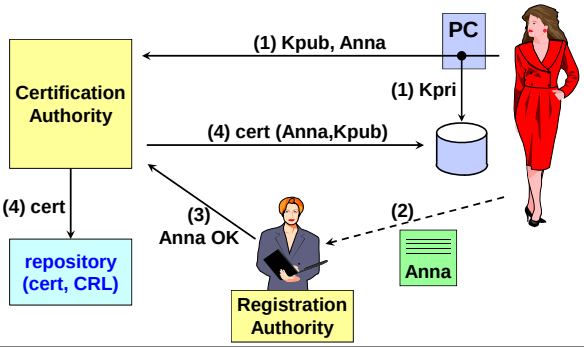
The X.509 standard, PKI and electronic documents

Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dipartimento di Automatica e Informatica

Certification Authority



X.509 certificates

- **standard ITU-T X.509:**
  - v1 (1988)
  - v2 (1993) = minor
  - v3 (1996) = v2 + extensions + attribute certificate v1
  - v3 (2001) = v3 + attribute certificates v2
- **is part of the standard X.500 for directory services (white pages)**
- **is a solution to the problem of identifying the owner of a cryptographic key**
- **definition in ASN.1 (Abstract Syntax Notation 1)**

X.509 version 3

- **standard completed in June 1996**
- **groups together in a unique document the modifications required to extend the definition of certificate and CRL**
- **two types of extensions:**
  - public, that is defined by the standard and consequently made public to anybody
  - private, unique for a certain user community

Critical extensions

- **an extension can be defined as critical or non critical:**
  - in the verification process the certificates that contain an unrecognized critical extension MUST be rejected
  - a non critical extension MAY be ignored if it is unrecognized
- **the different (above) processing is entirely the responsibility of the party that performs the verification: the Relying Party (RP)**

Public extensions

- **X.509v3 defines four extension classes:**
  - key and policy information
  - certificate subject and certificate issuer attributes
  - certificate path constraints
  - CRL distribution points

Key and policy information

- **authority key identifier**
- **subject key identifier**
- **key usage**
- **private key usage period**
- **certificate policies**
- **policy mappings**

Key and policy information

- **key usage**
  - identifies the application domain for which the public key can be used
  - can be critical or not critical
  - if it is critical then the certificate can be used only for the scopes for which the corresponding option is defined

Key and policy information

- ***key usage* – the applications that can be defined are:**
  - digitalSignature (CA, user)
  - nonRepudiation (user)
  - keyEncipherment (user)
  - dataEncipherment
  - keyAgreement (encipherOnly, decipherOnly)
  - keyCertSign (CA)
  - cRLSign (CA)

Certificate subject and certificate issuer attributes

- **subject alternative name**
- **issuer alternative name**
- **subject directory attributes**

Certificate subject and certificate issuer attributes

- **subject alternative name**
  - allows to use different formalisms to identify the owner of the certificate (e.g. e-mail address, IP address, URL)
  - always critical if the field subject-name is empty

X.509 alternative names

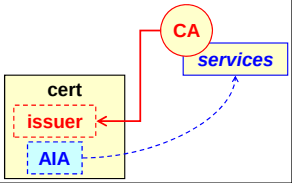
- **various possibilities:**
  - rfc822Name
  - dNSName
  - iPAddress
  - uniformResourceIdentifier
  - directoryName
  - X400Address
  - ediPartyName
  - registeredID
  - otherName

CRL distribution point

- **CRL distribution point**
  - identifies the distribution point of the CRL to be used in validating a certificate
  - can be:
    - directory entry
    - e-mail or URL
  - critical o non critical

PKIX private extensions

- **authority information access**
  - indicates how to access information and services of the CA that issued the certificate:
    - certStatus
    - certRetrieval
    - cAPolicy
    - caCerts
  - critical or not critical



Extended key usage

- **in addition or in substitution of keyUsage**
- **possible values:**
  - (id-pkix.3.1) serverAuth [DS, KE, KA]
  - (id-pkix.3.2) clientAuth [DS, KA]
  - (id-pkix.3.3) codeSigning [DS]
  - (id-pkix.3.4) emailProtection [DS, NR, KE, KA]
  - (id-pkix.3.8) timeStamping [DS, NR]

CRL X.509

- **Certificate Revocation List**
- **list of revoked certificates**
- **CRLs are issued periodically and maintained by the certificate issuers**
- **CRLs are digitally signed:**
  - by the CA that issued the certificates
  - by a revocation authority delegated by the (indirect CRL, iCRL)

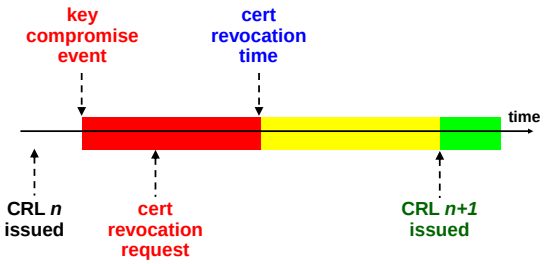
CRL X.509 version 2

```
CertificateList ::= SEQUENCE {
   tbsCertList          TBSCertList,
   signatureAlgorithm   AlgorithmIdentifier,
   signatureValue       BIT STRING }
TBSCertList ::= SEQUENCE {
   version              Version OPTIONAL,
                        -- if present, version must be v2
   signature            AlgorithmIdentifier,
   issuer               Name,
   thisUpdate           Time,
   nextUpdate           Time OPTIONAL,
   revokedCertificates  SEQUENCE {
      userCertificate   CertificateSerialNumber,
      revocationDate    Time,
      crlEntryExtensions Extensions OPTIONAL
   } OPTIONAL,
   crlExtensions        [0] Extensions OPTIONAL
}
```

Extensions of CRLv2

- **crlEntryExtensions:**
  - reason code
  - hold instruction code
  - invalidity date
  - certificate issuer
- **crlExtensions:**
  - authority key identifier
  - issuer alternative name
  - CRL number
  - delta CRL indicator
  - issuing distribution point

Certificate revocation timeline

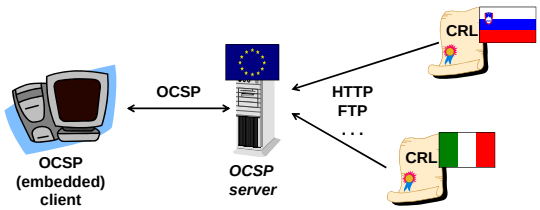


OCSP

- **RFC-2560: On-line Certificate Status Protocol**
- **IETF-PKIX standard to verify online if a certificate is valid:**
  - good
  - revoked
    - revocationTime
    - revocationReason
  - unknown
- **response signed by the server (not by the CA!)**
- **the OCSP server certificate cannot be verified with OCSP itself!**

Architecture of OCSP

- **possible pre-computed responses**
  - decreases the computational load on the server … but makes possible replay attacks!
- **possible to obtain information not from CRL**

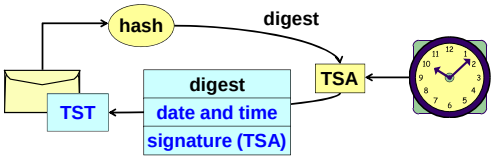


Models of OCSP responder

- **Trusted Responder**
  - the OCSP server signs the responses with a pair key:cert independent of the CA for which it is responding
  - company responder or TTP paid by the users
- **Delegated Responder**
  - the OCSP server signs the responses with a pair key:cert which is (can be) different based on the CA for which it is responding
  - TTP paid by the CA

Time-stamping

- **proof of creation of data before a certain point in time**
- **TSA (Time-Stamping Authority)**
- **RFC-3161:**
  - request protocol (TSP, Time-Stamp Protocol)
  - format of the proof (TST, Time-Stamp Token)

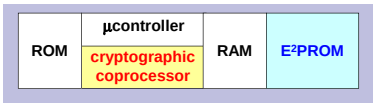


PSE (Personal Security Environment)

- **each user should protect:**
  - his own private key (secret!)
  - the certificates of the trusted root CAs (authentic!)
- **software PSE:**
  - (encrypted) file of the private key
- **hardware PSE:**
  - passive = protected keys (same as sw PSE)
  - active = protected keys + crypto operations
- **mobility is possible in both cases (but with problems)**

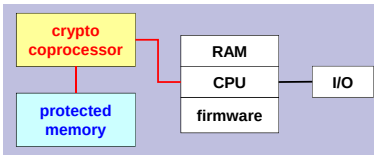
Cryptographic smart-card

- **chip cards with memory and/or autonomous cryptographic capacity**
- **simple: DES**
- **complex: RSA**
  - length of the key?
  - generation of the private key on board?
- **few memory (EEPROM): 4 - 32 Kbyte**



HSM (HW Security Module)

- **cryptographic accelerator for servers**
  - secure storage of private key
  - autonomous encryption capabilities (RSA, sometimes symmetric algorithms too)
- **form factor: PCI board or external device (USB, IP, SCSI, …)**



Security API (low level)

- **PKCS-11 = (only) crypto engine**
  - in software
  - in hardware
    - smart card
    - cryptographic card
  - part of the CDSA architecture
- **MS-CAPI CSP (Crypto Service Provider)**
  - same functions as PKCS-11 but proprietary API of MS

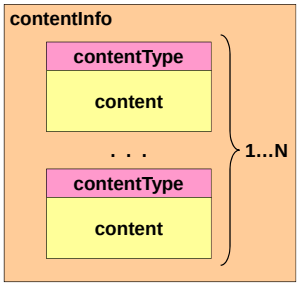
Secure data formats

- **PKCS-7 = secure envelope**
  - signed and/or encrypted
- **PKCS-10 = certificate request**
  - used in the communication among the client and CA / RA
- **PKCS-12 = software PSE (Personal Security Environment)**
  - transport of keys and certificates
- **are not application formats:**
  - S/MIME? IDUP-GSS-API? XML-DSIG?
  - legal electronic documents?

PKCS-7 and CMS formats

- **cryptographic message syntax**
- **PKCS-7 is the RSA standard for secure envelope (v1.5 is also RFC-2315)**
- **CMS is the evolution of PKCS-7 inside IETF, numbered as RFC-2630**
- **allows signing and/or encryption of data, with symmetric or asymmetric algorithms**
- **allows to put more signatures on the same object (hierarchical or parallel)**
- **can include the certificates used for the signature**
- **is a recursive format**

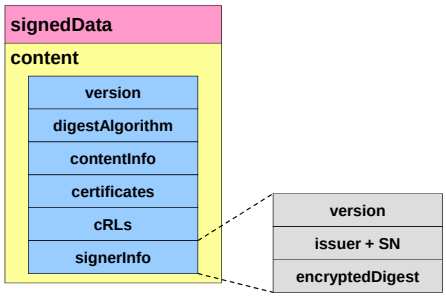
PKCS-7: structure



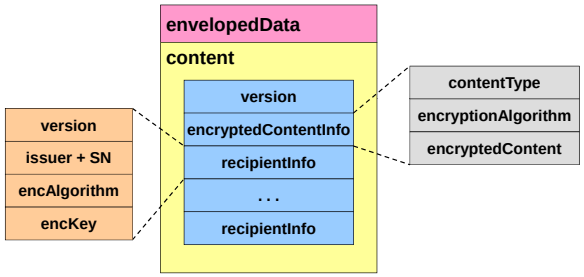
PKCS-7: contentType

- *data*
  encoding of a generic sequence of bytes
- *signedData*
  data + parallel digital signatures (1..N)
- *envelopedData*
  data encrypted symm. + key encrypted with RSA
- *signedAndEnvelopedData*
  RSA encryption of (data + digital signatures)
- *digestData*
  data + digest
- *encryptedData*
  data encrypted with a symmetric algorithm

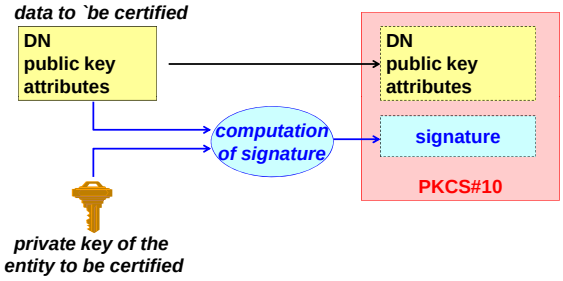
PKCS-7: signedData



PKCS-7: envelopedData

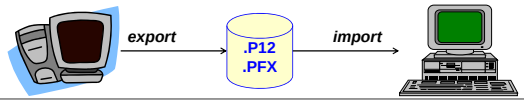


PKCS-10

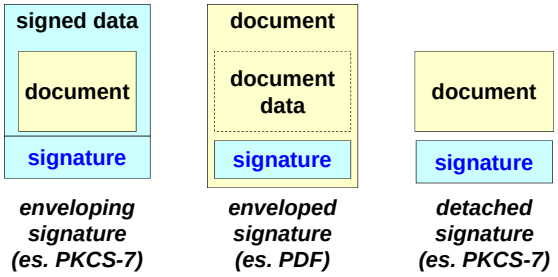


PKCS-12 format (security bag)

- transport of (personal) cryptographic material among applications / different systems
- transports a private key and one or more certificates
- transports the digital identity of a user
- used by Netscape, Microsoft, Lotus, ...
- criticized from the technical point of view (especially in the MS implementation) but widely used



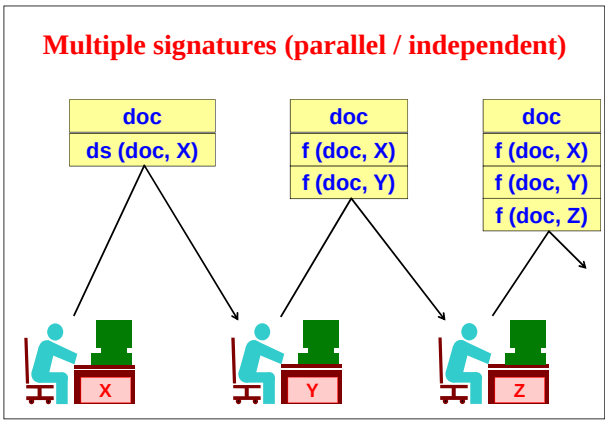
Formats of signed documents

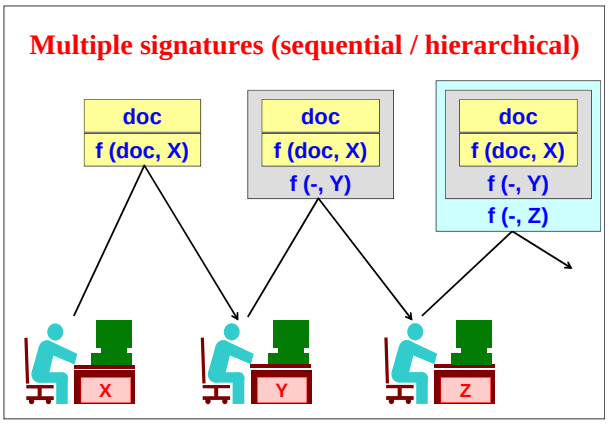


enveloping signature (es. PKCS-7)

enveloped signature (es. PDF)

detached signature (es. PKCS-7)


Multiple signatures (parallel / independent)
doc
ds (doc, X)
doc
f (doc, X)
f (doc, Y)
doc
f (doc, X)
f (doc, Y)
f (doc, Z)
X
Y
Z


Multiple signatures (sequential / hierarchical)
doc
f (doc, X)
doc
f (doc, X)
f (-, Y)
doc
f (doc, X)
f (-, Y)
f (-, Z)
X
Y
Z