Security
PKI

2024/25 Q2

Jaime Delgado *

DAC – UPC

* Part of the material comes from other sources.

Security

- Introduction
- Cryptography
- **Public Key Infrastructure (PKI)**
- Security in Applications

Security

- Public Key Infrastructure (PKI)
  - Basic concepts
  - Validation protocols
  - Trust models
  - X.509 certificates
  - PKCS

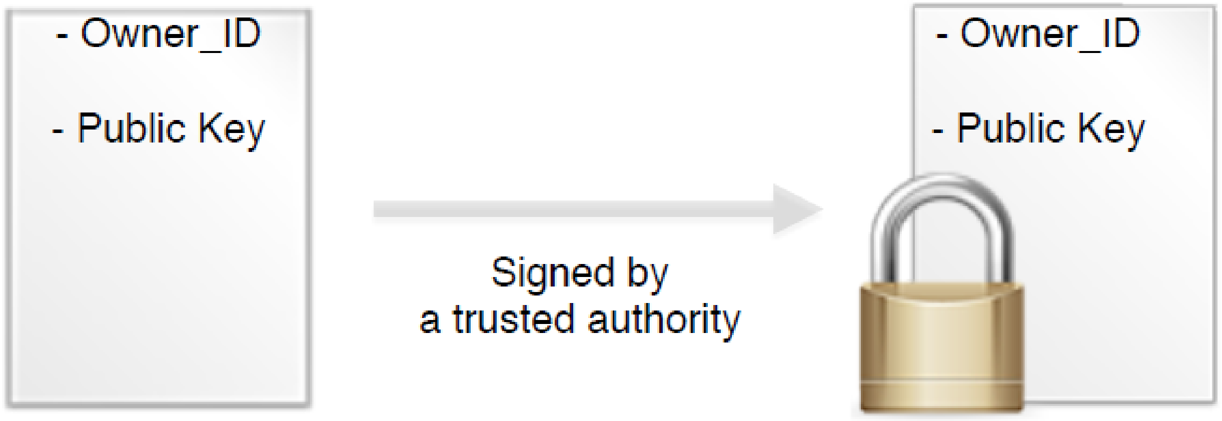
PKI - Basic concepts

- Private (symmetric) key → Need to protect key distribution
- Public (asymmetric) key → No need (public+secret)
- RSA, ElGamal algorithms work, *but …*
- **How do I trust in the public key I get?**
- Need to trustworthily link **User ← → Public key!**

PKI - Basic concepts

- Link User id ← → Public key!
- Certificate:

  Owner (id), Public key, "Authority" signature
- Authority → "Trust", signature verification



- Certification Authority + Registration Authority

PKI - Basic concepts

- Certification/*Certificate* Authority:
  - Issues and signs certificates (+ keys)
  - CA certificate?
    - Signed by another CA, or
    - Self-signed
  - Well protected secret key
- Registration Authority:
  - User identification
  - Administrative issues

PKI - Basic concepts

- Certification Authorities in Spain:
  - Fábrica Nacional de Moneda y Timbre (FNMT)
  - Agència Catalana de Certificació (CATCert)
  - Agencia Notarial de Certificación (ANCERT)
  - Autoridad de Certificación de la Abogacía (ACA)
  - Camerfirma
  - Firma Profesional
  - IZENPE
  - …

PKI - Basic concepts

- Certification Authorities around the World:
  - Symantec (formerly VeriSign)
  - Comodo
  - GlobalSign
  - Digicert
  - GoDaddy
  - Let’s Encrypt
  - …

Security

- Public Key Infrastructure (PKI)
  - Basic concepts
  - **Validation protocols**
  - Trust models
  - X.509 certificates
  - PKCS

Validation authority (VA)

- To check correctness of certificates
- External or internal to the CA
- Validate status of the certificate (or *certification chain*)
- Possible output:
  - Valid
  - Suspended
  - Revoked *(→ **CRL**: Certificate Revocation List)*
  - Unknown

Validation protocols

- To obtain the certificate state in “real time” (*validation of certificates*).
- Examples:
- **OCSP** (X.509 Internet Public Key Infrastructure Online Certificate Status Protocol)
  - RFC 6960 (2013)
  - Updates RFC 5912 (2010): New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)
- **SCVP** (Server-based Certificate Validation Protocol)
  - RFC 5055 (2007)
- Time stamping! Time Stamping Authority (TSA)

OCSP

- Online Certificate Status Protocol

- Features:
  - VA (OCSP server) gets CRLs from the CAs, validates and signs
  - ⬆ User (OCSP client) gets real time information
  - ⬆ User just receives an entry with the specific requested information
  - ⬇ Clients need to repeat the process for all the certification tree

SCVP

- Server-Based Certificate Validation Protocol

- Features:
  - ⬆ – Allows a client to delegate *certification path construction* and *certification path validation* to a server

    “**Certification path validation**”:
    Making sure that none of the certificates in the path are revoked.
  - ⬆ – Allows simplification of client implementations
  - ⬇ – More complex servers

Time Stamping Authority

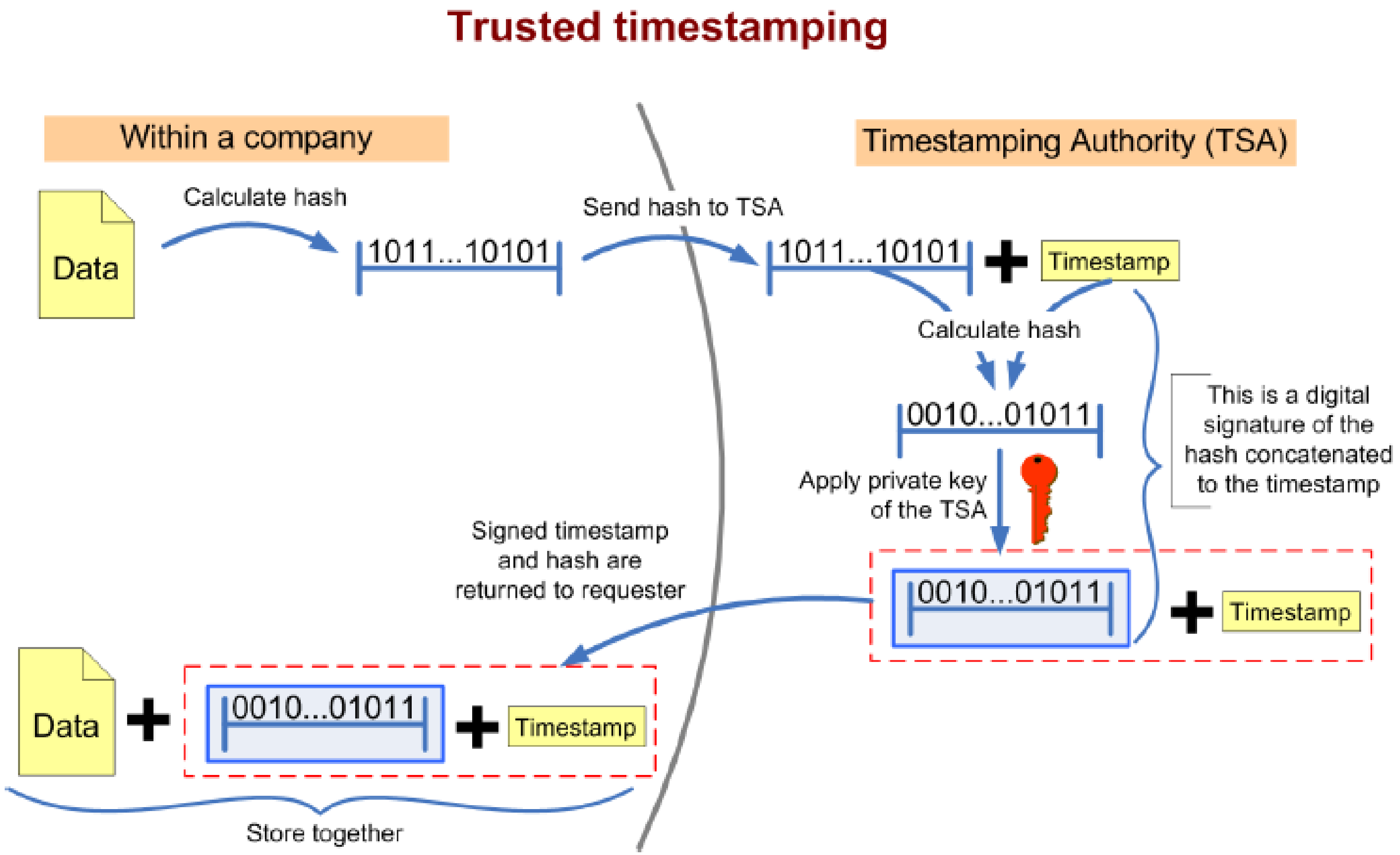
Time stamping authority (TSA) is in charge to sign a message to prove that it was generated before a given time

This is important to fulfill the non-repudiation property

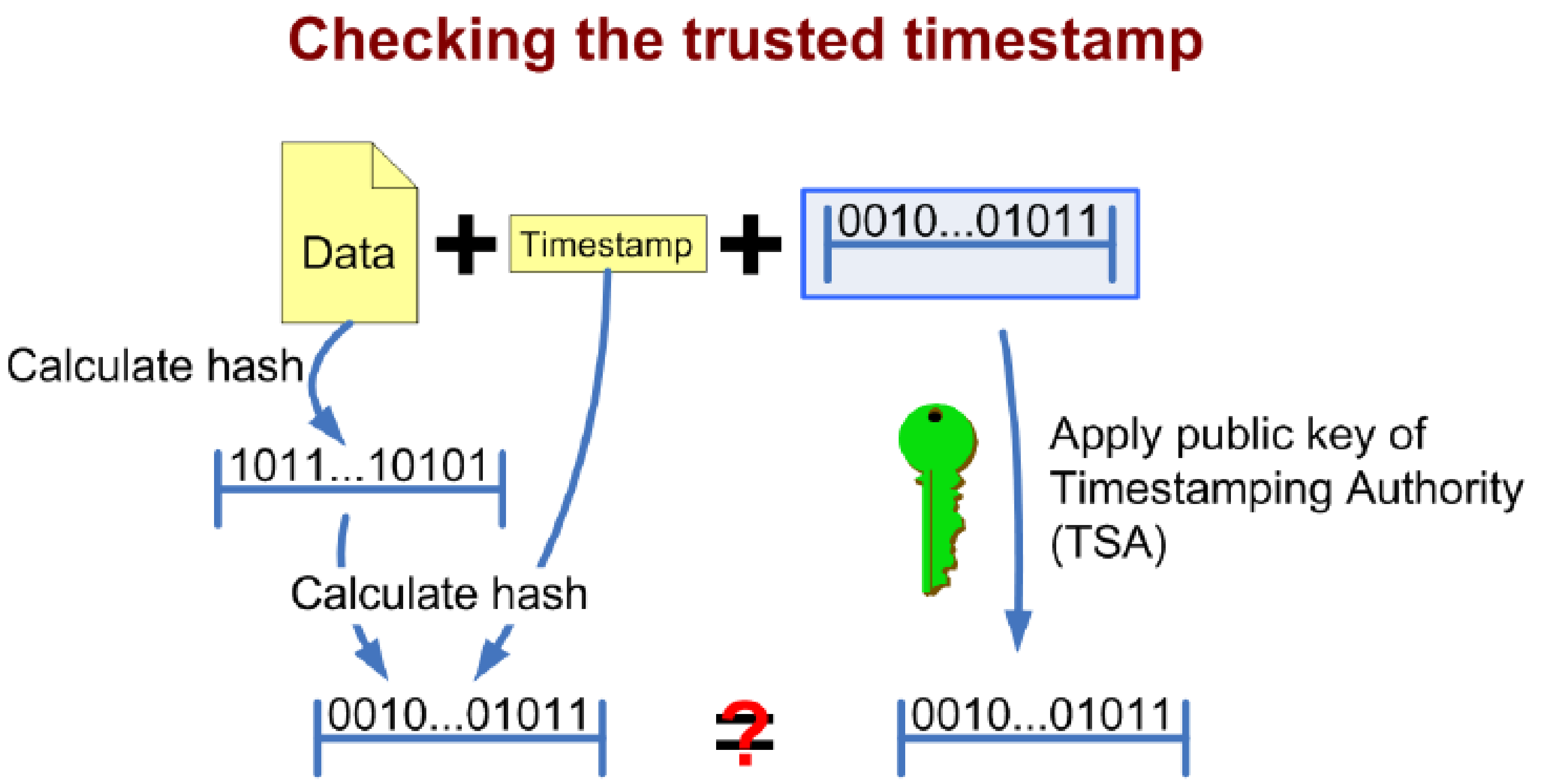
Scenarios were TSAs are very important:

- **Verification of a digitally signed document**. If the corresponding certificate has been revoked, the time stamp allows us to decide if the document was signed before revocation
- **Deadline document delivery**. Time stamp allows us to check if the document was delivered on time
- **Audits**. Time stamps permits to date all the historical entries

Time Stamping Authority



Time Stamping Authority



If the calculated hashcode equals the result of the decrypted signature, neither the document or the timestamp was changed and the timestamp was issued by the TTP. If not, either of the previous statements is not true.

Time Stamping protocols

- *RFC 3161-based Public Key Infrastructure*
- RFC 3161 (2001): **Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)**
- Describes:
  - format of request sent to a TSA
  - returned response
- Establishes:
  - several security-relevant requirements for TSA operation (processing requests to generate responses)
- Uses ASN.1

Security

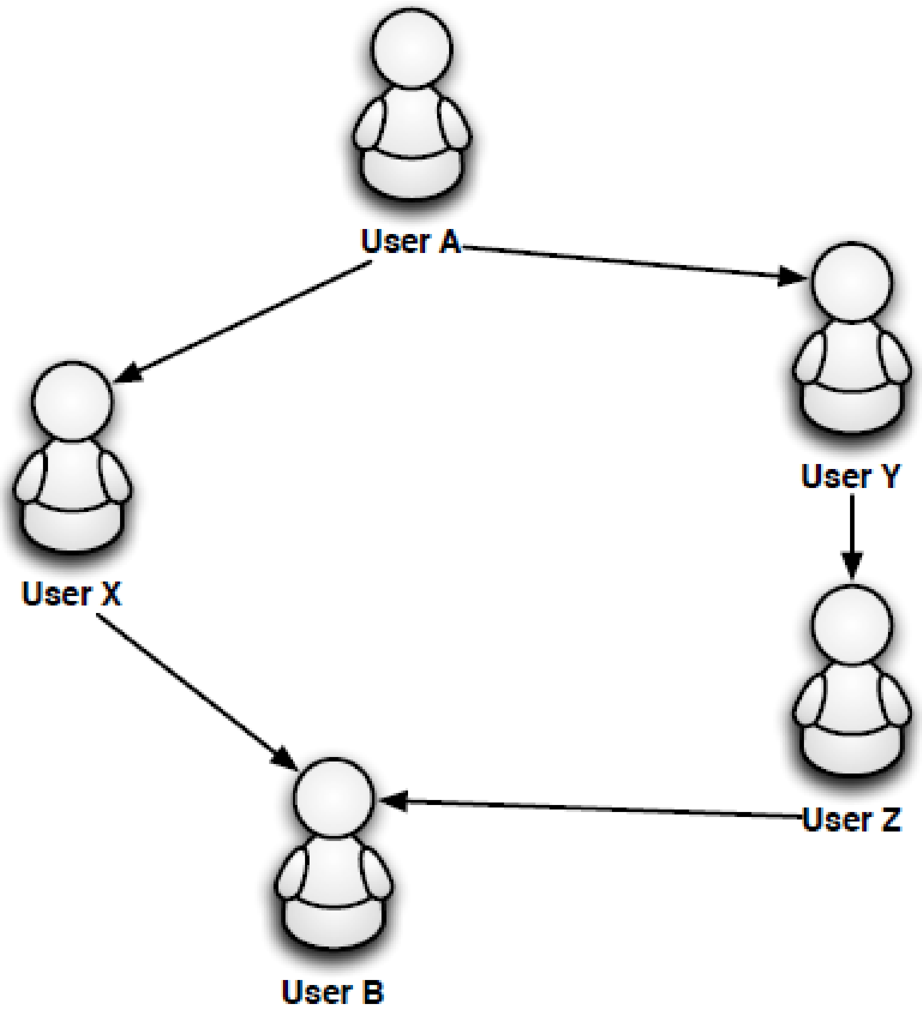
- Public Key Infrastructure (PKI)
  - Basic concepts
  - Validation protocols
  - **Trust models**
  - X.509 certificates
  - PKCS

PKI- Trust models

- Distributed: No central TTP (Trusted Third Party)
- Plain: One CA
- Hierarchical: Root (self-signed) $\rightarrow$ tree of CAs
- Hybrids:
  - Hierarchical trust **list** model (*user*)
  - Cross-certification (*CA*)
  - Bridge certification model (Bridge CA)

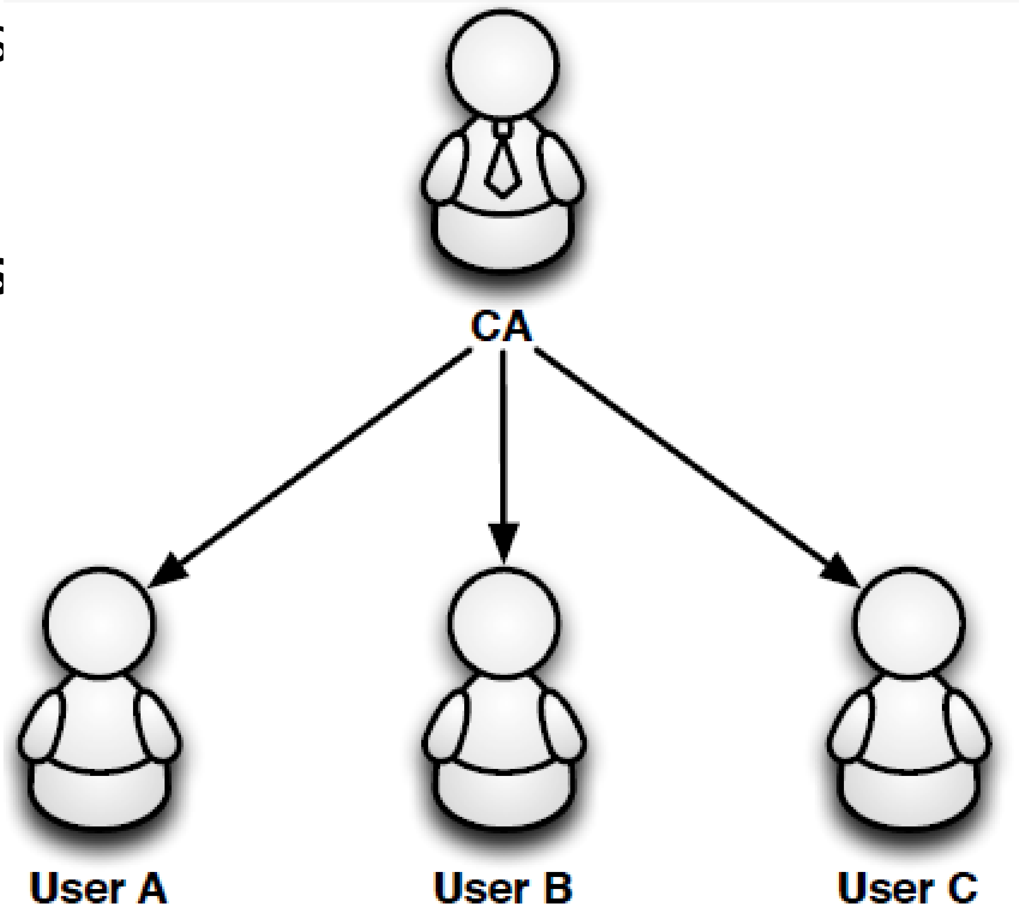
PKI- Trust models

- **Distributed**: No central TTP (Trusted Third Party)
- Plain: One CA
- Hierarchical: Root
- Hybrids:
  - Hierarchical trust l
  - Cross-certification
  - Bridge certificatior



PKI- Trust models

- Distributed: No central TTP
- **Plain**: One CA (self-signed)
- Hierarchical: Root (s
- Hybrids:
  - Hierarchical trust **lis**
  - Cross-certification (
  - Bridge certification

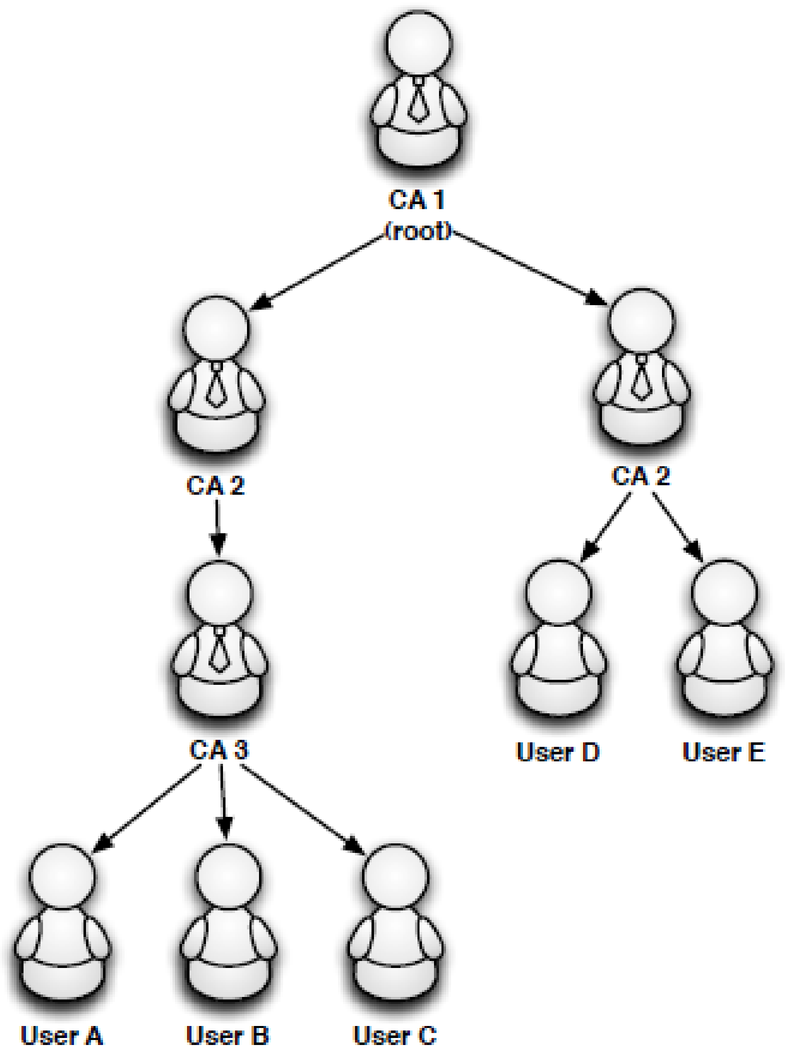


PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- **Hierarchical**: Root (self-signed) → tree of CAs
- Hybrids:
  - Hierarchical trust **list** model (*user*)
  - Cross-certification (*CA*)
  - Bridge certification model (Bridge CA)

PKI- Trust models

- Distribut
- Plain: Or
- **Hierarch** e of CAs
- Hybrids:
  - Hierarc
  - Cross-c
  - Bridge

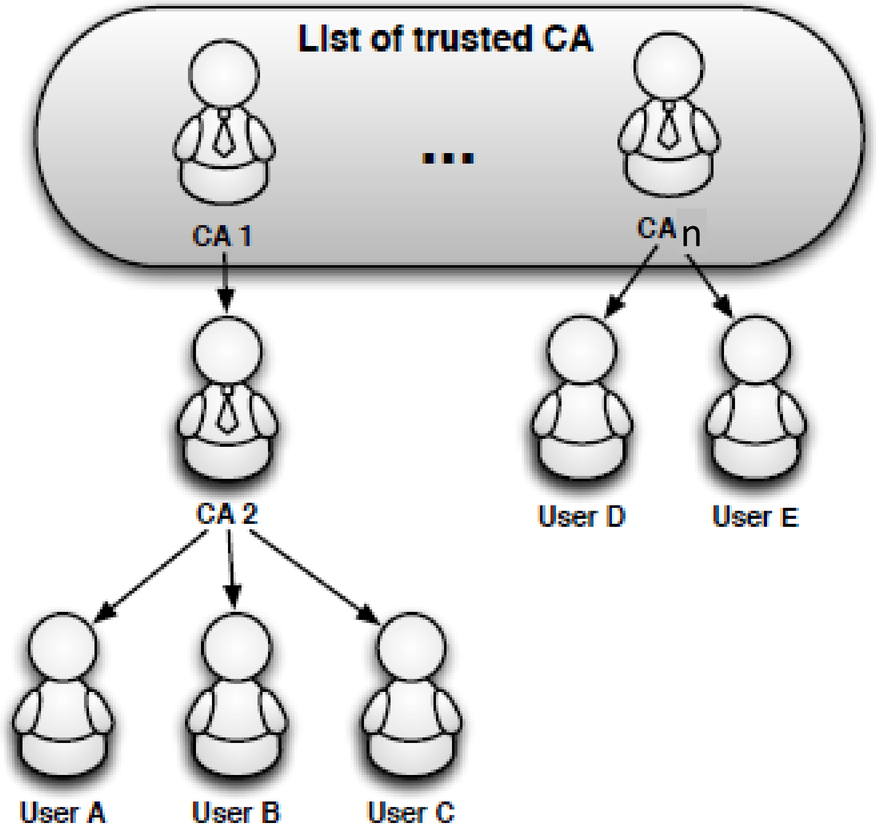


PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- Hierarchical: Root (self-signed) → tree of CAs
- **Hybrids**:
  - Hierarchical trust **list** model (*user*)
  - Cross-certification (*CA*)
  - Bridge certification model (Bridge CA)

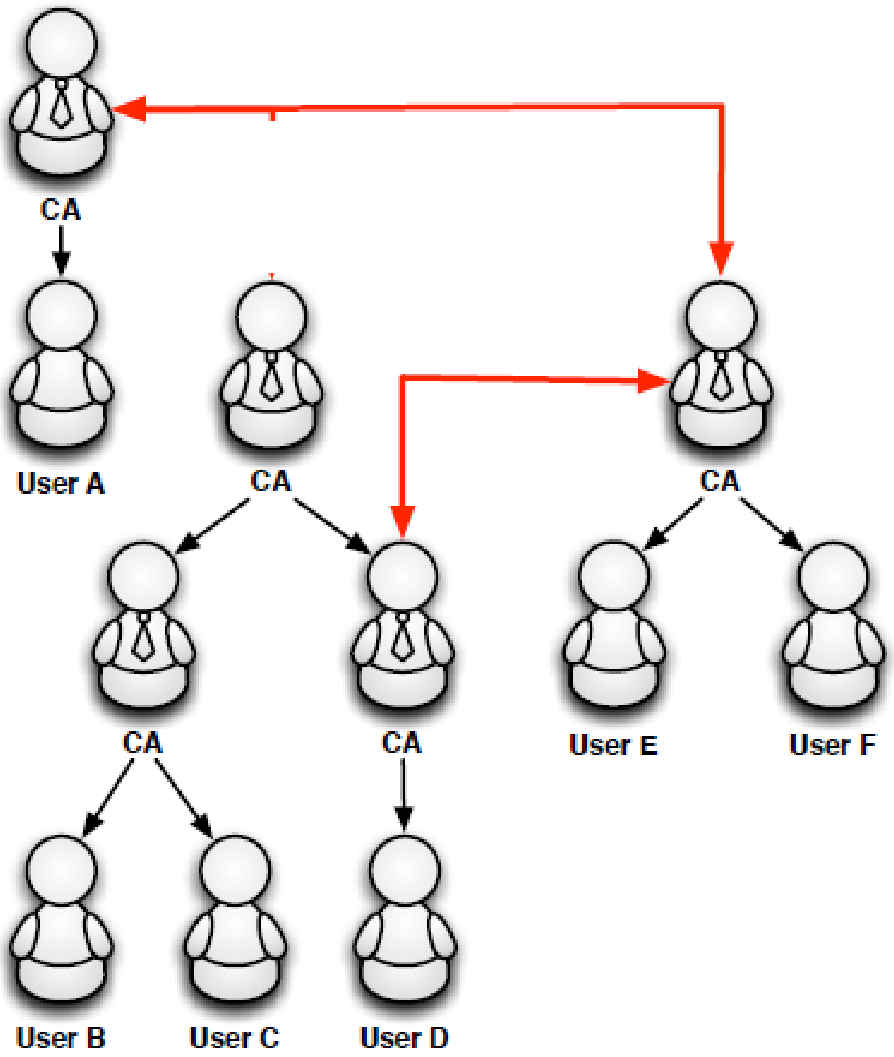
PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- Hierarchical: Root (self-signed) → tree of CAs
- Hybrids:
  - Hierarchical trust **list** m
  - Cross-certification (*CA*)
  - Bridge certification mo



PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- Hierarchical: Root (self-s
- Hybrids:
  - Hierarchical trust **list** m
  - Cross-certification (*CA*)
  - Bridge certification moc

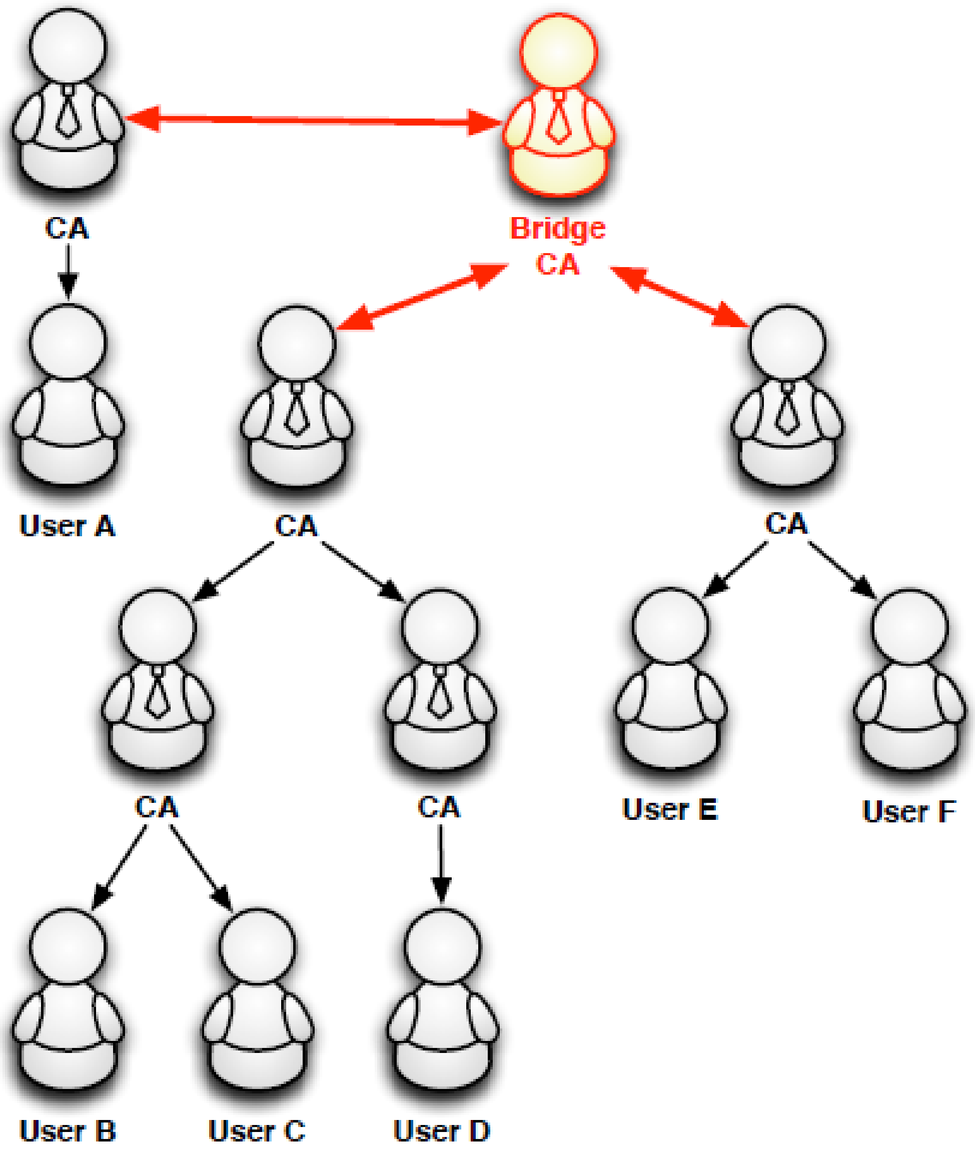


PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- Hierarchical: Root (self-signed) → tree of CAs
- Hybrids:
  - Hierarchical trust **list** model (*user*)
  - Cross-certification (*CA*)
  - **Bridge** certification model (Bridge CA)

PKI- Trust models

- Distribu
- Plain: O
- Hierarcl   of CAs
- Hybrids
  - Hierar
  - Cross-
  - **Bridge**



PKI- Trust models

- Distributed: No central TTP
- Plain: One CA
- Hierarchical: Root (self-signed) → tree of CAs
- Hybrids:
  - Hierarchical trust **list** model (*user*)
  - Cross-certification (*CA*)
  - Bridge certification model (Bridge CA)

Security

- Public Key Infrastructure (PKI)
  - Basic concepts
  - Validation protocols
  - Trust models
  - **X.509 certificates**
  - PKCS

X.509 - Basic ideas

- Part of The Directory:
  ITU-T X.500 series of recommendations -
  ISO/IEC 9594 (1st version 1988. Some versions till 2008 and 2012)
- Version 3 (1997). Still more extensions
- “Distinguished name” concept. Includes “Alternative names” (e-mail, IP address, …)
- Using **ASN.1** for data representation
- Adapted to Internet and Web environment
- ***Moved to Internet IETF***

X.509 - Basic ideas

- Part ITU-T s - ISO/I 3 and 2012)
- Versi
- "Dist s "Alte , ...).
- Using **ASN.1** for data representation
- Adapted to Internet and Web environment
- ***Moved to Internet IETF***

Expressed in ASN.1
(Abstract Syntax Notation 1)
(ITU-T X.680 & 690, ISO/IEC 8824 & 8825)

→ BER (Basic Encoding Rules),
DER (Distinguished ER)

Originally developed for OSI standards
No XML → Binary coding!

X.509 - IETF

- *Adapted to Internet and Web environment*
- "Internet X.509 Public Key Infrastructure Certificate and
  Certificate Revocation List (CRL) Profile"
- RFC 3280 (2002) → *RFC 5280 (2008)*
- Updates in **RFC 6818** (2013)
  + RFCs 8398 & 8399 (2018)
  RFC 8399 → RFC 9549 (March 2024)

Attributes

Basic (data) attributes

- Version
- Serial Number
- Signature Algorithm ID
- Issuer
- Validity:

  Not Before, Not After
- Subject (owner)
- Subject Public Key Info:

  Public Key Algorithm, Subject Public Key

Signature attributes

- Certificate Signature Algorithm
- Certificate Signature

Attributes

Basic (data) attributes

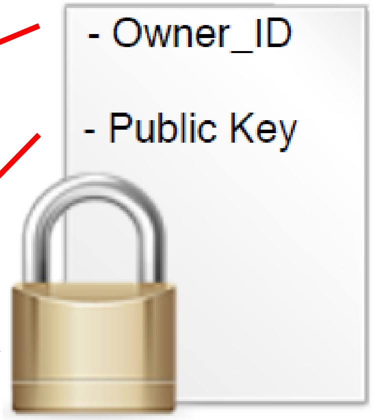
- Version
- Serial Number
- Signature Algorithm ID
- Issuer
- Validity:

  Not Before, Not After
- Subject (owner)
- Subject Public Key Info:

  Public Key Algorithm, Subject Public Key

Signature attributes

- Certificate Signature Algorithm
- Certificate Signature



X.509 certificate example

DATA:

Version: 3 (0x2)

Serial Number: 7829 (0x1e95)

Signature Algorithm: md5WithRSAEncryption

Issuer: C=ZA, ST=Western Cape, L=Cape Town,O=Thawte Consulting cc,
OU=Certification Services Division, CN=Thawte Server CA
emailAddress=server-certs@thawte.com

Validity:

Not Before:	Jul 9 16:04:02 2011 GMT
Not After:	Jul 9 16:04:02 2012 GMT

Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala, OU=FreeSoft,
CN=www.freesoft.org
emailAddress=baccala@freesoft.org

Subject Public Key Info:

Public Key Algorithm:	rsaEncryption
RSA Public Key:	(1024 bit) Modulus (1024 bit): … Exponent: 65537 (0x10001)

SIGNATURE:

Certificate Signature Algorithm: md5WithRSAEncryption

Certificate Signature: …

X.509 certificate example

Certificate: Data: Version: 3 (0x2) Serial Number: 7829 (0x1e95) Signature Algorithm: md5WithRSAEncryption Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, OU=Certification Services Division, CN=Thawte Server CA/emailAddress=server-certs@thawte.com Validity Not Before: Jul 9 16:04:02 1998 GMT Not After : Jul 9 16:04:02 1999 GMT Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala, OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org Subject Public Key Info: Public Key Algorithm: rsaEncryption RSA Public Key: (1024 bit)

Modulus (1024 bit):
00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb: 33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17: 16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77: 8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8: e8:35:1c:9e:27:52:7e:41:8f
Exponent: 65537 (0x10001)

Certificate Signature Algorithm: md5WithRSAEncryption Certificate

Certificate Signature:
93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22: 68:9f

Some web examples

- Secure communication
  - (http:// →) https://www.elcorteingles.es
  - http://www.catcert.cat → https://epscd.aoc.cat/ca/
    vs. https://www.catcert.cat
  - https://www.sede.fnmt.gob.es (Firefox vs. Chrome)
- Web identification
- Certificates

Security

- Public Key Infrastructure (PKI)
  - Basic concepts
  - Validation protocols
  - Trust models
  - X.509 certificates
  - **PKCS**

PKCS - basic ideas

- **Public Key Cryptography Standard**
- RSA Labs. Since 1991.
- Numbered rules (#1 .. #15, not all numbers)
- Example:
  PKCS#1 RSA cryptography standard (v2.2, 2012):
  primitives, encryption and signature schemas,
  ASN.1 for keys, …
  (a kind of "implementation guidelines")

PKCS – standards

#1: RSA Cryptography

#3: Diffie-Hellman Key Agreement

#5: Password-Based Cryptography (*RFC 2898, 2000*)

#6: Extended-Certificate Syntax (*obsoleted by X.509 v3*)

#7: Cryptographic Message Syntax (*RFC 2315, 1998*)

#8: Private-Key Information Syntax (*RFC 5208, 2008*)

#9: Selected Attribute Types (*RFC 2985, 2000*)

#10: Certification Request Syntax (*RFC 2986, 2000*)

#11: Cryptographic Token Interface

#12: Personal Information Exchange Syntax

#13: Elliptic Curve Cryptography (*proposal since 1998!*)

#15: Cryptographic Token Information Format

PKCS – standards

#1: RSA Cryptography

#3: Diffie-Hellman Key Agreement

#5: Password-Based Cryptography (*RFC 2898, 2000*)

#6: Extended-Certificate Syntax (*obsoleted by X.509 v3*)

#7: Cryptographic Message Syntax (*RFC 2315, 1998*)

#8: Private-Key Information Syntax (*RFC 5208, 2008*)

#9: Selected Attribute Types (*RFC 2985, 2000*)

#10: Certification Request Syntax (*RFC 2986, 2000*)

#11: Cryptographic Token Interface

#12: Personal Information Exchange Syntax

#13: Elliptic Curve Cryptography (*proposal since 1998!*)

#15: Cryptographic Token Information Format

PKCS#7: Cryptographic Message Syntax Standard

- Version 1.5, 1993
- Extensions and Revisions (towards 1.6), 1997
- RFC 2315, 1998
- **Defines syntax for encrypted and/or signed messages.**
- Based on PEM (Privacy Enhanced Mail).
  Basis of S/MIME.
- Used to sign and/or encrypt messages under a PKI, for certificate dissemination, for single sign-on.

PKCS#7: Cryptographic Message Syntax Standard

- Obsoleted by RFC5652 (after others), 2009:

Cryptographic Message Syntax (CMS)

- CMS syntax used to:

digitally sign, digest, authenticate, or encrypt arbitrary message content

- Describes encapsulation syntax for data protection

- Uses ASN.1

sign-on.

PKCS#7: Cryptographic Message Syntax Standard

- Obsoleted by RFC5652 (after others), 2009:

Cryptographic Message Syntax (CMS)

- Updated by RFC 8933 (2020):

“Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection”

sign on.

PKCS#7

- Content types:
  - data
  - signedData:
    - **Content + Encrypted message digest**
  - envelopedData:
    - **Content encrypted** (with symmetric key) **+ (content-)encryption key encrypted** (with asym. key)
  - signedAndEnvelopedData:
    - **envelopedData +**
    - **Doubly encrypted message digest** (i.e., signature encrypted with content key)
  - digestedData
  - encryptedData

PKCS#7 → CMS (RFC5652)

Content types

- Data
- SignedData
- EnvelopedData
- DigestedData
- EncryptedData
- AuthenticatedData
- …

PKCS#7 → CMS (RFC 5652)

Content types

- Data
- SignedData
- EnvelopedData
- DigestedData
- EncryptedD
- Authenti a
- …

SignedAndEnvelopedData
dropped

PKCS#7 → CMS (RFC 5652)

Content types

- Data
- SignedData
- EnvelopedData
- DigestedData
- EncryptedD
- Authenti ta
- …

SignedAndEnvelopedData
dropped

AuthEnvelopedData
added by RFC 5083

PKCS#7 → CMS (RFC5652)

- SignedData

```
SignedData ::= SEQUENCE {
   version CMSVersion,
   digestAlgorithms DigestAlgorithmIdentifiers,
   encapContentInfo EncapsulatedContentInfo,
   certificates [0] IMPLICIT CertificateSet OPTIONAL,
   crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
   signerInfos SignerInfos }
```

- EnvelopedData

```
EnvelopedData ::= SEQUENCE {
  version CMSVersion,
  originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos RecipientInfos,
  encryptedContentInfo EncryptedContentInfo,
  unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL
}
```