kc = A8 (Ki, RAND) (64 bits)



**Frequently uses COMP128 for the A3/A8 algorithms**

- Recommended by the GSM consortium
- [SRES, Kc] = COMP128 (Ki, RAND)



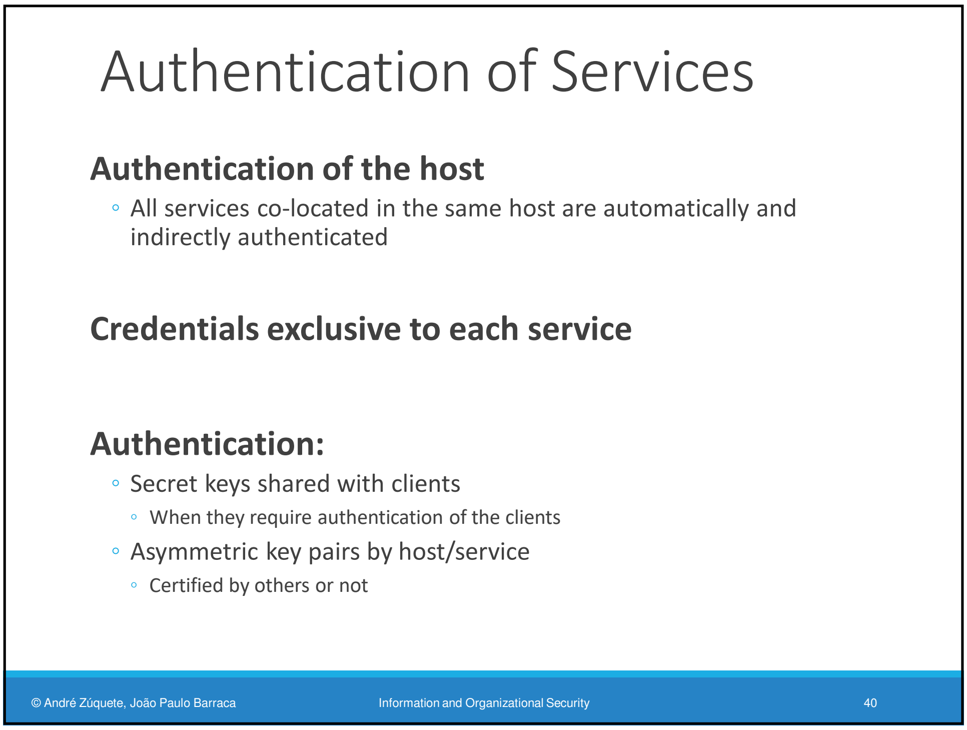
# Authentication of Systems

## By name (DNS) or MAC/IP address

- Extremely weak, without cryptographic proof
  - Still... it is used by some services
  - e.g., NFS, TCP wrappers

## With cryptographic keys

- Secret keys, shared between entities that communicate frequently
- Asymmetric key pairs, one per host
  - Public keys pre-shared with entities that communicate frequently
  - Public keys certified by a third party (a CA)



# Authentication of Services

## Authentication of the host

- All services co-located in the same host are automatically and indirectly authenticated

## Credentials exclusive to each service

## Authentication:

- Secret keys shared with clients
  - When they require authentication of the clients
- Asymmetric key pairs by host/service
  - Certified by others or not

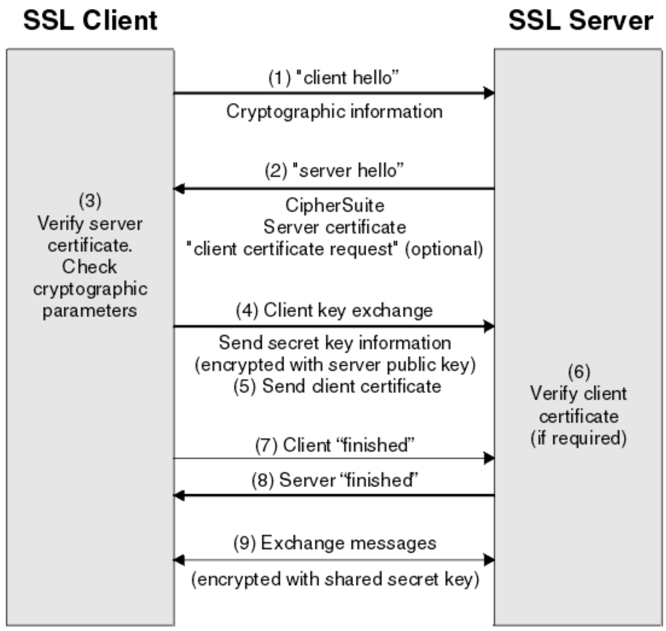
# TLS (Transport Layer Security, RFC 2246)

## Secure Communication Protocol over TCP/IP

- Evolved from the SSL V3 (Secure Sockets Layer) standard
- Manages secure sessions over TCP/IP, individual to each application
  - Initially designed for HTTP traffic
  - Currently used for many other types of traffic

## Security mechanisms

- Confidentiality and integrity of the communication between entities
  - Key distribution, negotiation of ciphers, digests and other mechanisms
- Authentication of the intervenient entities
  - Servers, services, etc...
  - Clients (not so common)
  - Both executed with asymmetric keys and X.509 certificates



SSL Client

SSL Server

(1) "client hello"
Cryptographic information

(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)

(3)
Verify server certificate.
Check cryptographic parameters

(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate

(6)
Verify client certificate
(if required)

(7) Client "finished"

(8) Server "finished"

(9) Exchange messages
(encrypted with shared secret key)

# TLS Ciphersuites

**If a server supports a single algorithm, it cannot expected for all clients to also support it**

- More powerful/limited, older/newer

**The Ciphersuite concept allows the negotiation of mechanisms between client and server**

- Both send their supported ciphersuites, and select one they both share
- The server choses

**Exemplo: ECDHE-RSA-AES128-GCM-SHA256**

**Format:**

- Key negotiation algorithm: ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)
- Authentication algorithm: RSA
- Cipher algorithm and cipher mode: AES-128 GCM
- Integrity control algorithm: SHA256

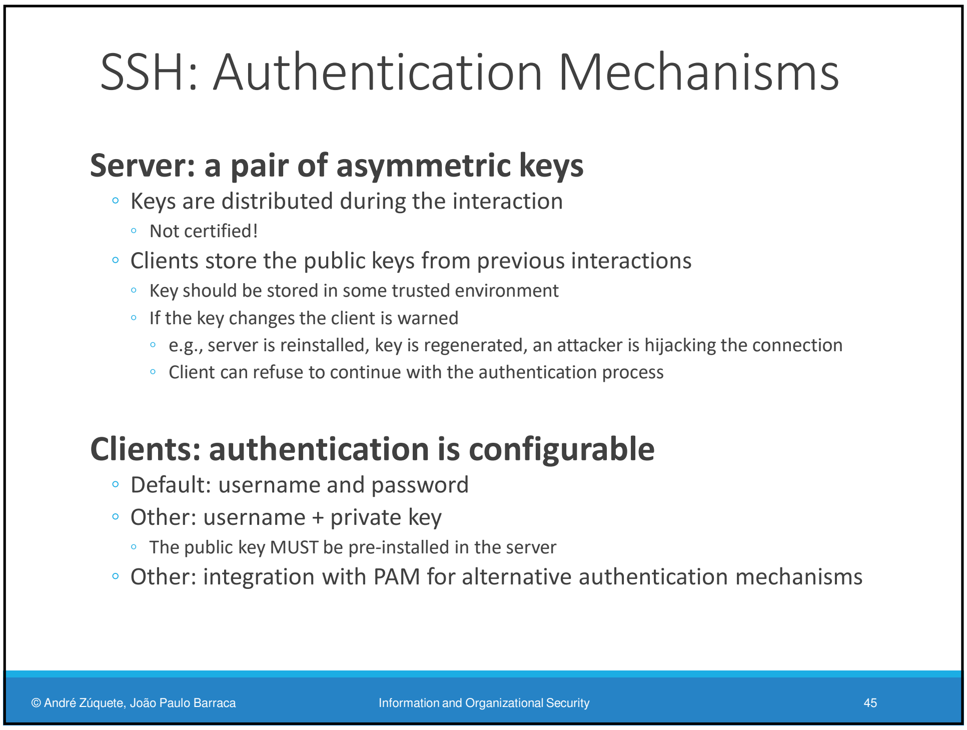
# SSH (Secure SHell)

**Manages secure console sessions over TCP/IP**

- Initially designed to replace the Telnet application/protocol
- Currently used in many other applications
  - Execution of remote commands in a secure manner (rsh/rexec)
  - Secure copy of contents from/to remote hosts (rcp)
  - Secure FTP (sftp)
  - Secure (Generic) communication tunnels (carry standard IP packets)

**Security Mechanisms**

- Confidentiality and integrity of the communications
  - Key distribution
- Authentication of the intervenient entities
  - Server / Hosts
  - Client users
  - Both achieved through several, and differentiated mechanisms



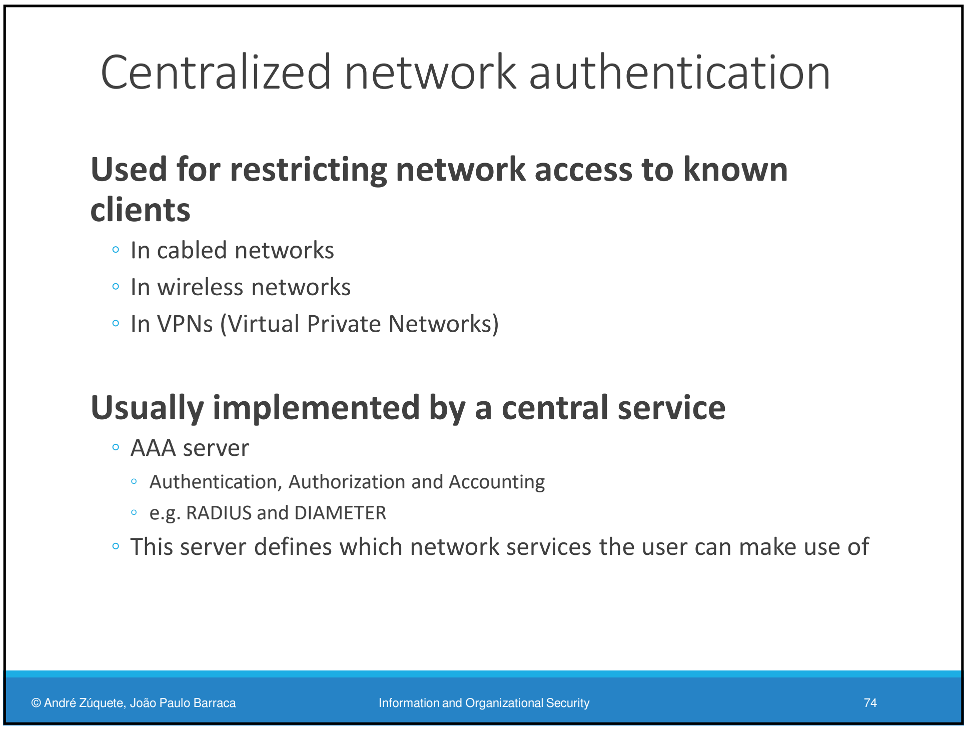
# SSH: Authentication Mechanisms

**Server: a pair of asymmetric keys**

- Keys are distributed during the interaction
  - Not certified!
- Clients store the public keys from previous interactions
  - Key should be stored in some trusted environment
  - If the key changes the client is warned
    - e.g., server is reinstalled, key is regenerated, an attacker is hijacking the connection
    - Client can refuse to continue with the authentication process

**Clients: authentication is configurable**

- Default: username and password
- Other: username + private key
  - The public key MUST be pre-installed in the server
- Other: integration with PAM for alternative authentication mechanisms



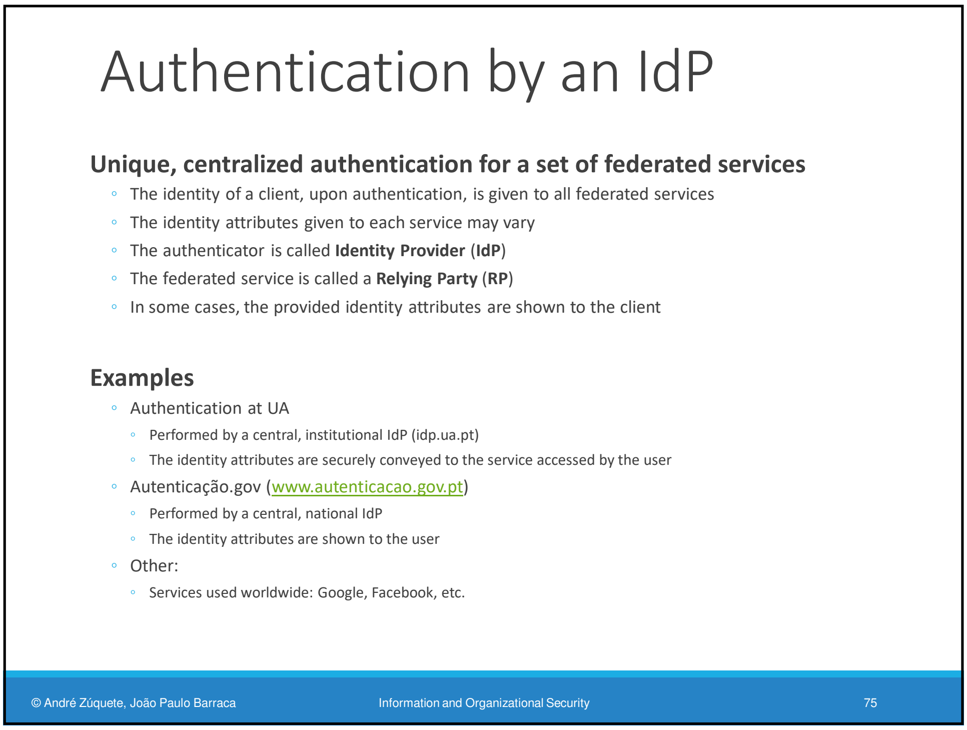
# Centralized network authentication

**Used for restricting network access to known clients**

- In cabled networks
- In wireless networks
- In VPNs (Virtual Private Networks)

**Usually implemented by a central service**

- AAA server
  - Authentication, Authorization and Accounting
  - e.g. RADIUS and DIAMETER
- This server defines which network services the user can make use of



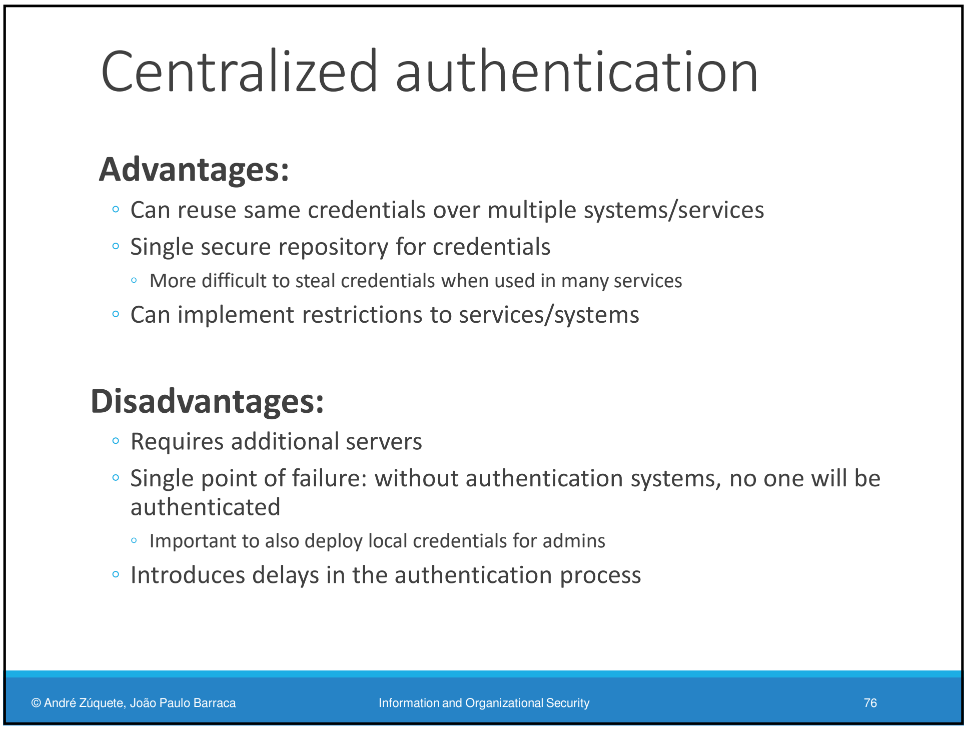
# Authentication by an IdP

### Unique, centralized authentication for a set of federated services

- The identity of a client, upon authentication, is given to all federated services
- The identity attributes given to each service may vary
- The authenticator is called **Identity Provider** (**IdP**)
- The federated service is called a **Relying Party** (**RP**)
- In some cases, the provided identity attributes are shown to the client

### Examples

- Authentication at UA
  - Performed by a central, institutional IdP (idp.ua.pt)
  - The identity attributes are securely conveyed to the service accessed by the user
- Autenticação.gov (www.autenticacao.gov.pt)
  - Performed by a central, national IdP
  - The identity attributes are shown to the user
- Other:
  - Services used worldwide: Google, Facebook, etc.



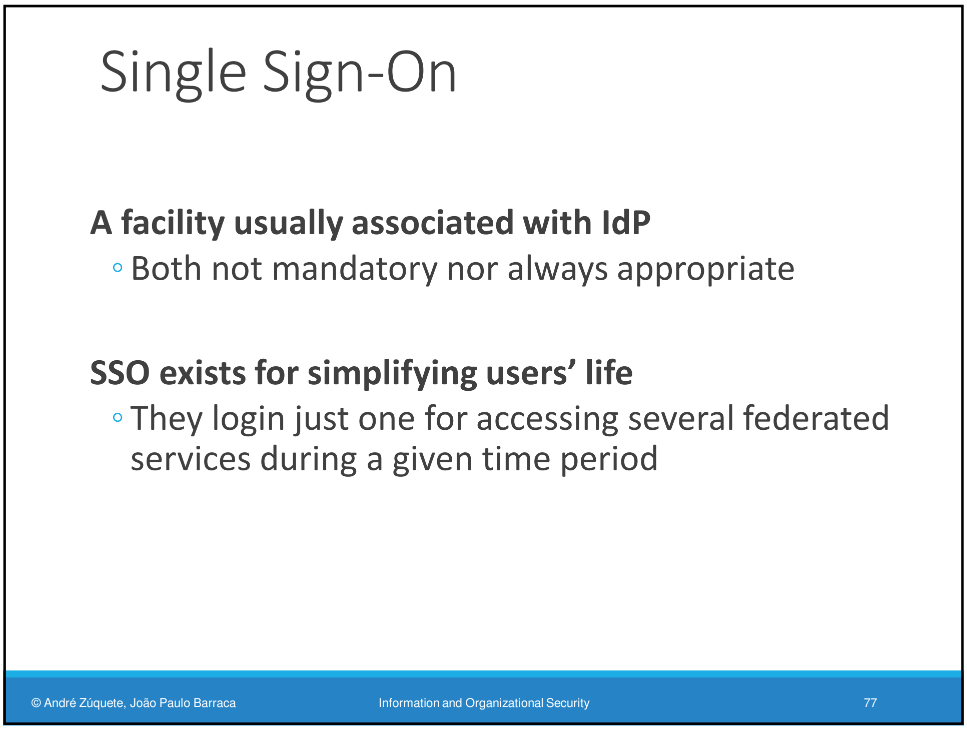
# Centralized authentication

## Advantages:

- Can reuse same credentials over multiple systems/services
- Single secure repository for credentials
  - More difficult to steal credentials when used in many services
- Can implement restrictions to services/systems

## Disadvantages:

- Requires additional servers
- Single point of failure: without authentication systems, no one will be authenticated
  - Important to also deploy local credentials for admins
- Introduces delays in the authentication process



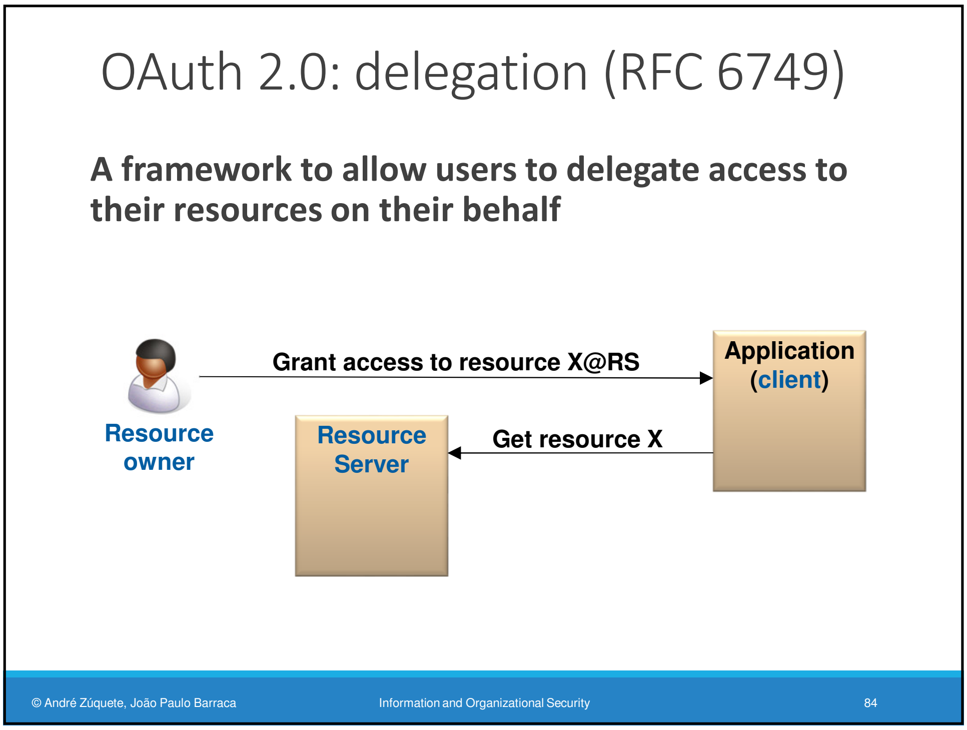
# Single Sign-On

**A facility usually associated with IdP**

- Both not mandatory nor always appropriate

**SSO exists for simplifying users' life**

- They login just one for accessing several federated services during a given time period



# OAuth 2.0: delegation (RFC 6749)

**A framework to allow users to delegate access to their resources on their behalf**

# OAuth 2.0 roles

### Resource owner

- An entity capable of granting access to a **protected resource**
- **End-user**: a resource owner that is a person

### Client

- An **application** making requests for protected resources on behalf of the resource owner and with its authorization

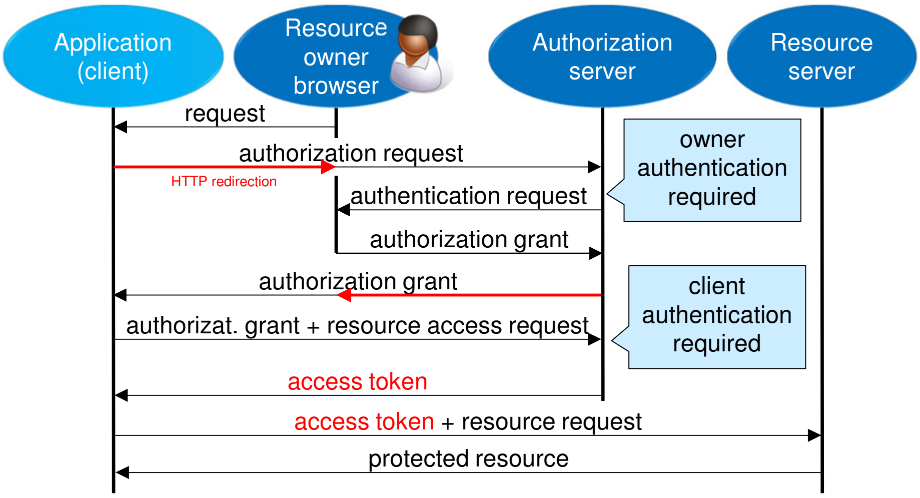
### Resource Server

- The server hosting protected resources
- Responds to protected resource requests using **access tokens**

### Authorization Server

- The server issuing access tokens to clients after successfully **authenticating resource owners** and obtaining their **authorization** for the clients to access one of their (users) resources

# Protocol flow



Application (client)

Resource owner browser

Authorization server

Resource server

request

authorization request

HTTP redirection

authentication request

owner authentication required

authorization grant

authorization grant

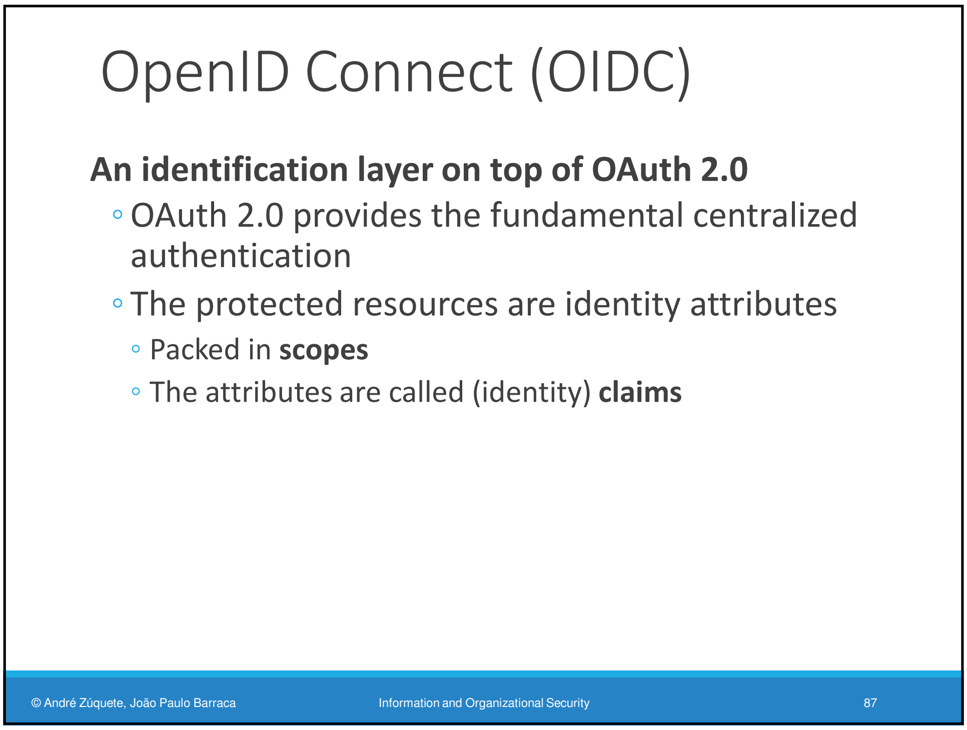
client authentication required

authorizat. grant + resource access request

access token

access token + resource request

protected resource



# OpenID Connect (OIDC)

**An identification layer on top of OAuth 2.0**

- OAuth 2.0 provides the fundamental centralized authentication
- The protected resources are identity attributes
  - Packed in **scopes**
  - The attributes are called (identity) **claims**

# Reliable storage

## Problems

### Storage devices develop faults

- It should be minimized the failures in storage devices and loss of data
- Failure is certain and cannot be ignored

### Access to mechanical disks is slow (hard disks)

- Access Time = Translation time + Rotation Time
- More information → higher impact of storage media

# Problems

**Solid State Devices (SSDs) have a limited number of write operations**

- 2000-3000 writes per sector for MLC (2 bits per cell)

**Specific events may result in total data loss**

- Fire, robbery, “energy peaks”, floods, user mistakes, attacks

**May be required to distribute data in an intelligent manner**

- To maximize performance
- To reduce costs

# Solutions

**Data backups**

- Local
- Remote

**Redundant Storage**

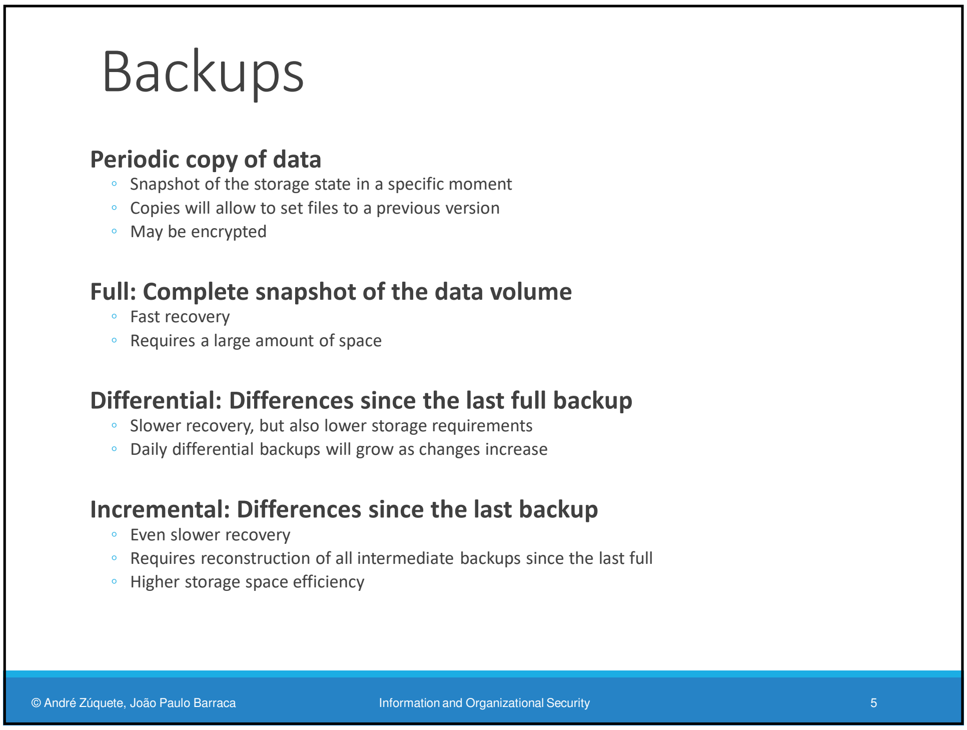
- RAID
- Other: ZFS

**Better storage devices, environments with higher control**

- SLED (Single Large Expensive Disks)
- Enterprise Grade devices
- Temperature and Humidity Control

**Infrastructures dedicated for storage**

- Single policy control point



# Backups

**Periodic copy of data**

- Snapshot of the storage state in a specific moment
- Copies will allow to set files to a previous version
- May be encrypted

**Full: Complete snapshot of the data volume**

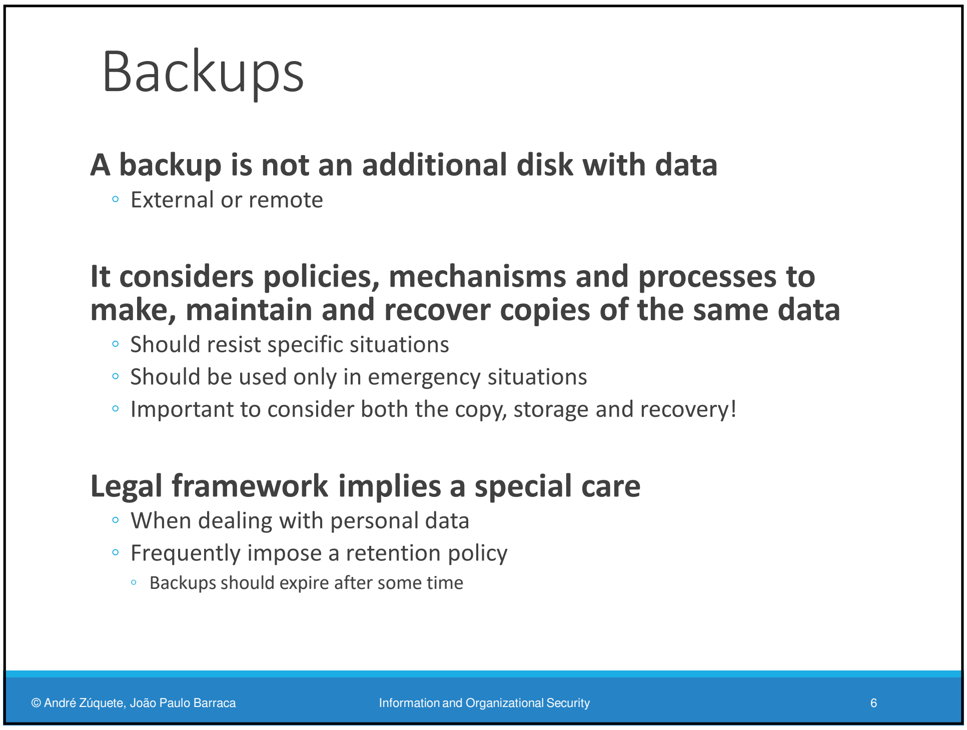
- Fast recovery
- Requires a large amount of space

**Differential: Differences since the last full backup**

- Slower recovery, but also lower storage requirements
- Daily differential backups will grow as changes increase

**Incremental: Differences since the last backup**

- Even slower recovery
- Requires reconstruction of all intermediate backups since the last full
- Higher storage space efficiency



# Backups

**A backup is not an additional disk with data**

- External or remote

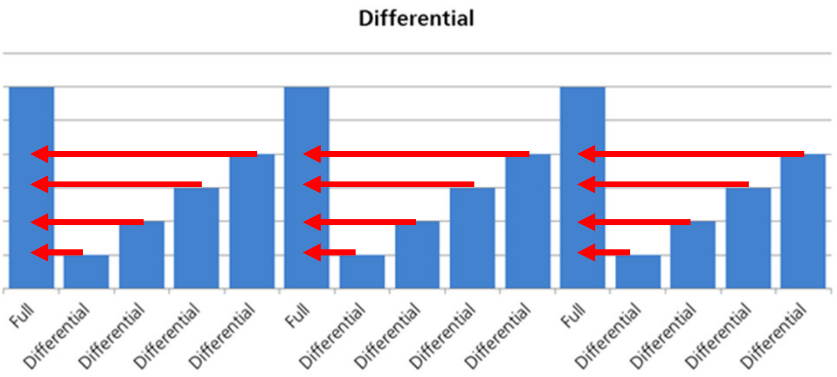
**It considers policies, mechanisms and processes to make, maintain and recover copies of the same data**

- Should resist specific situations
- Should be used only in emergency situations
- Important to consider both the copy, storage and recovery!

**Legal framework implies a special care**

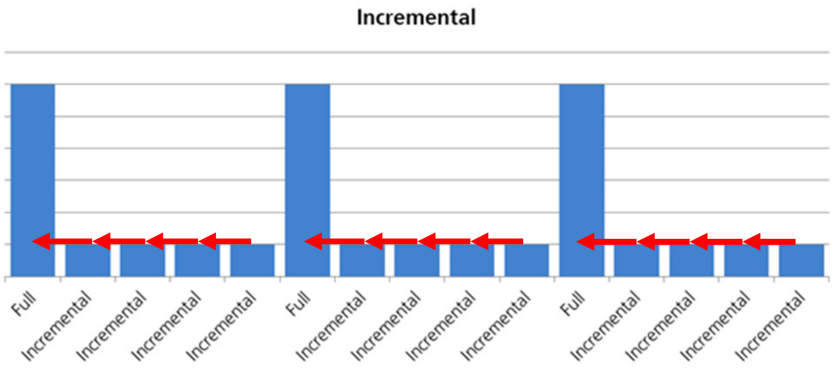
- When dealing with personal data
- Frequently impose a retention policy
  - Backups should expire after some time

# Backups types: Differential

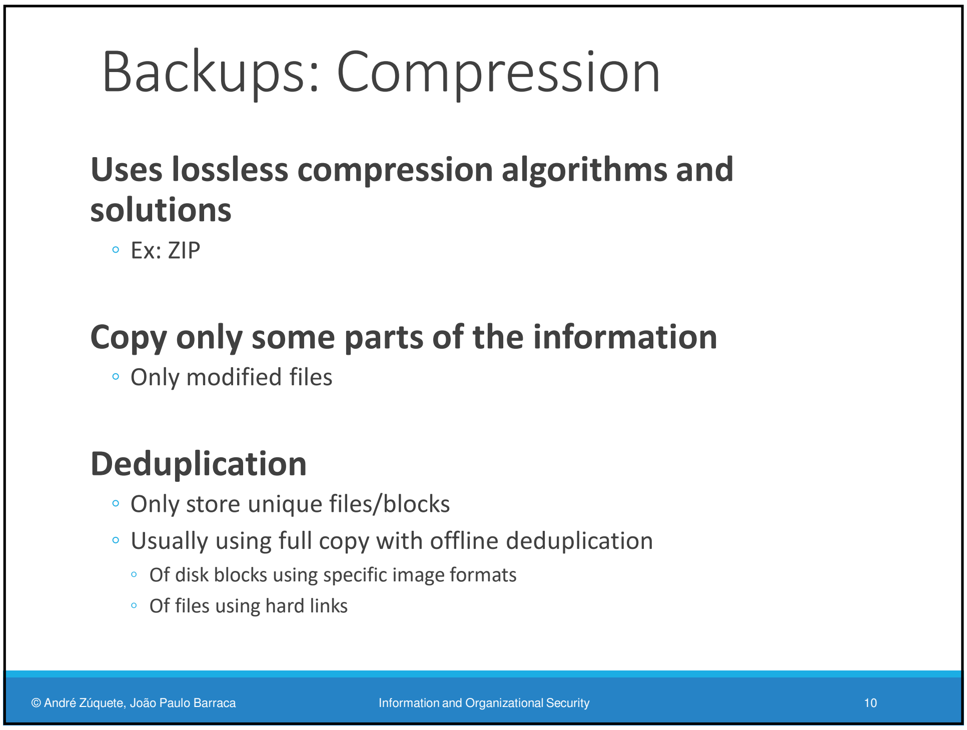


http://www.teammead.co.uk/

# Backups types: Incremental



http://www.teammead.co.uk/



# Backups: Compression

## Uses lossless compression algorithms and solutions

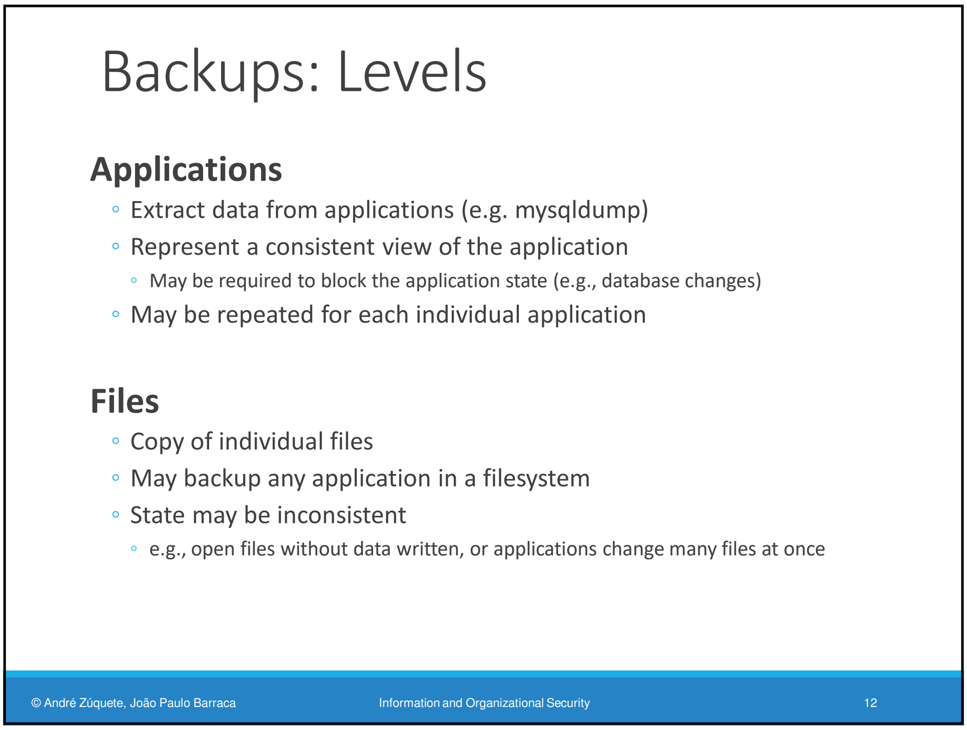
- Ex: ZIP

## Copy only some parts of the information

- Only modified files

## Deduplication

- Only store unique files/blocks
- Usually using full copy with offline deduplication
  - Of disk blocks using specific image formats
  - Of files using hard links



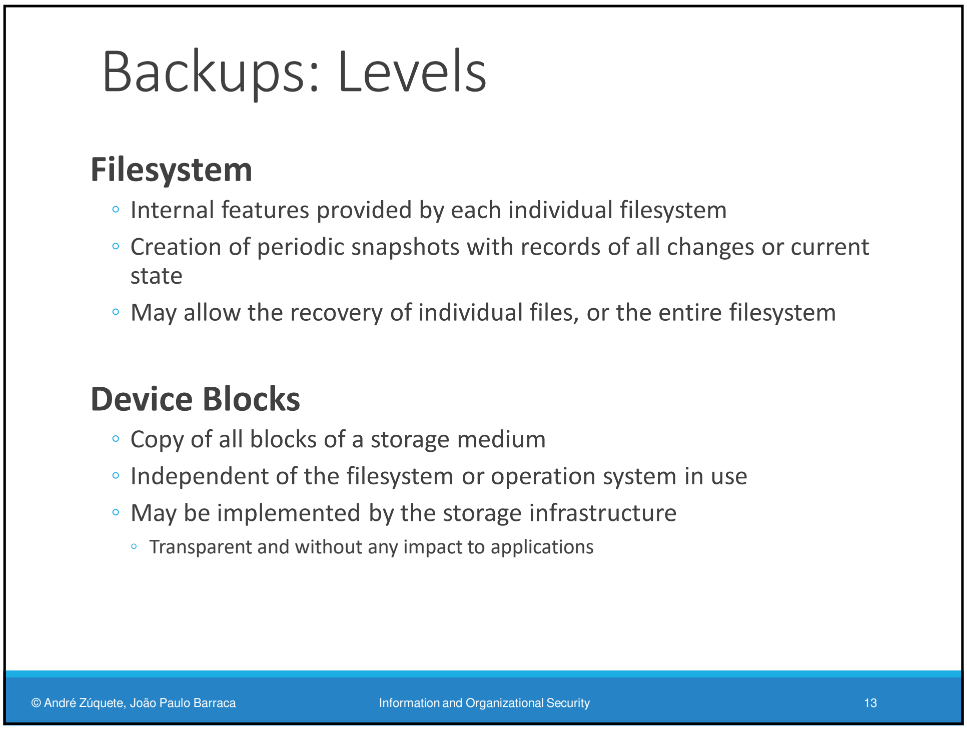
# Backups: Levels

## Applications

- Extract data from applications (e.g. mysqldump)
- Represent a consistent view of the application
  - May be required to block the application state (e.g., database changes)
- May be repeated for each individual application

## Files

- Copy of individual files
- May backup any application in a filesystem
- State may be inconsistent
  - e.g., open files without data written, or applications change many files at once



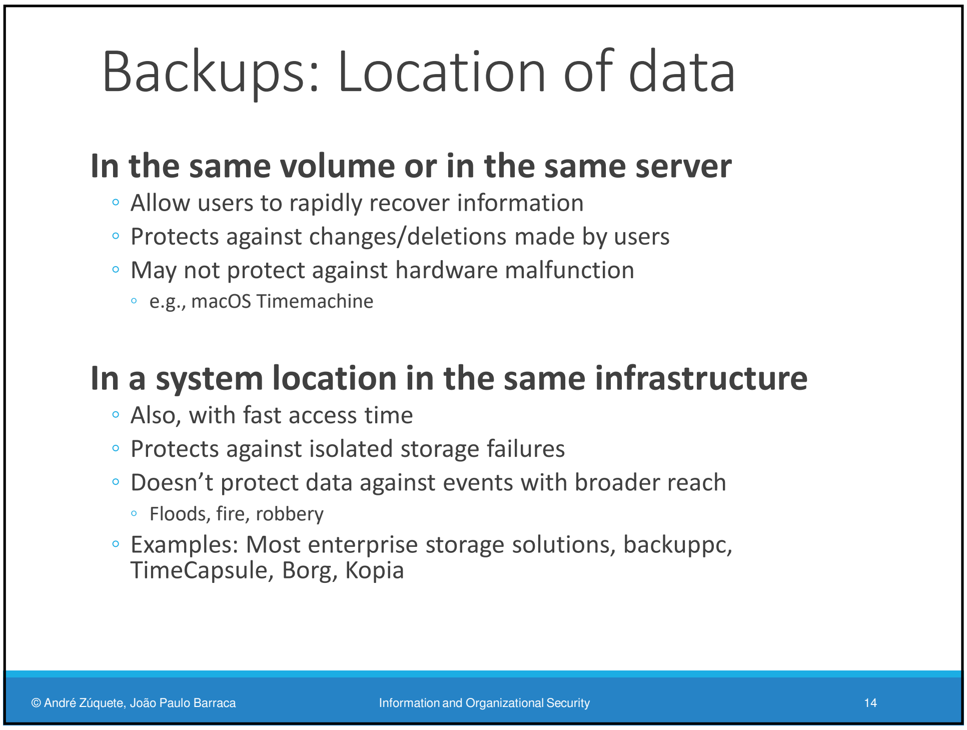
# Backups: Levels

## Filesystem

- Internal features provided by each individual filesystem
- Creation of periodic snapshots with records of all changes or current state
- May allow the recovery of individual files, or the entire filesystem

## Device Blocks

- Copy of all blocks of a storage medium
- Independent of the filesystem or operation system in use
- May be implemented by the storage infrastructure
  - Transparent and without any impact to applications



# Backups: Location of data

## In the same volume or in the same server

- Allow users to rapidly recover information
- Protects against changes/deletions made by users
- May not protect against hardware malfunction
  - e.g., macOS Timemachine

## In a system location in the same infrastructure

- Also, with fast access time
- Protects against isolated storage failures
- Doesn’t protect data against events with broader reach
  - Floods, fire, robbery
- Examples: Most enterprise storage solutions, backuppc, TimeCapsule, Borg, Kopia

# Backups: Location of data

## Remote (off-site)

- Implemented to a system outside the local datacenter
  - Dedicated service or through the internet
    - e.g., Amazon S3, or to servers in a dedicated datacenter
    - Encryption if recommended (or mandatory) in the case of external services!
- Implemented with specialized secure transport
  - Armored car transporting backups to a secure place
- Allow recovery even if far reaching events occur
  - Terrorism, Earthquake
- Recovery will be slower
  - Limited by the speed of a network link or the physical transport

# Selecting Storage Devices

## Different device grades: Enterprise vs Desktop

- Different construction quality and recovery features
- Different MTBF: Mean Time Between Failures
  - Enterprise HDD: 1.2M hours, at 45°C, working 24/7, 100% use rate (1)
  - Desktop HDD: 700K hours, at 25°C, working 8/5, 10-20% use rate(1)

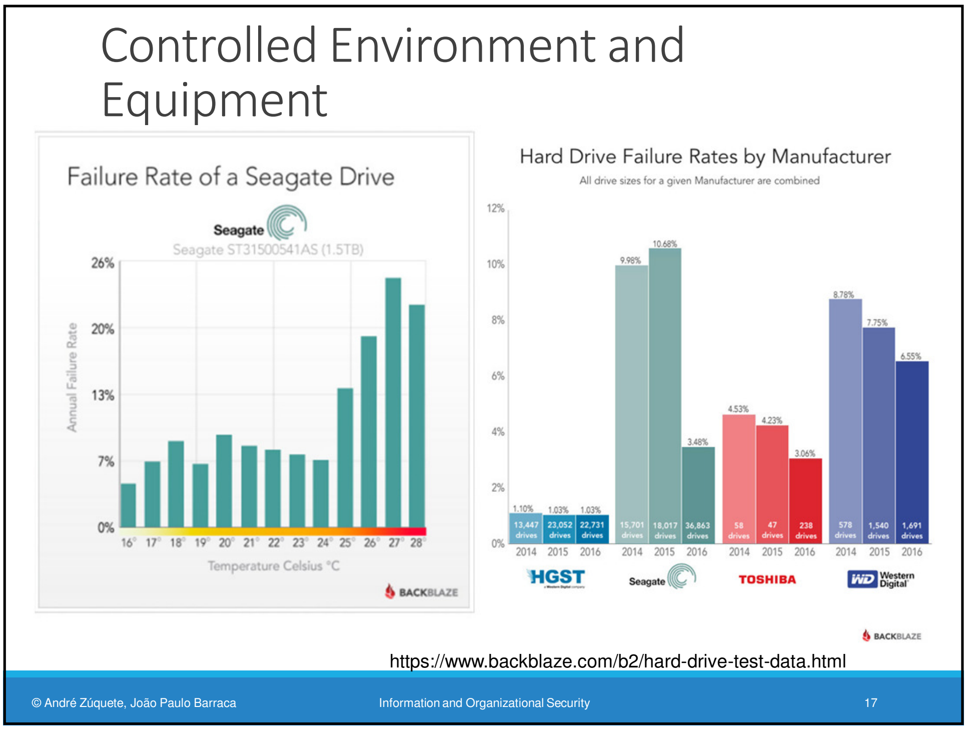
## Adjusted to each use case

- Write intensive vs Read Intensive
- NAS vs Video vs Desktop vs Cold Storage vs Data Center
  - Differences in power consumption, reliability and performance

## Adjusted to a specific performance level

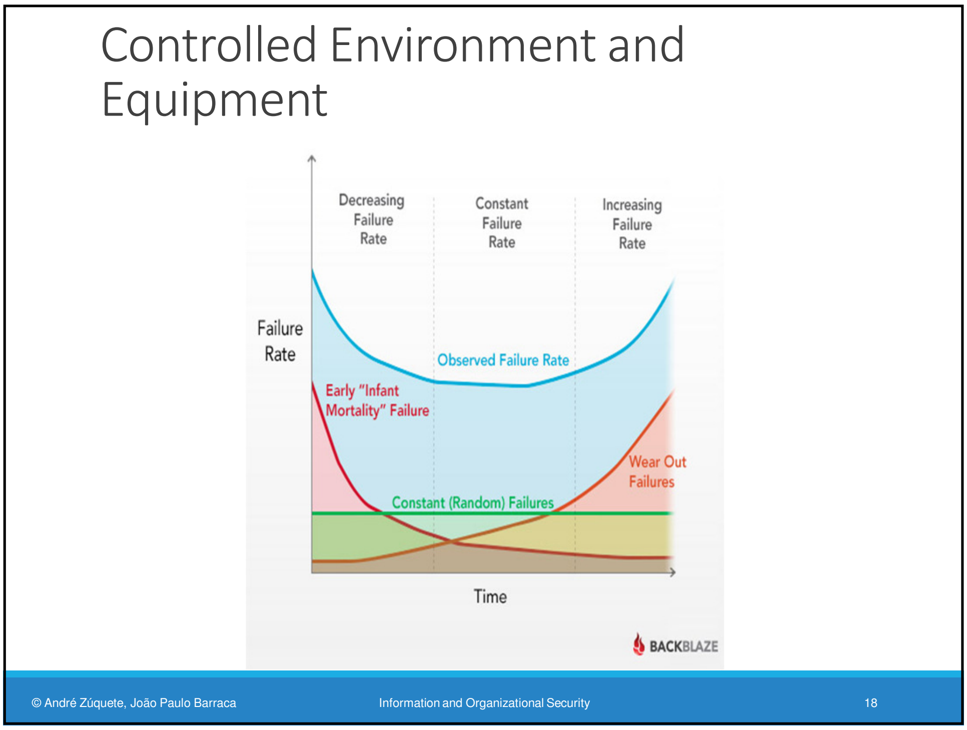
- Tier 0: Highest performance, low capacity (PCIe NVME SLC SSD)
- Tier 1: Some performance, high capacity and availability (M2 SATA SSD)
- Tier 3: Low performance, high capacity, low price (SATA HDD)

1) Enterprise-class versus Desktop-class Hard Drives, rev 1.0, Intel, 2008



# Controlled Environment and Equipment

https://www.backblaze.com/b2/hard-drive-test-data.html



# Controlled Environment and Equipment

# RAID: Redundant Array of Inexpensive Drives

### Improves the survivability of information

- Data is only lost after several devices are lost
- The number of lost devices is configurable

### Low cost and efficient solution

- Can use cheap, lower quality hardware
- Can improve read and write performance

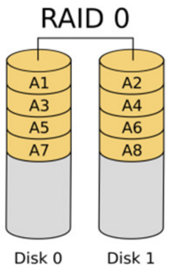
### RAID doesn’t replace backups

- Only tolerates the failure of a limited number of devices
- Cannot cope with user mistakes (file modification/deletion)

### RAID can even increase the failure probability

- As it can be tweaked towards performance

# RAID 0 (Striping)



### Objectives

- Speedup data access

### Approach

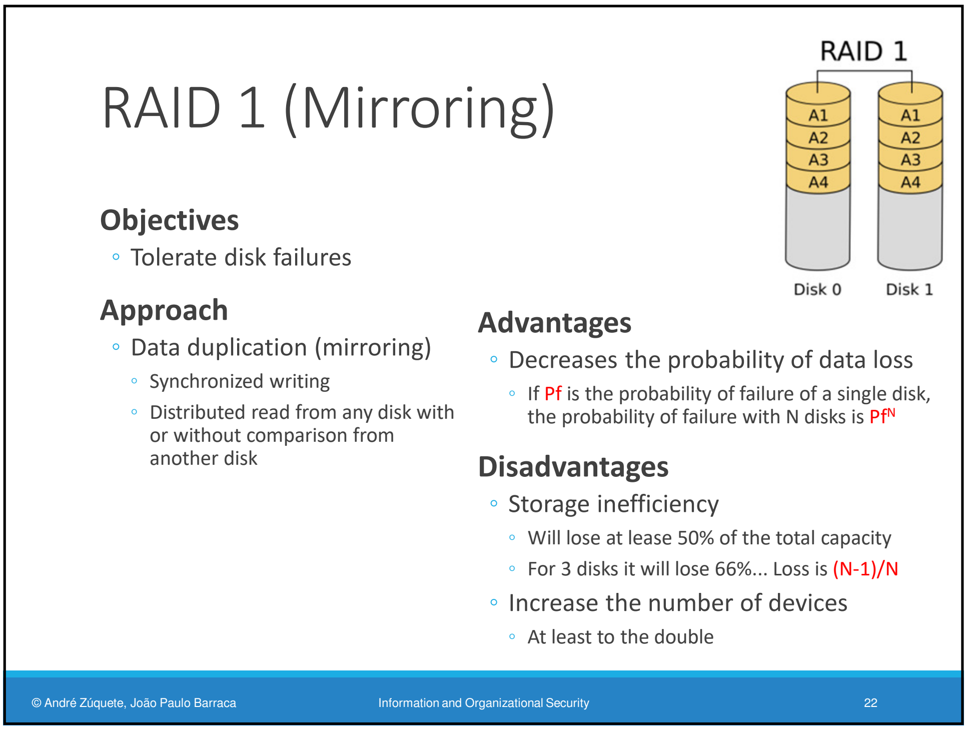
- Access disks in parallel
- Striping
  - Data is split in small chunks (stripes)
  - Stripes are stored among all disks in a distributed manner

### Advantages

- May speedup performance as a factor of the number of disks

### Disadvantages

- Increases the probability of loosing data
  - If Pf is the probability of failure of a single disk, an N-disk RAID 0 volume will have a $1-(1-Pf)^N$ failure probability
- Increases the number of devices
  - At least it will double the number



# RAID 1 (Mirroring)

## Objectives

- Tolerate disk failures

## Approach

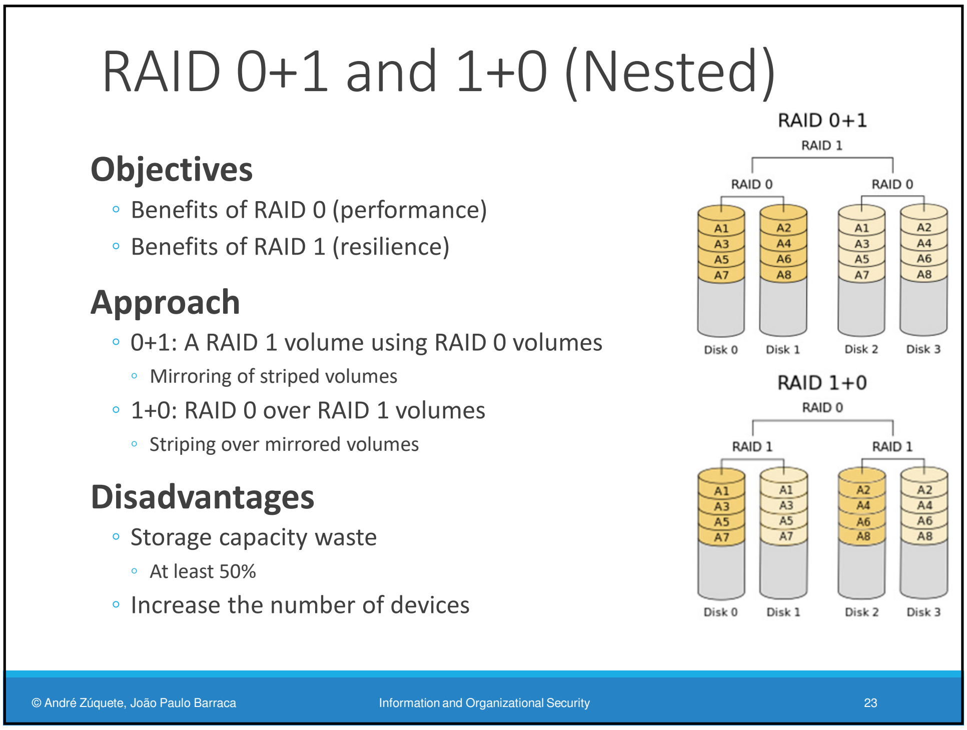
- Data duplication (mirroring)
  - Synchronized writing
  - Distributed read from any disk with or without comparison from another disk

## Advantages

- Decreases the probability of data loss
  - If Pf is the probability of failure of a single disk, the probability of failure with N disks is $Pf^N$

## Disadvantages

- Storage inefficiency
  - Will lose at lease 50% of the total capacity
  - For 3 disks it will lose 66%... Loss is (N-1)/N
- Increase the number of devices
  - At least to the double



# RAID 0+1 and 1+0 (Nested)

## Objectives

- Benefits of RAID 0 (performance)
- Benefits of RAID 1 (resilience)

## Approach

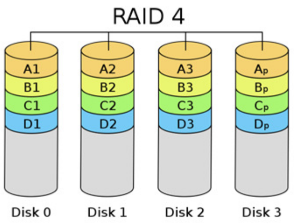
- 0+1: A RAID 1 volume using RAID 0 volumes
  - Mirroring of striped volumes
- 1+0: RAID 0 over RAID 1 volumes
  - Striping over mirrored volumes

## Disadvantages

- Storage capacity waste
  - At least 50%
- Increase the number of devices

RAID 0+1
RAID 1
RAID 0
RAID 0
A1 A3 A5 A7
A2 A4 A6 A8
A1 A3 A5 A7
A2 A4 A6 A8
Disk 0
Disk 1
Disk 2
Disk 3
RAID 1+0
RAID 0
RAID 1
RAID 1
A1 A3 A5 A7
A1 A3 A5 A7
A2 A4 A6 A8
A2 A4 A6 A8
Disk 0
Disk 1
Disk 2
Disk 3

# RAID 4



## Objectives

- Have some resilience as RAID 1
- With a performance close to RAID 0

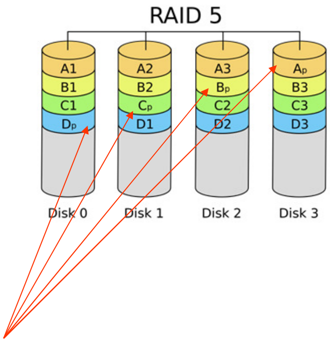
## Approach

- Store data in N-1 disks
- Store parity data in an additional disk
  - Total waste is dependent on the capacity and number of disks
  - Data from any N-1 disk can be used to recreate another one

## Disadvantages

- Requires at least 3 disks
  - Updating parity data is complex and will require specific hardware
  - Imposes the need to read before any write
    - Read data from existing block (e.g., C1) and from the corresponding parity disk (Cp)
    - Compare old data block with new, and change the parity block (Cp')
    - Write the new data block (C1') and the new parity block (Cp')
  - Writes must be serialized due to the existence of a parity disk
- Recovery is way more complex than with RAID 1

# RAID 5



## Objectives

- Similar to RAID 4
- But with higher write efficiency

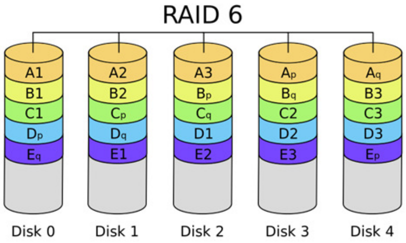
## Approach

- Distribute the parity blocks among all disks
- Waste is similar to RAID 4
- Write concurrency is improved

## Disadvantages

- More complex to be implemented

# RAID 6



## Objectives

- Improve the reliability of RAID 5

## Approach

- Use 2 parity blocks, distributed among all disks
- Capacity waste will be higher than in RAID 5 (equal to 2 disks)
- Concurrency is slightly worse than with RAID 5

## Advantages

- Allows the failure of two disks without data loss

## Disadvantages

- Even more complex than RAID 5

# NAS and SAN

## NAS: Network Attached Storage

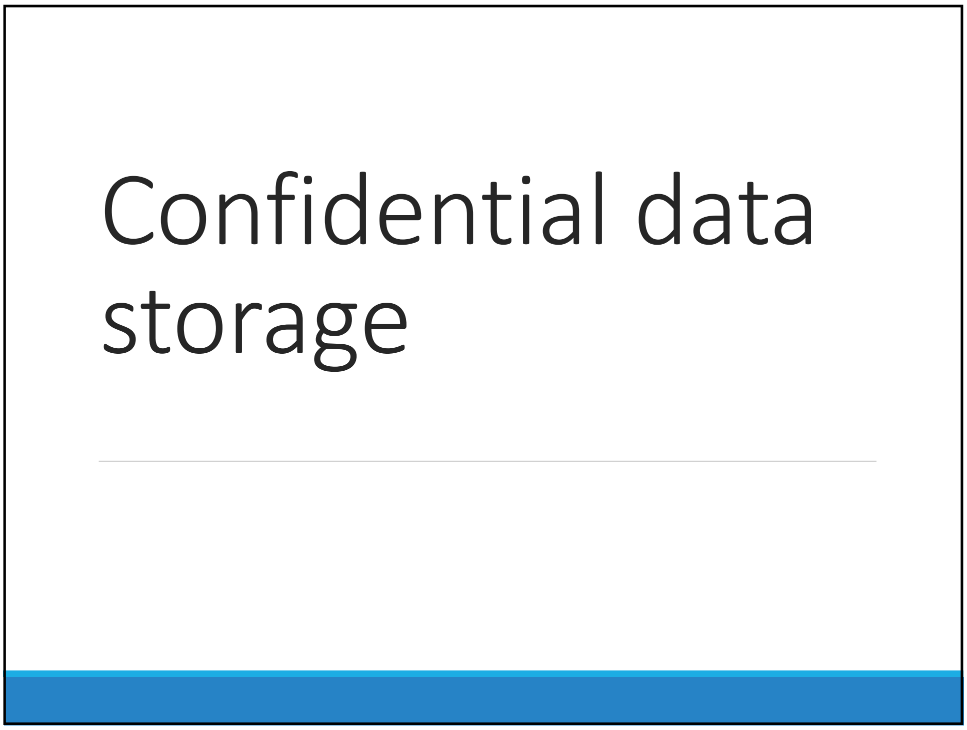
- Storage system available in the network
- Frequently created with RAID disks
- Cost: Hundreds to Thousands of Euro

## SAN: Storage Area Network

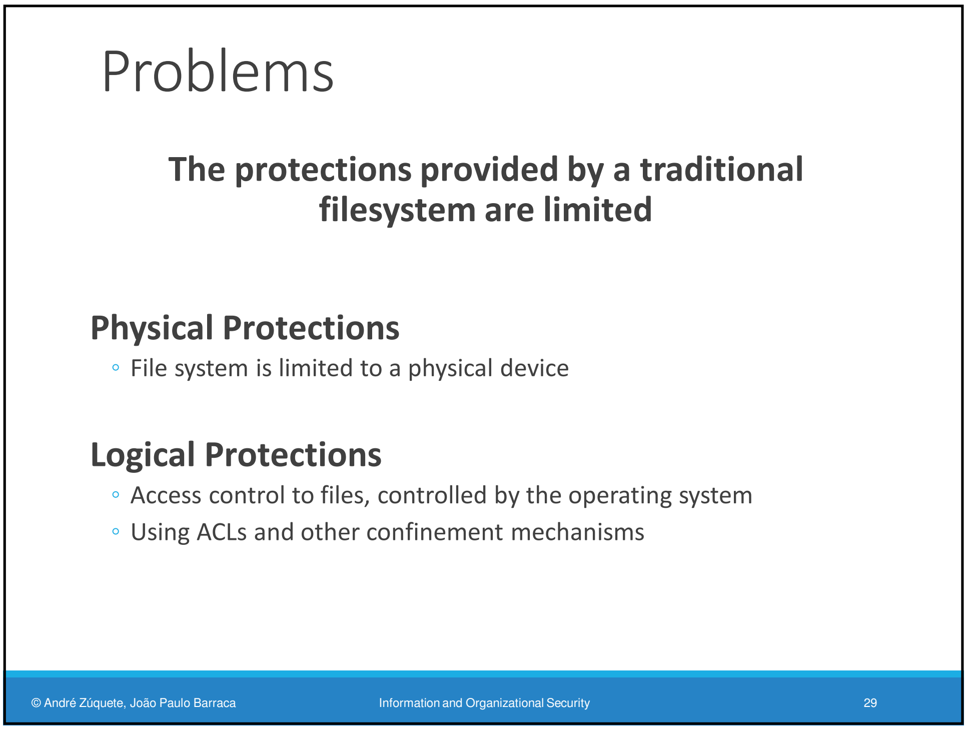
- Set of systems available in a network
- Implemented distributed storage with redundancy
- Cost: Hundreds of Thousands to Millions of Euro

## Advantages

- Allow centralizing the storage policies
- Provide a normalized interface, independent of the real storage
- May be used to distributed backups



# Confidential data storage



## Problems

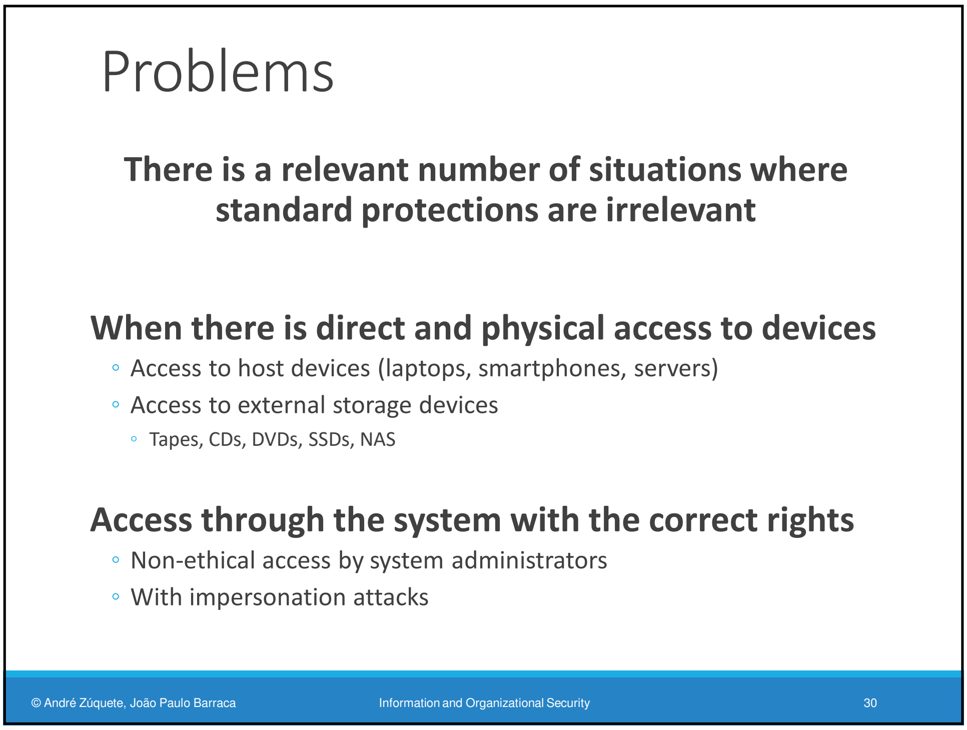
**The protections provided by a traditional filesystem are limited**

**Physical Protections**

- File system is limited to a physical device

**Logical Protections**

- Access control to files, controlled by the operating system
- Using ACLs and other confinement mechanisms



# Problems

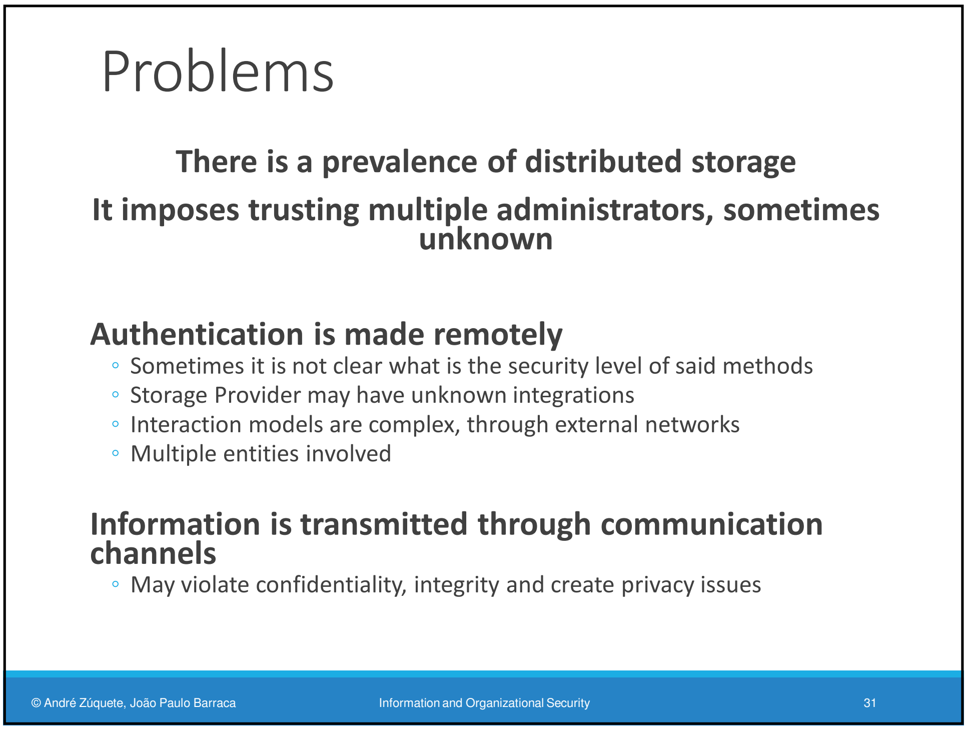
**There is a relevant number of situations where standard protections are irrelevant**

## When there is direct and physical access to devices

- Access to host devices (laptops, smartphones, servers)
- Access to external storage devices
  - Tapes, CDs, DVDs, SSDs, NAS

## Access through the system with the correct rights

- Non-ethical access by system administrators
- With impersonation attacks



# Problems

**There is a prevalence of distributed storage**

**It imposes trusting multiple administrators, sometimes unknown**

## Authentication is made remotely

- Sometimes it is not clear what is the security level of said methods
- Storage Provider may have unknown integrations
- Interaction models are complex, through external networks
- Multiple entities involved

## Information is transmitted through communication channels

- May violate confidentiality, integrity and create privacy issues

# Solution: Encrypt data

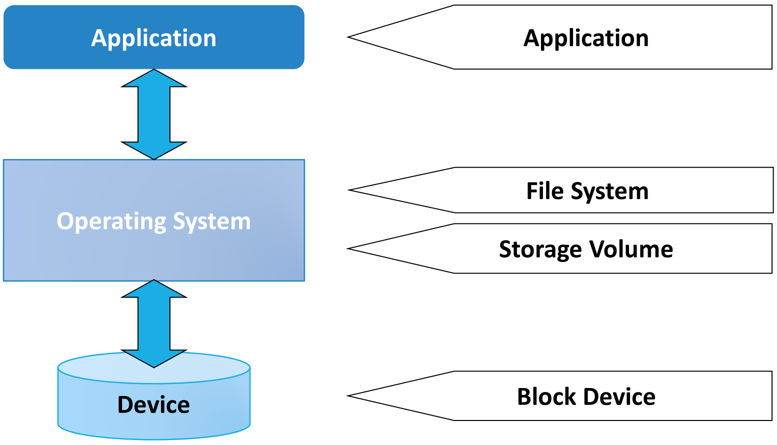
## Encryption/Decryption of file contents

- Enable secure transfer over insecure networks
- Enable secure storage in insecure locations
  - Managed by external entities, or in shared storages

## Problems of encryption

- Access to information
  - Users may lose the keys
    - Key loss = data loss
    - Key storage may reduce overall security
- File sharing
  - Sharing data implies sharing keys
- May interfere with standard management and recovery tasks
  - Content analysis, deduplication, indexing

# Approaches



Application

Operating System

Device

Application

File System

Storage Volume

Block Device

# Encryption in Applications

### Information is transformed by each application

- Little or no integration with other applications
- Usually, it is clear what is secure or not
  - Specific files with known file extensions

### Present  vulnerability windows

- Data must be decrypted to other files before being accessed

### Information may be processed by different algorithms/keys

- Adapted to a specific operating system or the security level
- May complicate the data recovery processes

### May difficult sharing data inside the encrypted package

- May imply extract data which is stored in a clear format

### Examples:

- PGP, AxCrypt, TrueCrypt, Veracrypt, etc.
- Also: RAR, ZIP, 7Zip, LZMA…

# Encryption in the File Systems

### Information is transformed when is sent from memory to the filesystem

- May be broad, from the entire filesystem into the global memory cache
  - No protection in shared servers as data is available to all applications
  - Security mechanism is harder to implement in distributed environments
    - Coordination of ACLs
- May be specific to the cache of a specific process
  - Protection in the case of shared servers as data access is context-bound
  - Client API decrypts data

### Examples

- EncFS, EXT4, NTFS, CFS

# Encryption at the volume level

## Information is transformed by the volume driver

- Transparent to applications and almost transparent to the OS
  - Requires support through a specific driver
- The entire volume will be made available (partition)

## Policies defined through applications or the controller

- Agnostic to the actual filesystem on top
  - Protects everything, including metadata
- But it doesn't differentiate between individual users

## Unable to solve problems related with distributed systems, but solves those related with mobile devices

- Distributed systems expose the filesystem after decryption
- Mobile devices: lost of stolen devices will keep data secure

## Examples:

- PGPDisk, LUKS, BitLocker, Filevault

# Encryption at the Device Level

## Block Device applies security policy internally

- At boot, the device must be unlocked
  - After the correct credentials are provided
- Encryption is implemented at the hardware/firmware

## Advantages

- No performance loss
- Data access is not trivial as keys are internal
- May be coordinated with applications (e.g., USB devices)

## Disadvantages

- After the device is unlocked, all data is made available
- Security is limited by the algorithms present
- The possible existence of backdoors is difficult to find and correct

# Encryption at the Device Level

## Devices have two distinct areas

- Shadow Disk: Read-Only, ~100MB with software to unlock it
- Real Disk: Read/Write. Contains user data

## Two keys used

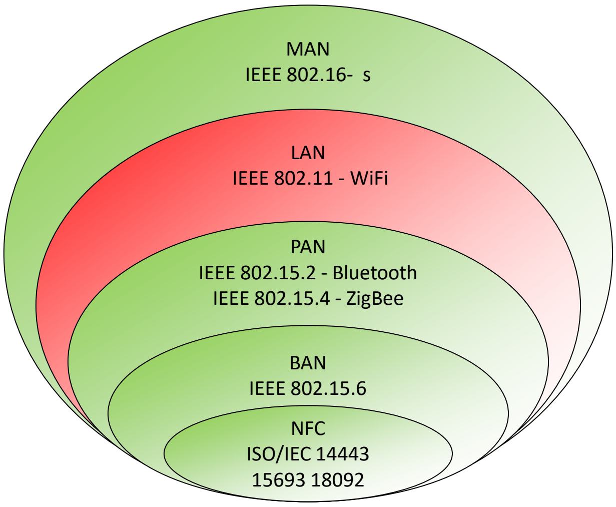
- KEK: Key Encryption Key (Authentication Key)
  - Provided by the user. Digest stored in the Shadow Disk
- MEK (or DEK): Media (Data) Encryption Key
  - Encrypted with the KEK

## Boot process

- BIOS will access Shadow Disk and boots
- Application in Shadow Disk requests password, decrypts KEK and verifies hash(KEK)
- If it matches, MEK is decrypted, and disk geometry is updated

# Security in 802.11 wireless networks

## Wireless (data) communications: A glance



MAN
IEEE 802.16- s

LAN
IEEE 802.11 - WiFi

PAN
IEEE 802.15.2 - Bluetooth
IEEE 802.15.4 - ZigBee

BAN
IEEE 802.15.6

NFC
ISO/IEC 14443
15693 18092

# Wireless vs. cabled communications: Security issues

## Broadcast communication

- Hard to enforce physical propagation boundaries
- Typical physical boundaries are useless to avoid:
  - Interference with communications
  - Eavesdropping of communications

## Mitigation

- Reduce interference and eavesdropping capabilities
  - At the physical layer
  - At the data link layer

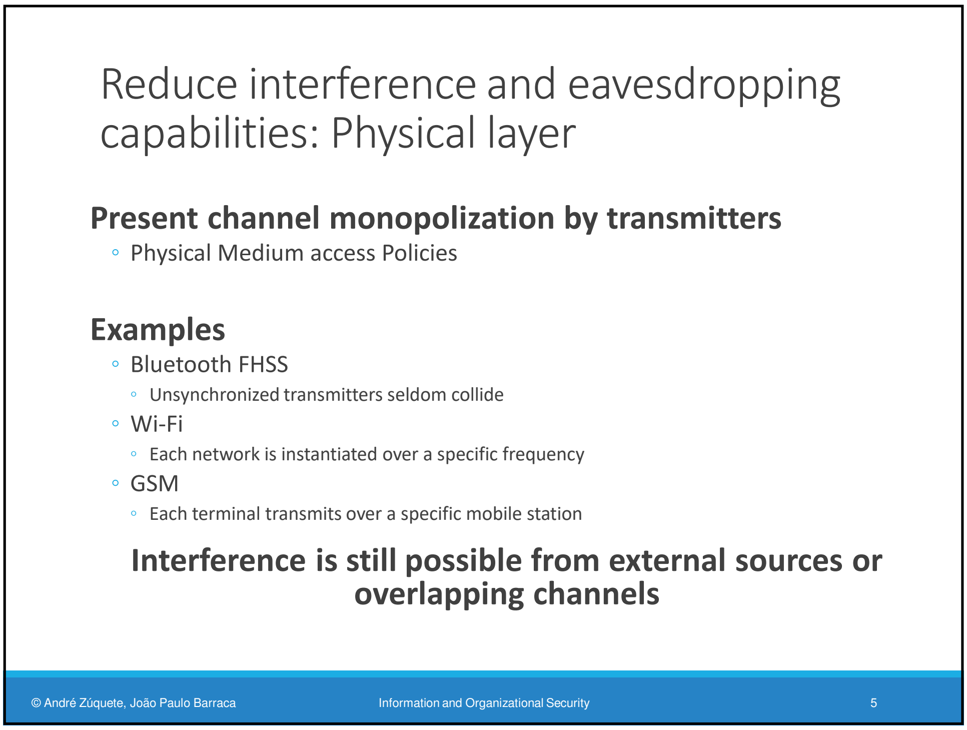
# Reduce interference and eavesdropping capabilities: Physical layer

## Prevent eavesdroppers from decoding the channel

- Channel coding needs to use some shared secret

## Example: Bluetooth FHSS (Frequency Hoping Spread Spectrum)

- Carrier changes frequency in a pattern known to both transmitter and receiver
  - The data is divided into packets and transmitted over 79 hop frequencies in a pseudo random pattern
  - Only transmitters and receivers that are synchronized on the same hop frequency pattern will have access to the transmitted data
- FHSS appears as short-duration impulse noise to eavesdroppers
  - The transmitter switches hop frequencies 1,600 times per second to assure a high degree of data security



## Reduce interference and eavesdropping capabilities: Physical layer

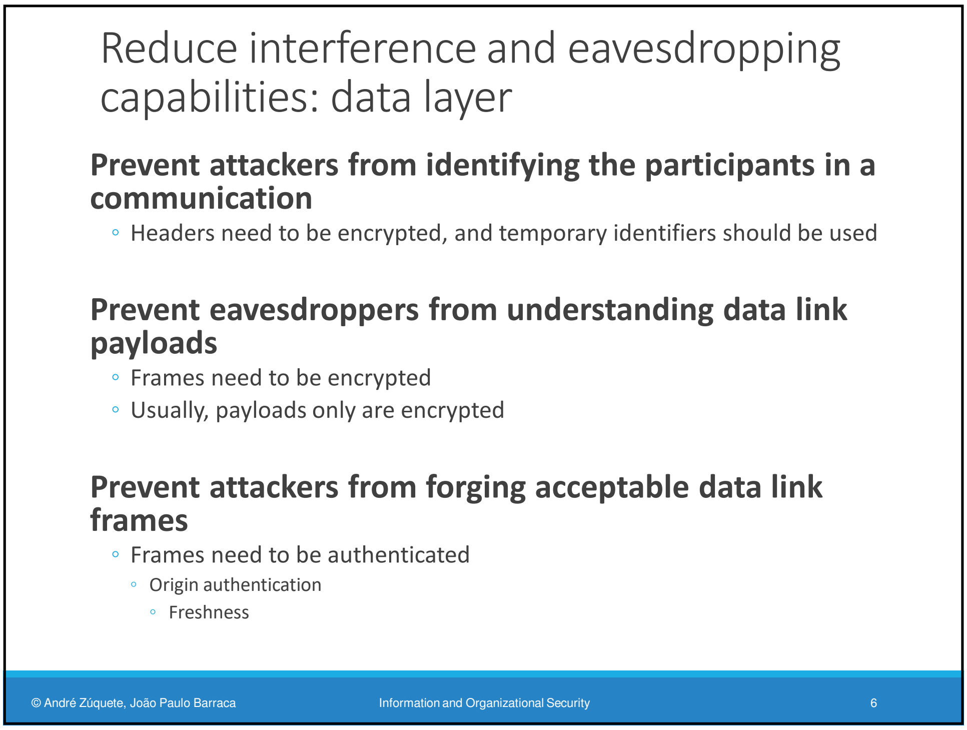
**Present channel monopolization by transmitters**

- Physical Medium access Policies

**Examples**

- Bluetooth FHSS
  - Unsynchronized transmitters seldom collide
- Wi-Fi
  - Each network is instantiated over a specific frequency
- GSM
  - Each terminal transmits over a specific mobile station

**Interference is still possible from external sources or overlapping channels**



## Reduce interference and eavesdropping capabilities: data layer

**Prevent attackers from identifying the participants in a communication**

- Headers need to be encrypted, and temporary identifiers should be used

**Prevent eavesdroppers from understanding data link payloads**

- Frames need to be encrypted
- Usually, payloads only are encrypted

**Prevent attackers from forging acceptable data link frames**

- Frames need to be authenticated
  - Origin authentication
    - Freshness

# IEEE 802.11:<br>Architecture (in structured networks)

## Station (STA)

- Device that can connect to a wireless network
- Has a (unique) identifier
  - Media Access Control (MAC) address
  - Today it is becoming popular its randomization (for anonymity sake)

## Access Point (AP)

- Device that allows the interconnection between a wireless network and other network devices or networks

## Wireless network

- Network formed by a set of STAs and AP that communicate using radio signals

# IEEE 802.11:<br>Structured network terminology

## Basic Service Set (BSS)

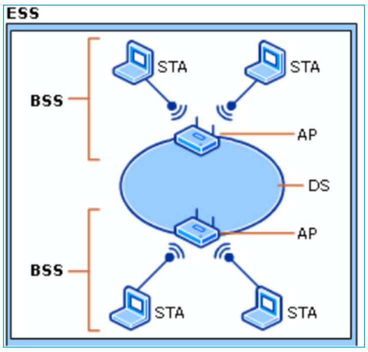
- Network formed by a set of STA associated to an AP

## Extended Service Set (ESS)

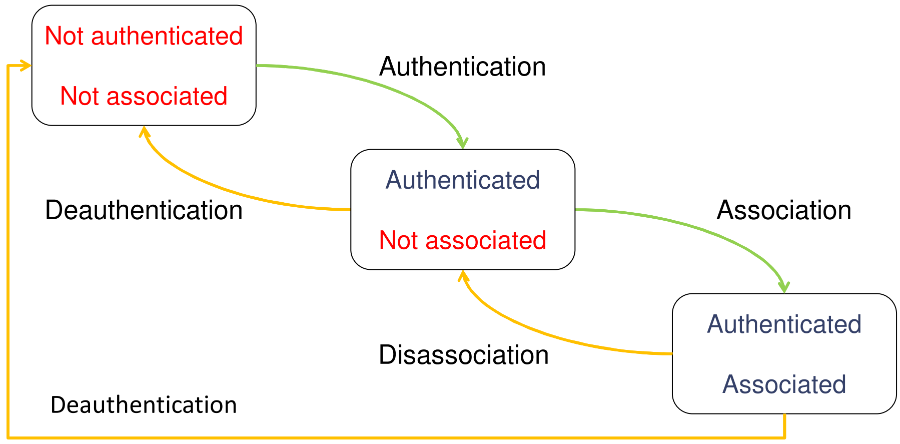
- Network formed by several BSS interconnected by a Distribution System (DS)

## Service Set ID (SSID)

- Identifier of a wireless network served by a BSS or ESS
- The same infrastructure can use several SSID



## IEEE 802.11:
## Authentication & Association state machine



## IEEE 802.11: Frame types

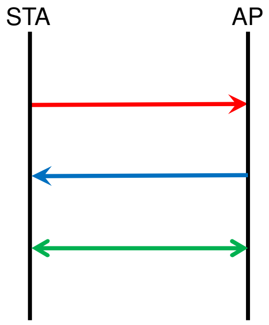
### Management frames

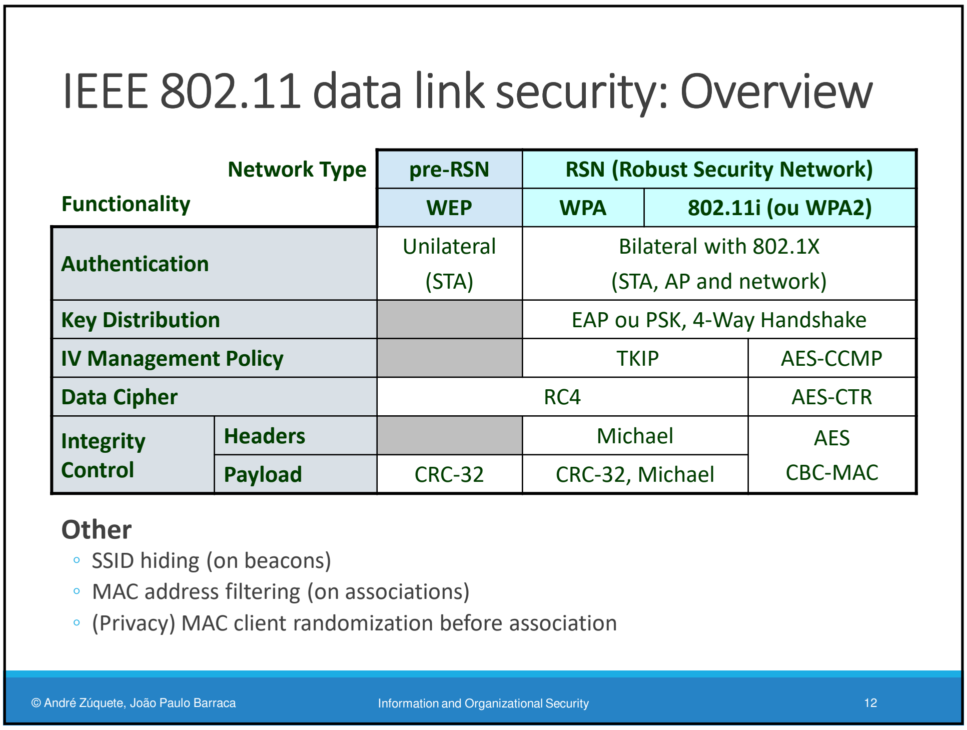
- Beacon
- Probe Request & Response
- Authentication Request & Response
- Deauthentication
- Association Request & Response
- Reassociation Request & Response
- Disassociation

### Control frames

- Request to Send (RTS)
- Clear to Send (CTS)
- Acknowledgment (ACK)

### Data Frames



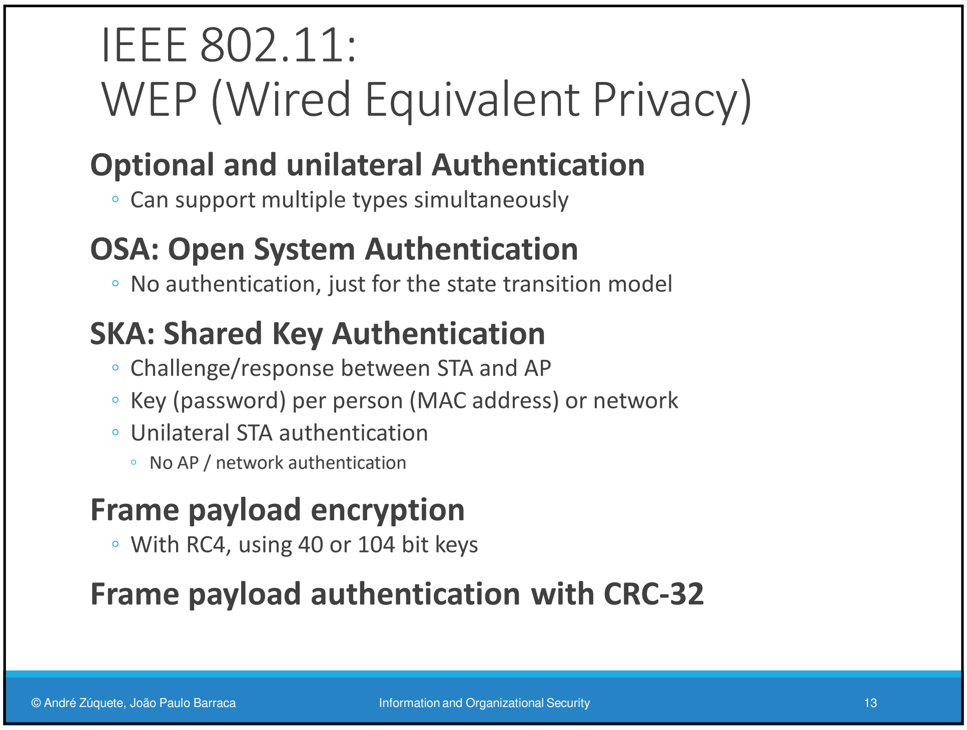


# IEEE 802.11 data link security: Overview

| Network Type | | pre-RSN | RSN (Robust Security Network) | |
|---|---|---|---|---|
| Functionality | | WEP | WPA | 802.11i (ou WPA2) |
| Authentication | | Unilateral (STA) | Bilateral with 802.1X (STA, AP and network) | |
| Key Distribution | | | EAP ou PSK, 4-Way Handshake | |
| IV Management Policy | | | TKIP | AES-CCMP |
| Data Cipher | | RC4 | | AES-CTR |
| Integrity Control | Headers | | Michael | AES CBC-MAC |
| | Payload | CRC-32 | CRC-32, Michael | |

**Other**

- SSID hiding (on beacons)
- MAC address filtering (on associations)
- (Privacy) MAC client randomization before association



# IEEE 802.11: WEP (Wired Equivalent Privacy)

**Optional and unilateral Authentication**

- Can support multiple types simultaneously

**OSA: Open System Authentication**

- No authentication, just for the state transition model

**SKA: Shared Key Authentication**

- Challenge/response between STA and AP
- Key (password) per person (MAC address) or network
- Unilateral STA authentication
  - No AP / network authentication

**Frame payload encryption**

- With RC4, using 40 or 104 bit keys

**Frame payload authentication with CRC-32**

# WEP: Lots of security problems …

## SKA is completely insecure

- An eavesdropper gets all it needs to impersonate a victim
  - No need to discover the password
- Rogue APs cannot be detected

## Same key for authentication and payload confidentiality

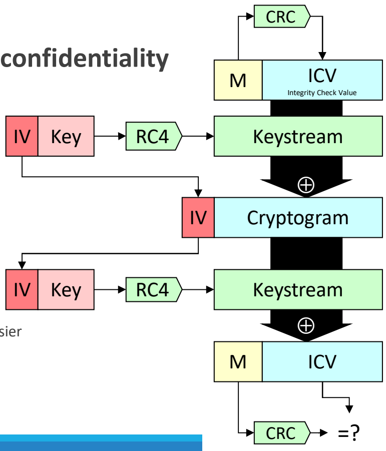
- No key distribution, keys overused

## Weak integrity control

- CRC-32 is linear
- Frame deterministic modification is trivial

## Mediocre IV management

- IV is too short (24 bits)
  - Easy to get cryptograms produced with the same IV
  - Same IV, same key ⇒ same keystream, cryptanalysis becomes easier
- IV is not managed at all
  - Reuse is not controlled / prevented



# Mitigation of WEP problems: WPA (WiFi Protected Access)

## WPA uses WEP in a safe way

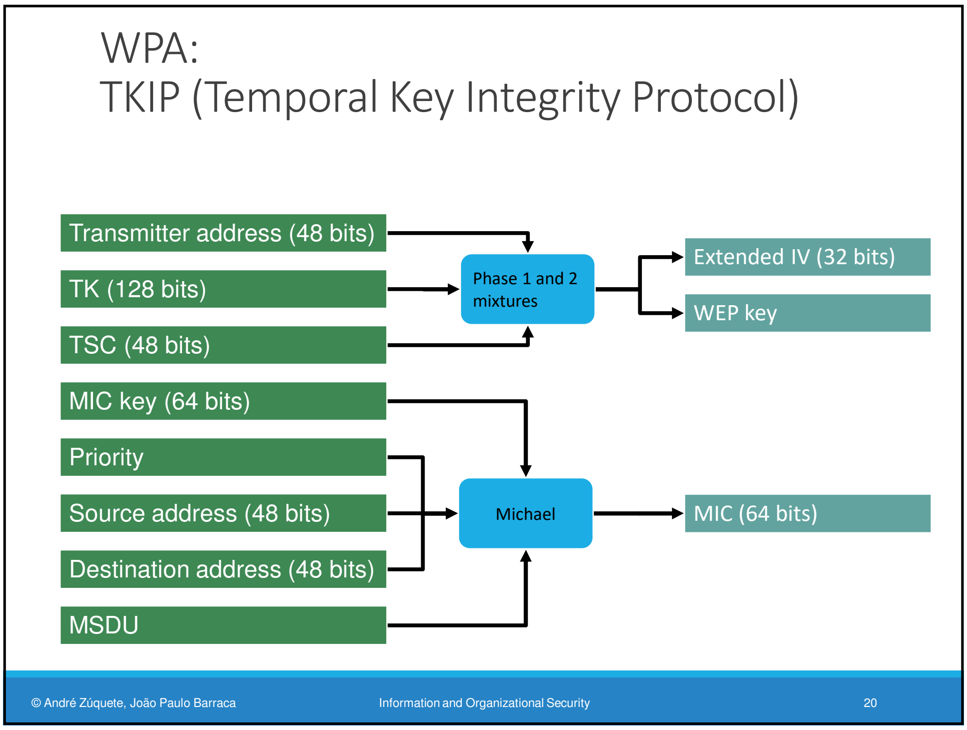
- A different RC4 key per frame
- RC4 week keys are avoided
- Extra cryptographic integrity control with Michael
- IV strict sequencing for preventing frame reuse

## Implemented first by device drivers

- Latter on firmware

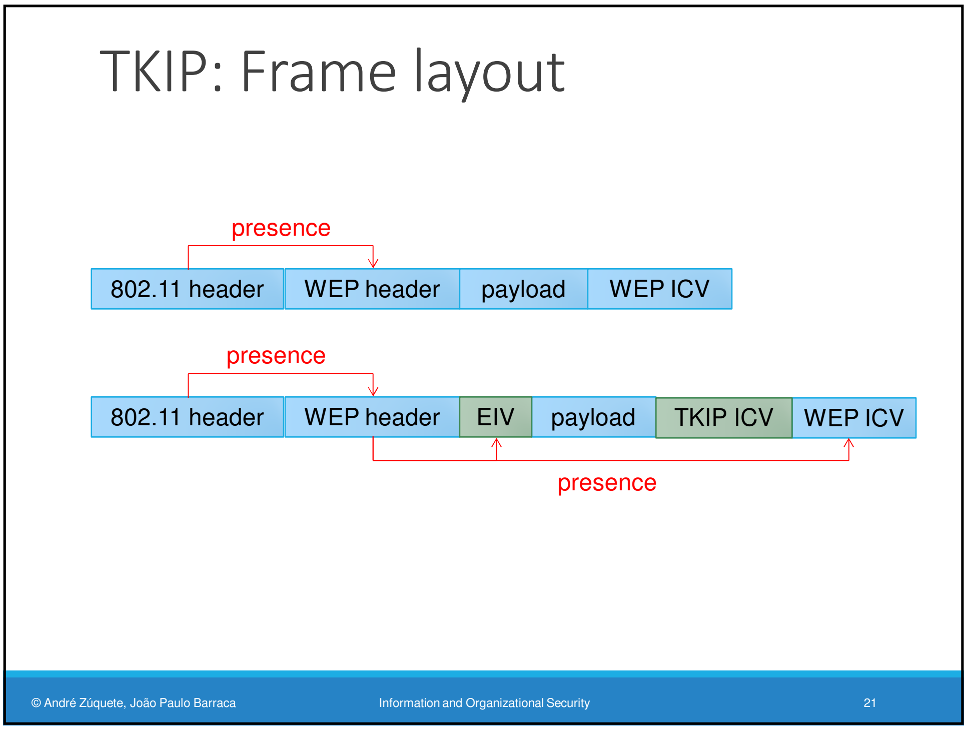
## Inline with 802.11i

- The actual 802.11 security standard
- WPA can be used with 802.1X for strong, mutual authentication



# WPA:
TKIP (Temporal Key Integrity Protocol)

Transmitter address (48 bits)
TK (128 bits)
TSC (48 bits)
Phase 1 and 2 mixtures
Extended IV (32 bits)
WEP key
MIC key (64 bits)
Priority
Source address (48 bits)
Destination address (48 bits)
MSDU
Michael
MIC (64 bits)



# TKIP: Frame layout

presence
802.11 header | WEP header | payload | WEP ICV

presence
802.11 header | WEP header | EIV | payload | TKIP ICV | WEP ICV
presence

# IEEE 802.1X:
# Port-Based Authentication

## Authentication model for all IEEE 802 networks

- Layer 2 mutual authentication

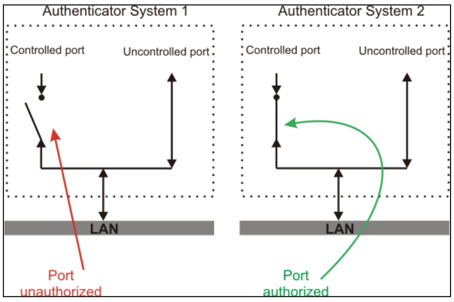
## Originally conceived for large networks

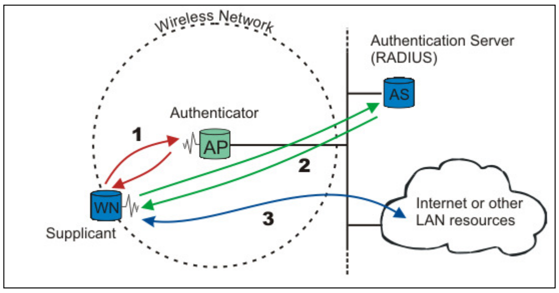
- University campus, etc.
- Model was extended for wireless networks

## Performs key distribution

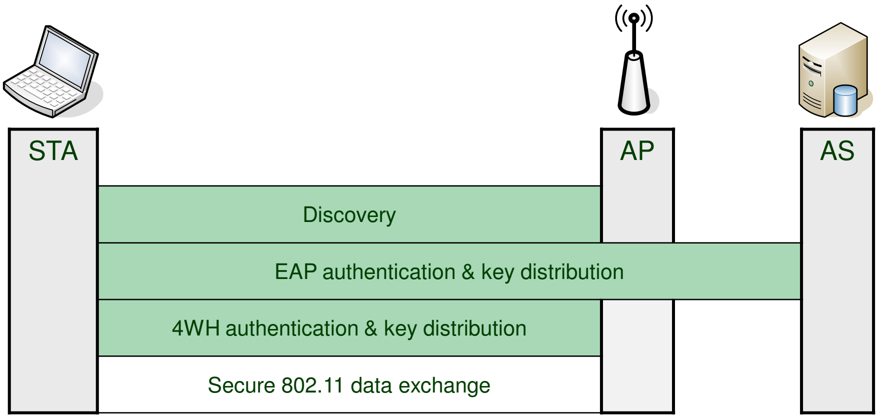
- Additional protocols focus in the remaining processes

# IEEE 802.1X:
# Architecture




Wireless Network
Authentication Server (RADIUS)
AS
Authenticator
AP
1
2
3
WN
Supplicant
Internet or other LAN resources

# IEEE 802.1X: Operational Phases



STA

AP

AS

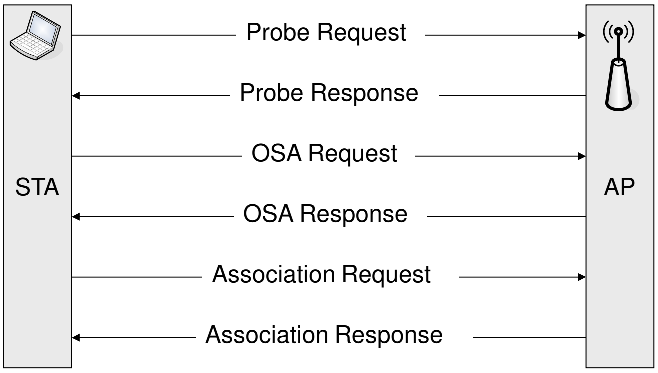
Discovery

EAP authentication & key distribution

4WH authentication & key distribution

Secure 802.11 data exchange

# IEEE 802.1X Phase 1: Discovery (802.11 messages)



STA

AP

Probe Request

Probe Response

OSA Request

OSA Response

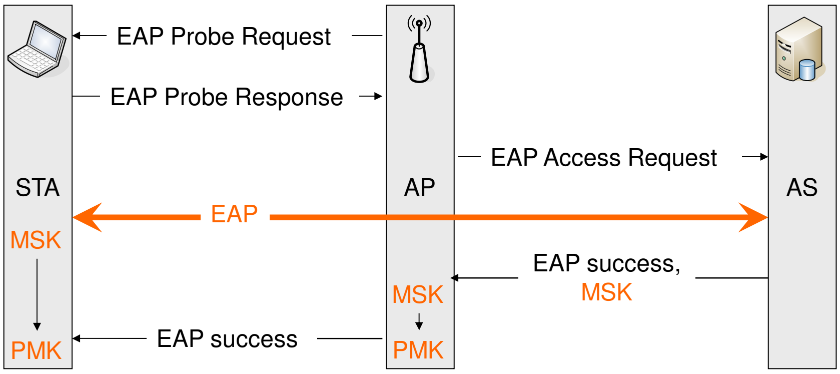
Association Request

Association Response

**STA only got access to the AP**

- 802.1X controlled port still closed

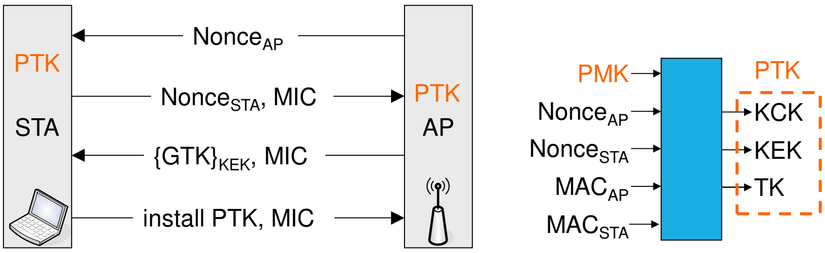
# IEEE 802.1X Phase 2: Authentication (EAP Messages)



**At the end of this phase AP and STA share crypto data**

- PMK (Pairwise Master Key)
- But 802.1X controlled port still closed

# IEEE 802.1X Phase 3: 4-Way Handshake (EAPoL Messages)



**At the end AP and STA share new, fresh crypto data**

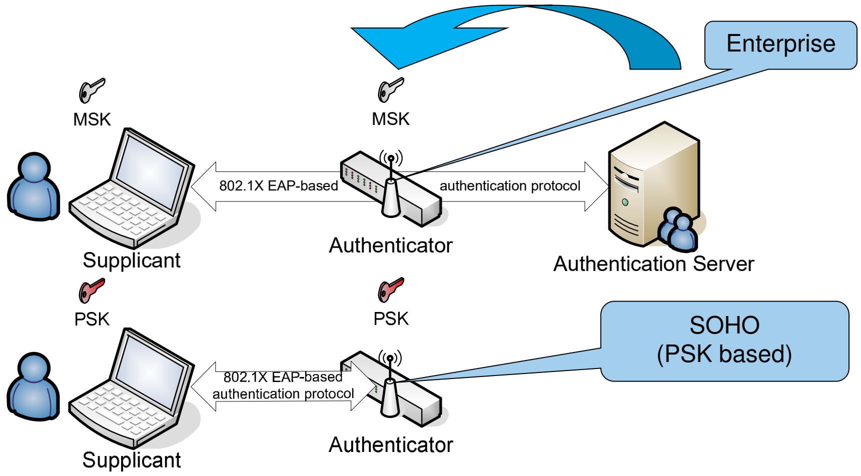
- PTK (*Pairwise Transient Key*)
- GTK (*Group Transient Key*)

**Both are convinced that the peer knows PMK and PTK**

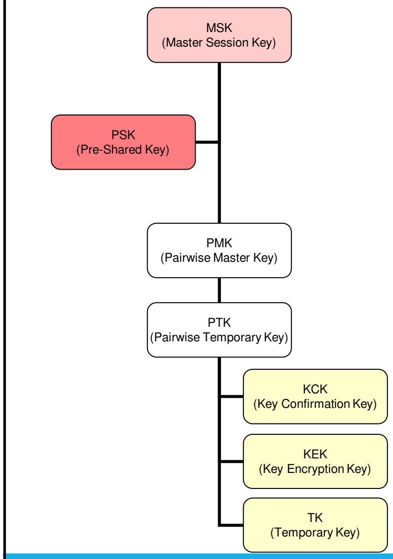
- Due to the use of MICs

**802.1X controlled port is now open for unicast traffic**

# IEEE 802.1X: Architectural options



# IEEE 802.1X: Complete key hierarchy



**MSK**

- Fresh outcome of an EAP protocol run
- Enterprise architecture

**PSK**

- Long-term AP-STA pre-shared key
- SOHO architecture

**PMK**

- Fresh key used for AP-STA mutual authentication and for key distribution in 4WH protocol runs

**PTK**

- Key used to protect AP-STA data exchanges
- KCK / KEK: 4WH protocol
- TK: 802.11 data frames

# EAP
(Extensible Authentication Protocol)

## Initially conceived for PPP

- Adapted to 802.1X

## AP not involved

- Relay EAP traffic
- Different EAP protocols do not imply changes in Aps

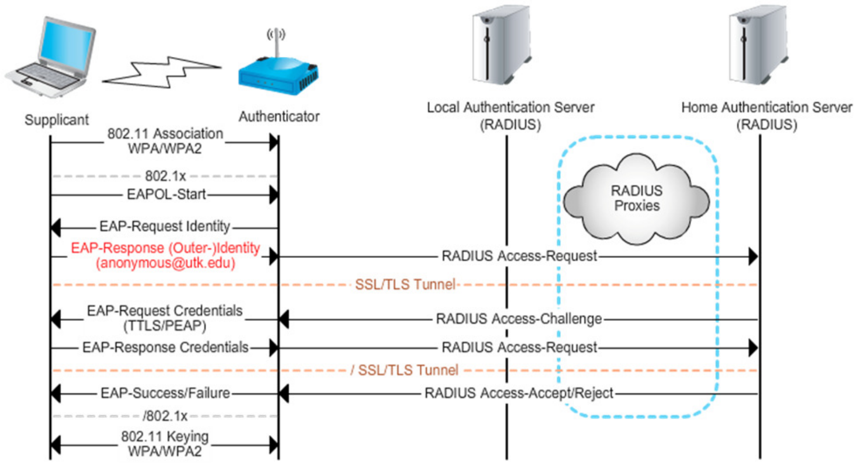
## Not conceived for wireless networks

- EAP traffic not protected
- Mutual authentication not mandatory
  - An STA can be fooled by a stronger (radio level), rogue AP

# Some EAP protocols for 802.1X

| | LEAP | EAP-TLS | EAP-TTLS | PEAP |
|---|---|---|---|---|
| **AS authentication** | digest (challenge, password) | Public Key (certificate) | | |
| **Supplicant authentication** | digest (challenge, password) | Public Key (certificate) | EAP,<br>Public Key (certificate) | PAP,<br>CHAP,<br>MS-CHAP, EAP |
| **Risks** | Identity exposure<br>Dictionary attacks<br>Host-in-the-Middle attacks | Identity exposure | | Possible identity exposure in phase 1 |

# Eduroam: 802.1X w/ PEAP + MS-CHAPv2



**Available on most University of the world**

- Local Authentication Servers (using RADIUS) for roaming access

# IEEE 802.11i (WPA2)

**Defines Robust Security Networks (RSN)**

- Those that support WPA and 802.11i

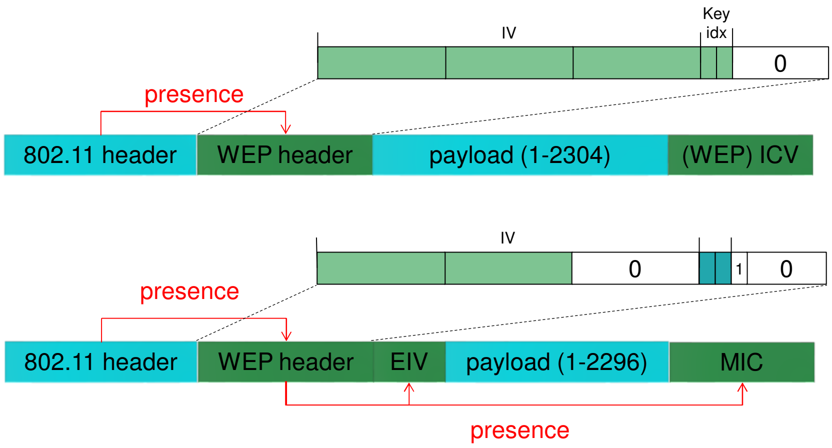
**Uses advanced security mechanisms for frame protection**

- Advanced Security Algorithm (AES) for payload encryption and frame integrity control

**Uses 802.1X for network access authentication**

- Simplified Pre-Shared Key (PSK) mode for SOHO (Small Office, Home Office) environments
- EAP-based protocol for enterprise environments

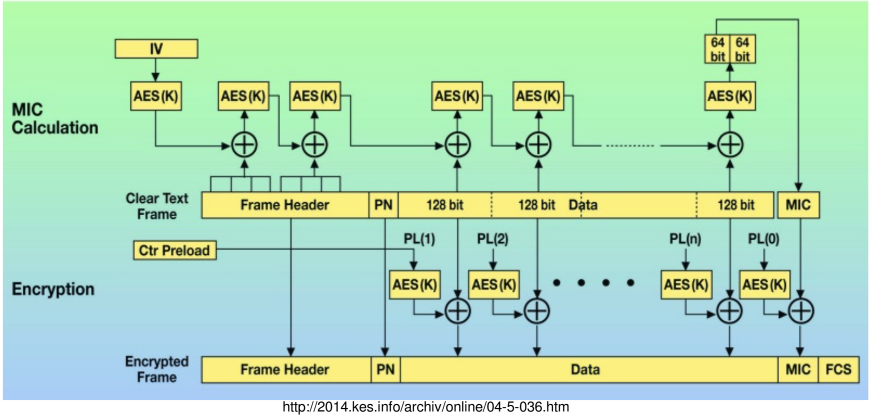
# WEP vs AES-CCMP: Frame layout



# WPA2 frame protection

## CCMP - Counter CBC-MAC Protocol

- 128-bit keys, protection of headers, data, with cipher and authentication



http://2014.kes.info/archiv/online/04-5-036.htm

# 802.11w: Protected Management Frames

### Management frames that can be used for DoS attacks are authenticated

- Deauthentication & Deassociation requests
- Other management frames unicasted or broadcast by an AP

### BIP (Broadcast Integrity Protocol)

- IGTK (Integrity GTK)
- For protecting part of the AP broadcast traffic

### Security Association Query Request / Response

- Help to deal with desynchronization issues

# IEEE 802.11 security: Are all the problems solved? No!

### Dictionary attacks are still possible with PSK or EAP-based authentication

- And they will continue to be as long as (weak) passwords are chosen by people

### There are still some unprotected frames

### Some weaknesses at the CSMA level

- Low Congestion Window (CW) values allow attackers to get all the bandwidth