DI-FCT-UNL
Segurança de Redes e Sistemas de Computadores
Network and Computer Systems Security

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
2º Semestre, 2018/2019

- Public Key Crypto and Key Management
- X509
- PKI (Public Key Infrastructure)

Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Problem… Public Key Crypto requires trusted key management

- **Generation of keypairs**: controlled by the the owners
- **Careful confinement and/or use in secure environments**
  - **Private Keys**: must be managed and maintained in a secure way (this included the control of exposition)
    - Memory operations (exposed in memory, ex., during digital signatures)
    - Secure Storage (encrypted)
    - Ex., Keystores – Protected by PBE and/or Symmetric Encryption
    - Better: stored and processed in "devices" or "appliances" where it may be impossible (or unlikely) the access by third parties
  - **Public keys**: can be distributed, disseminated and publicly disclosed
    - But the trusted association to the correct principals is an issue
    - Validation requires a trusted verification of such association
- Another issue: management of keys require the use of **standardized and interoperable representation formats**

Management with Smartcards

- Smartcards and Smartcard Readers
- Standardized interfaces: ISO 7816
- Contact (ex., Portuguese Citizen Card) or contactless (ex., Portuguese Passport)

Use of Smartcards

- Use of smartcards:
  - Interface (via reader) by sending commands / receiving results: APDUs or App. Protocol Data Units)
    - APDUs are standardized messages
  - Note: APDUs are standardized structures but the content may be different and dependent from specific implementations (smartcard manufacturers and variety of implementations and programming support)
    - Applications don't use directly (in general) APDUs (considered a low level abstraction)

  - Applications use more high-level abstractions or programming interfaces, providing standardized generic primitives allowing the manipulation of objects in the smartcard, as well as, cryptographic and key-management operations
    - Examples:
      - PKCS#11 (defined by the RSA Labs)
      - Microsoft CryptoAPI (Cryptographic Application Programming Interface)

PKCS#11 (aka, Cryptoki)

- Cryptoki: Cryptographic Token Interface
  - Provides an "uniform logic view" of physical devices (such as a smartcard) regarded as a "cryptographic token"
  - Implements an Object-Oriented Interface, through Middleware (libraries) provided by manufacturers
    - This is for example the case of the Portuguese Citizen Card and compatible Readers
    - In general a PKCS#11 middleware can be adopted by generic applications designed to support smartcards
      - Ex., Email User Agents, Browsers, etc.
      - Ex., Firefox (see Privacy and Security)

See https://en.wikipedia.org/wiki/PKCS_11 for more details

PKCS#11 in Java

- There is a Sun PKCS#11 Provider for Java JCA/JCE: can be used since the Java 5 (J2SE 5.0)
- In contrast to most other providers, it does not implement cryptographic algorithms itself. Instead, it acts as a bridge between the Java JCA and JCE APIs and the native PKCS#11 cryptographic API, translating the calls and conventions between the two.
- This means that Java applications calling standard JCA and JCE APIs **can, without modification, take advantage of algorithms offered by the underlying PKCS#11 implementations, such as, for example,**
  - Cryptographic Smartcards,
  - Hardware cryptographic accelerators, and
  - High performance software implementations.

PKCS#11 in Java

- A Java PKCS#11 Crypto Provider is installed as any other crypto provider

```
…
# configuration for security providers 1-9 omitted
security.provider10=sun.security.pkcs11.SunPKCS11 /opt/bar/cfg/pkcs11.cfg
```

See more in:
https://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html

Microsoft CryptoAPI (aka CAPI)

- High-Level Middleware Integration, including Smartcard interoperability for MS Windows
  - Architecture based on a generic module (providing an external API) and specific CSP (Cryptographic Service Providers), each one provided for specific physical devices
  - One CSP can or cannot use the PKCS#11 definition for specific smartcards

See https://en.wikipedia.org/wiki/Microsoft_CryptoAPI for details

Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X.509 standardization

- X509 is a standard framework, part of the ITU-T X500 standardization effort:
  - X509 focused on the provision of authentication services by the X500 directory service
  - Standard representation of certificates (formats) and their attributes and data types, as well as recommended cryptography (algorithms and parameters)
    - Encoding Standardization
  - Framework to address PKI systems (processes, entity roles, interfaces)
  - Life cycle of certificates: generation, enrollment, certification and validation
  - Initial approach: 1988 , 1993 (v1), 1995 (v2), 2000 (v3)

X509 Certificates - Life Cycle

1. Certificate Generation Phase (CSR)

User generates keypairs and a certificate request using different tools/sw available

- Keep safe the private key
- Generation of a certificate in a CSR format

CSR encoding is sent to the CA for a registration/ enrollment process

2. Registration (or enrollment) Phase

CA receives the CSR and decides on specific verification process and policies

3. Certification or issuing Phase

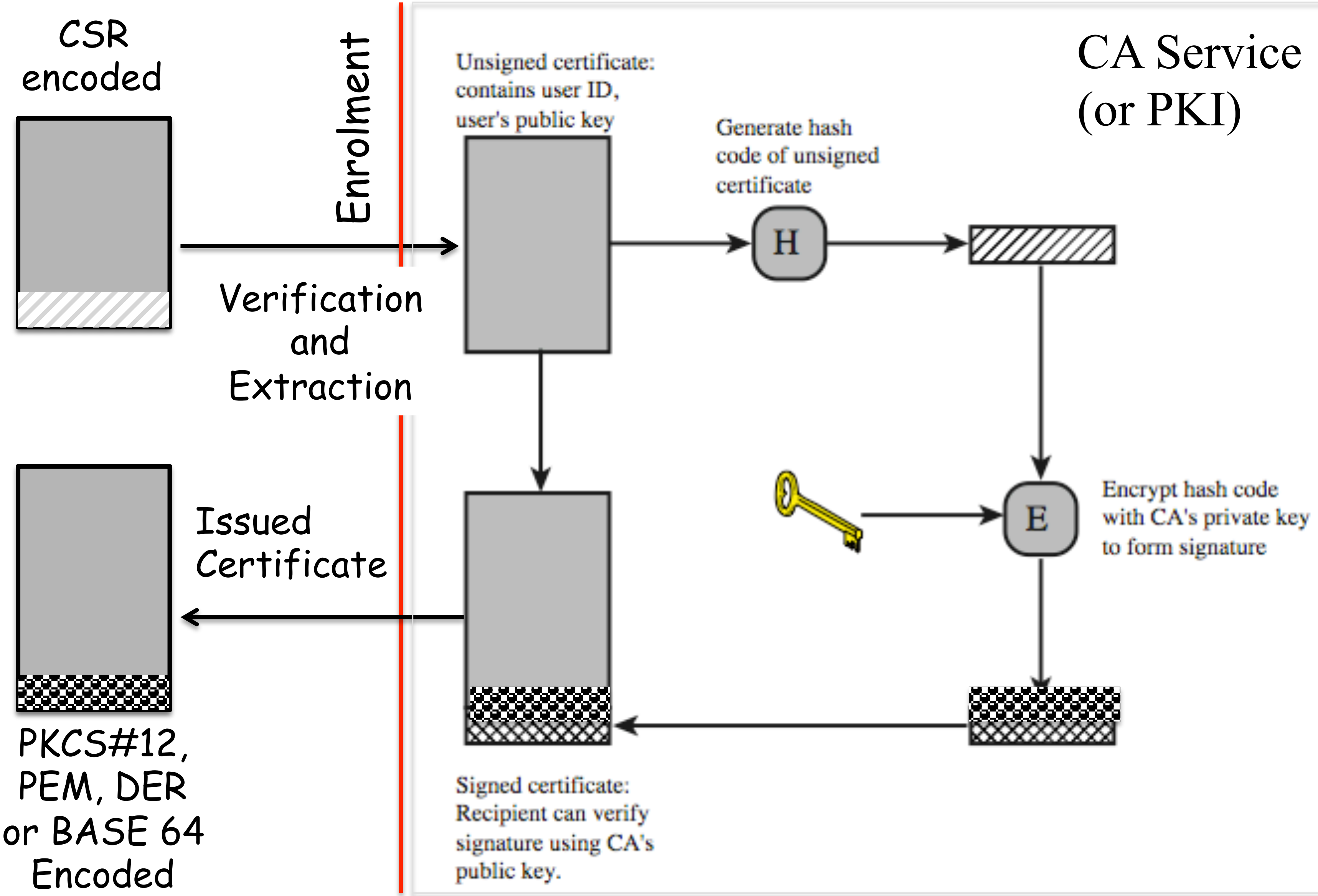
X509 certificate is issued (signed by the CA) according to a possible hierarchy chain inside the CA

4. Distribution Phase

The issued X509 certificate is provided by the CA in a directory service (Certification Repository Service)

User (or any other user) obtains the certificate and is ready to use, Together with the certification chain

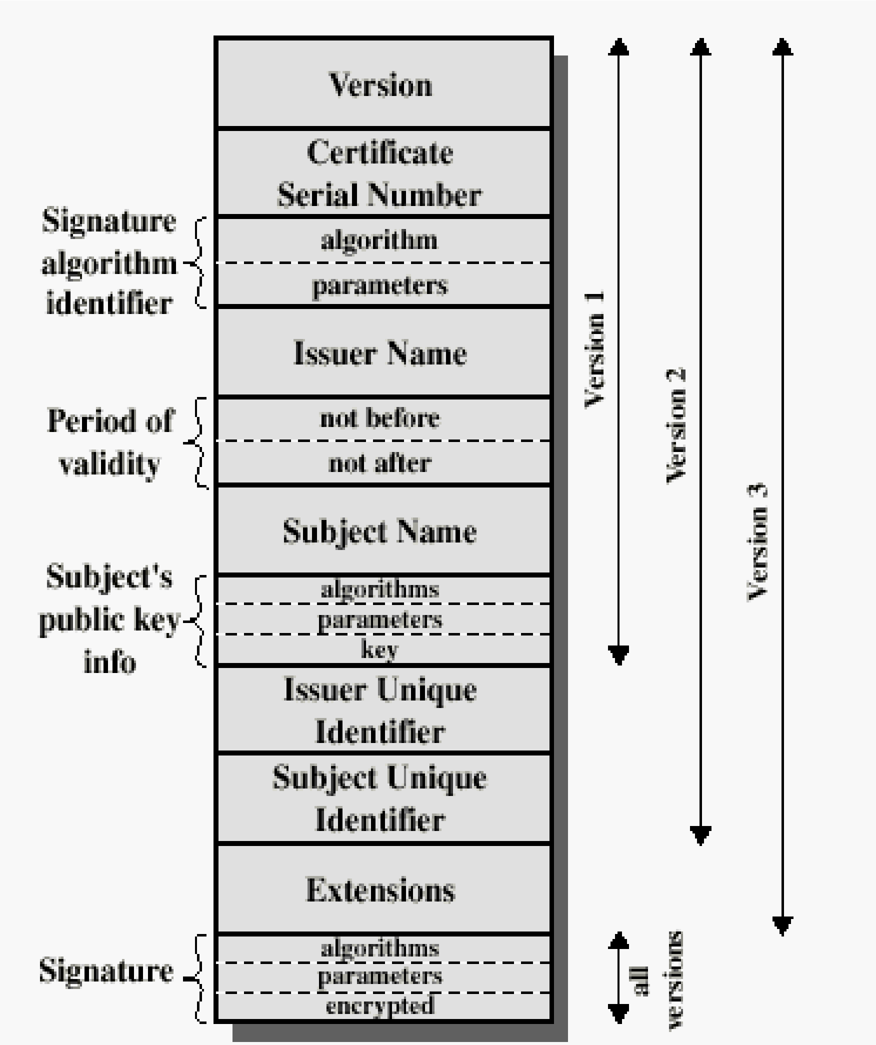
Extraction and Signing (Issuing)



Outline

- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X.509 Certificate and CRL Formats



X509 certificate (atttributes in different versions)

Notation:

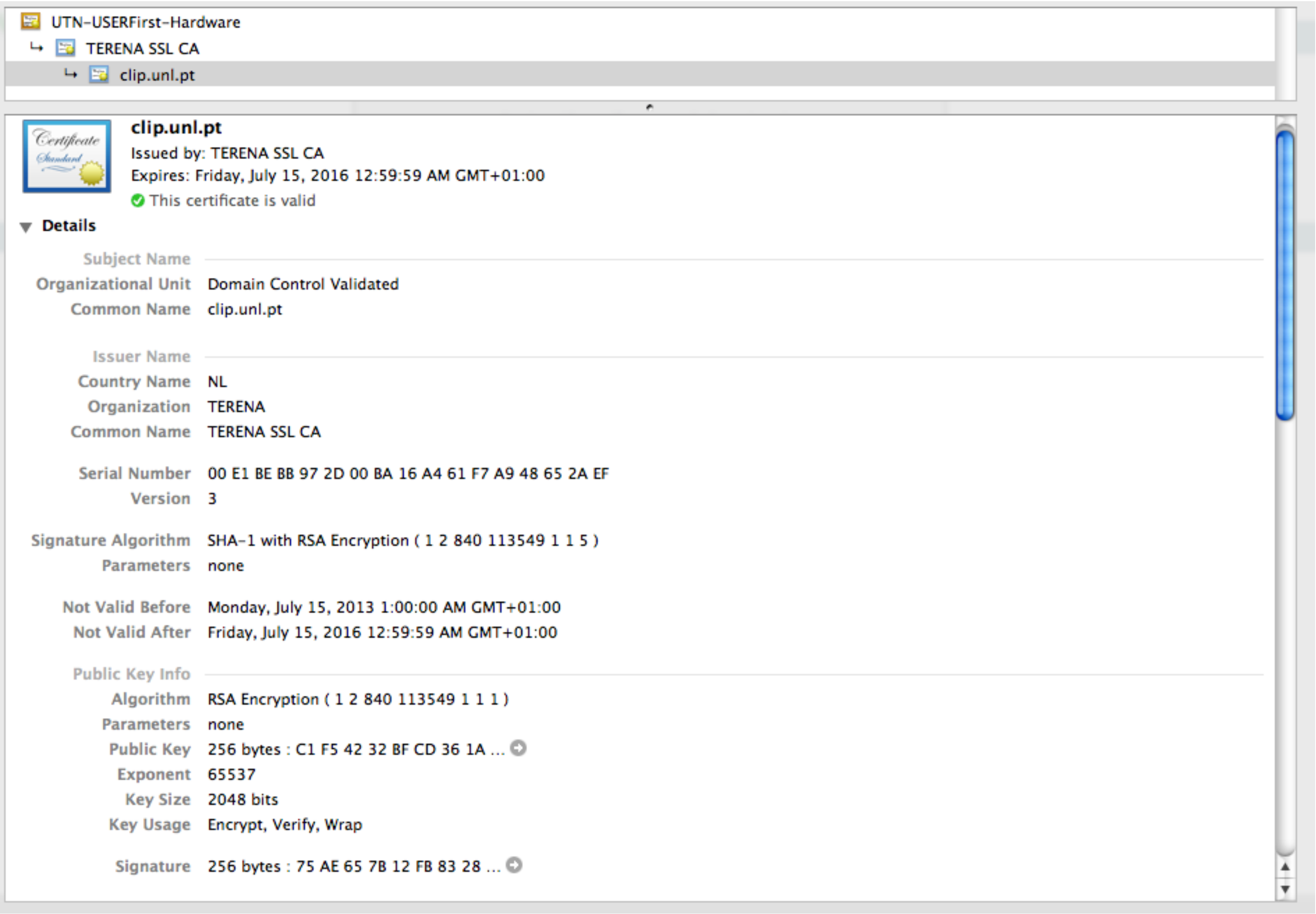
CA <<A>> =
{A, V, SN, AI, CA, TA, KpubA}$_{SigCA}$

X.509 Certificate

- **See fields and attributes in current X509v3 Certificates**

- Each certificate contains:
  - The public key of a distinguished subject name (principal, user)
    - Subject name, Subject's public key information fields
  - Other attributes with additional information as a list of other (field, value) pairs
    - Issuer UID, serial number, version, validity information, relevant information of cipher-suites used, verification control information, several extensions and fingerprints
  - Signed with the private key of a CA.
    - Digital signature covering all the other fields
      - Hash of fields, signed with the CA private key

Example of a current X509v3 Certificate



UTN-USERFirst-Hardware
↳ TERENA SSL CA
↳ clip.unl.pt

clip.unl.pt
Issued by: TERENA SSL CA
Expires: Friday, July 15, 2016 12:59:59 AM GMT+01:00
This certificate is valid

Details

Subject Name	
Organizational Unit	Domain Control Validated
Common Name	clip.unl.pt
Issuer Name	
Country Name	NL
Organization	TERENA
Common Name	TERENA SSL CA
Serial Number	00 E1 BE BB 97 2D 00 BA 16 A4 61 F7 A9 48 65 2A EF
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1 2 840 113549 1 1 5)
Parameters	none
Not Valid Before	Monday, July 15, 2013 1:00:00 AM GMT+01:00
Not Valid After	Friday, July 15, 2016 12:59:59 AM GMT+01:00
Public Key Info	
Algorithm	RSA Encryption (1 2 840 113549 1 1 1)
Parameters	none
Public Key	256 bytes : C1 F5 42 32 BF CD 36 1A ...
Exponent	65537
Key Size	2048 bits
Key Usage	Encrypt, Verify, Wrap
Signature	256 bytes : 75 AE 65 7B 12 FB 83 28 ...

Obtaining a User's Certificate

- Certificates: issued by CAs:
  - Any user with access to the public key of the CA can recover and validate the user public key that was certified (by a direct or reverse trust certification chain verification)
  - Users can exchange certificates and certification chains for verification
  - No part other than the CA can issue and modify the certificate, without this being detected.
    - Certificates are unforgeable. So it is possible to send/distribute them in protocols or place them in public directories or repositories

Outline

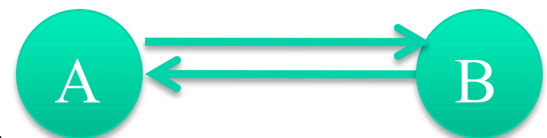
- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Authentication Procedures

One-way authentication and Key dist.

$A[\{t_a, r_a, Id_B\}K_{ab},\ signData,\ \{K_{ab}\}K_{pubB}]$

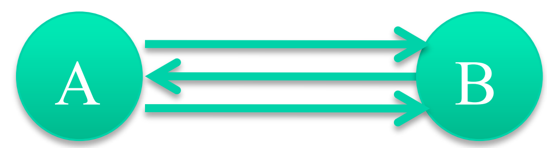
Two-way (mutual) authentication and Key dist.



$A\ [\{t_a, r_a, Id_B\}K_{ab},\ signData,\ \{K_{ab}\}K_{pubB}]$

$B\ [\{t_b, r_b, Id_A\}K_{ba},\ signData,\ \{K_{ba}\}K_{pubA}]$

Three-way (Mutual) authentication And Key Dist.



$A[\{t_a, r_a, Id_B\}K_{ab},\ signData,\ \{K_{ab}\}K_{pubB}]$

$B[\{t_b, r_b, Id_A\}K_{ba},\ signData,\ \{K_{ba}\}K_{pubA}]$

$A\{r_b\}$

One-Way Authentication

- 1st message (A->B) used to establish:
  - the authenticated identity of A and that message is from A
  - that the message was intended for B
  - integrity & originality of message
- Message must include timestamp, nonce, B's identity and is signed by A
- May include additional info for B
  - Eg., session key, for implicit key-establishment (session key-envelope)
    - Allows the concatenation of additional confidential content or messaging

Two-Way Authentication

- 2 messages (A->B, B->A) which also establishes in addition to "one-way":
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply

- Reply includes original nonce from A, also timestamp and nonce from B

- May include additional info for A
  - May establish "half-duplex" session symmetric keys
  - May establish "full-duplex" session symmetric keys (generated from pre-master keys or exchanged seed-material)

Three-Way Authentication

- 3 messages (A->B, B->A, A->B), adding a final round to mutual authentication
  - Enables above authentication **without dependency from synchronized clocks**

- Has reply from A back to B containing signed copy of nonce iterated from B
  - means that timestamps need not be checked or relied upon, preserving anyway message-freshness and ordering (protocol termination) control (no dependency of sync. clocks)

Authentication Procedures (usage)

Autenticação one-way model:

Ex., One-Way TLS Authentication, S/MIME or PGP Message Authentication

Autenticação two-way (mutual)

Ex., Two-Way TLS Authentication, SET Protocol

Autenticação three-way (mutual)

Ex., Two-Way TLS Authentication and Key-Session Generation and Agreement

Practical protocols

Two forms of management of chain trust

Certificates pre-cached (and managed orthogonally) in trusted certificate stores

Ex., JAVA, keystores

> Advantages ? Drawbacks ?

"On the Fly" validation of trust chains

- Only need "root" certificate pre-cached in trusted stores
- Send certification chains in the authentication handshake

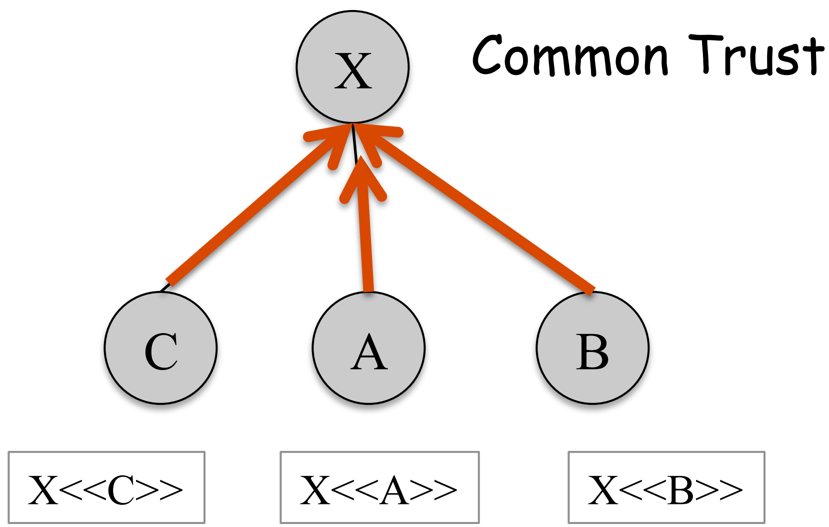
> Advantages ? Drawbacks ?

Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Trust and validation chains

- **Common trust based Validation**
  - When all users subscribe to the same CA
  - Ex., Model for a small community of users (non-scalable, centralized-root trust)
  - Any user A transmits directly the certificate to any other
    - Message authentication with digital signatures
    - Key-distribution protocols



Trust and validation chains

- **No common trust verification conditions**
  - Model for a large community of users (scalable model)
  - Problem: Users need to have Public Keys of all the CAs

  - It may be more practical to consider that
    - There will be several CAs,
    - But each of which securely provides its public key to some fraction of the users
    - Additionally, we can use cross-certification links in a certification hierarchy

Notation

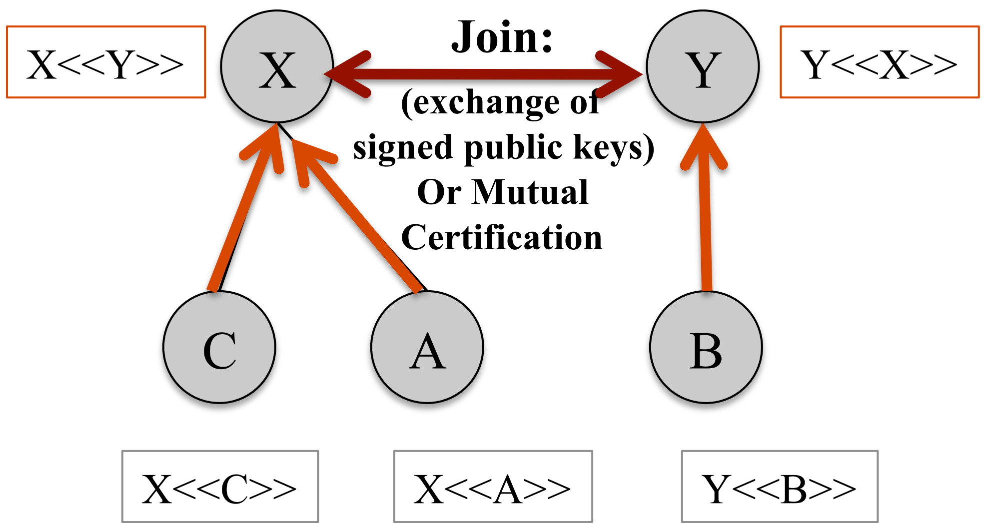
- Notation:
  $CA \ll A \gg = \{A, V, SN, AI, CA, T_A, KpubA\}_{SigCA}$

  Y<<X>>
  Certificate of entity X issued by the CA Y

  Verification of certificates => imply that the verifiers previously obtained, in a trusted way, the CA public key
  - Or trust based on Certification Chains

Solution for no common trust

No Common Trust



- A obtains **X<<Y>>** from a directory
- A obtains **Y<<B>>** from a directory (or directly from B)
- A uses the chain **Y <<B>>, X<<Y>>**
  B can use the chain: **X<<A>> Y<<X>>**

  or reverse chain X<<A>> X<<Y>

- Possible generalization for long paths (when joins are at higher levels)

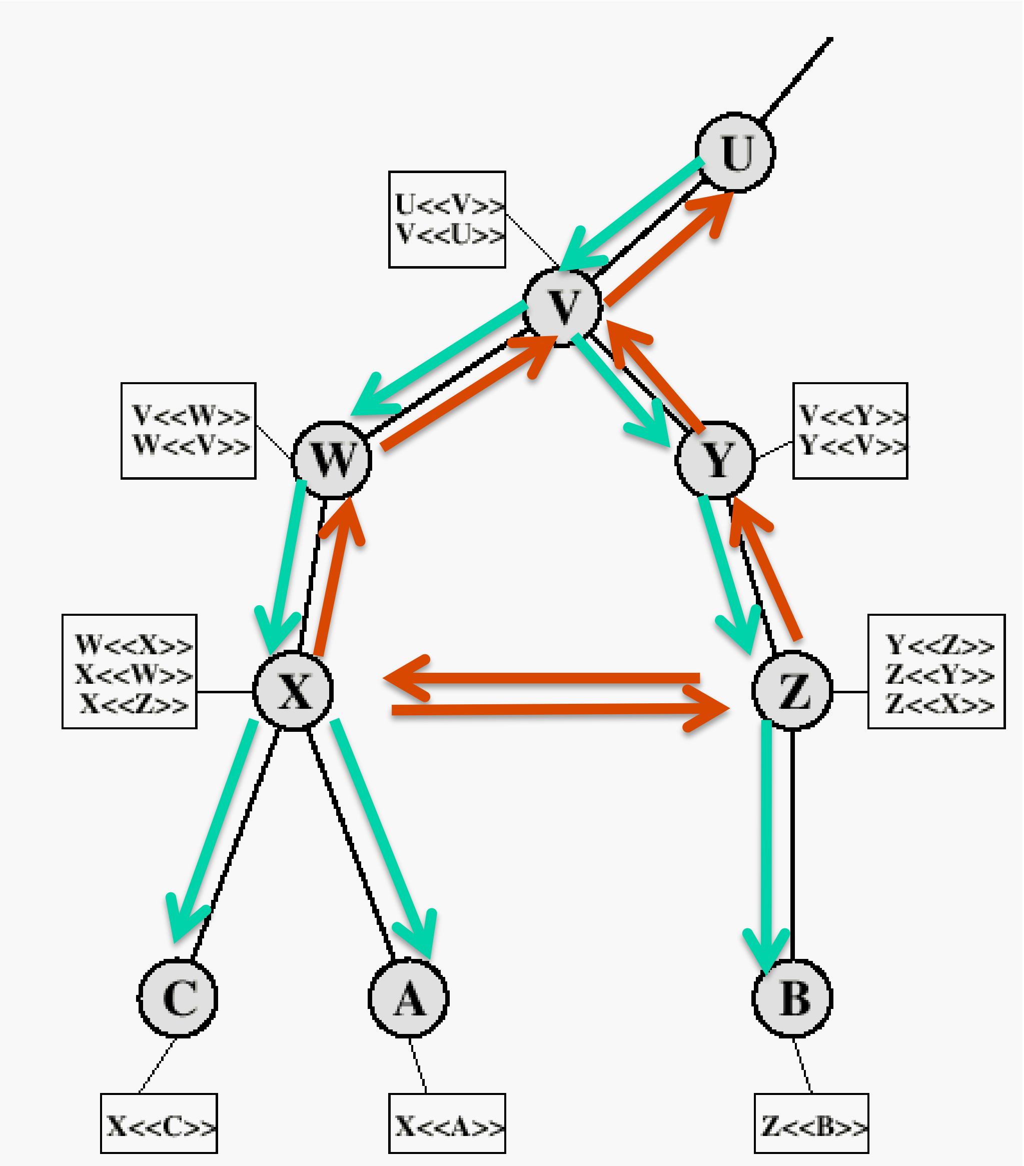
X.509 CA Hierarchy and Chains

- Forward certificates

Forward Chain Validation

- Reverse certificates

Reverse Chain Validation



See a X509v3 Direct Certification Chain in a TLS (HTTPS) connection

- In general the more common is to have Root CA Public Key certificates in local trusted stores, and the authentication processing supported with a direct certification chain validation

- Ex., see the CA's Root Certificates in your Java installation

- See the certificatioj chain in a TLS (HTTPS) connection:
  - With your Browser
  - Using openssl
    - openssl s_client -connect www.feistyduck.com:443

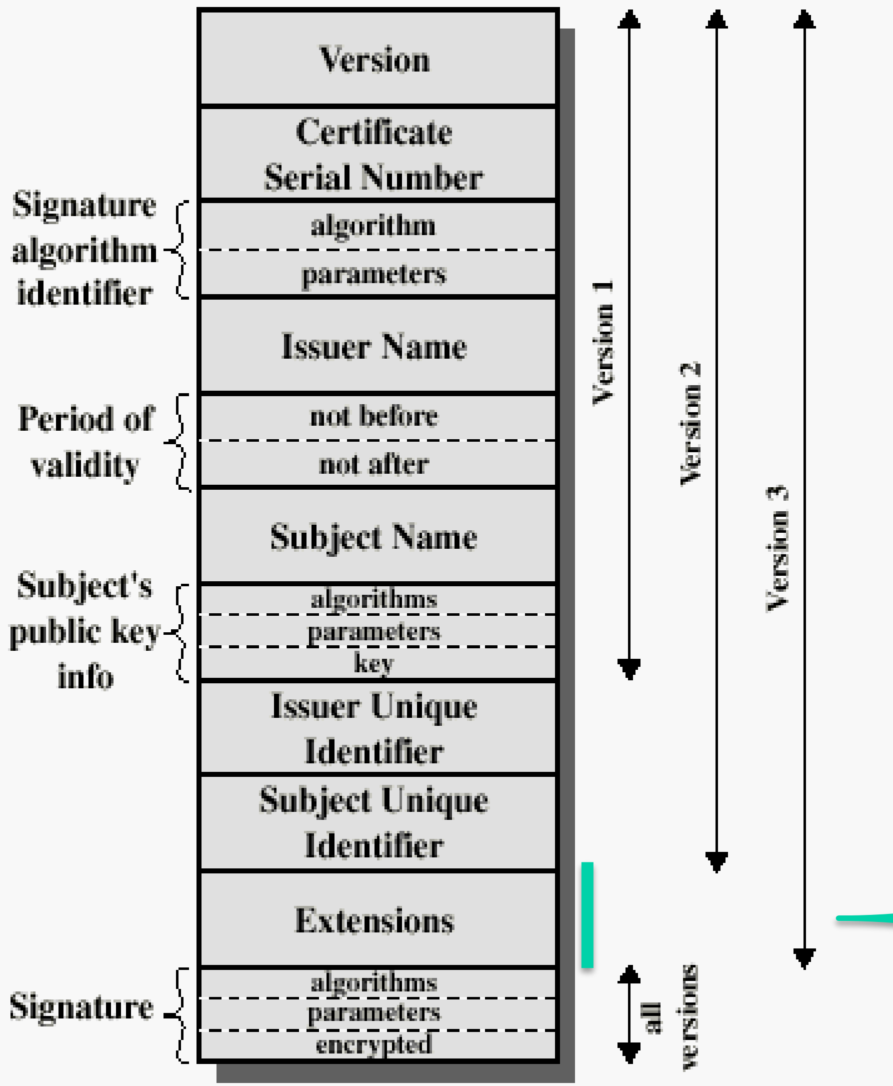
Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Outline

- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X.509 Certificate and CRL Formats



X509 certificate (versions and attributes)

A set of one or more Extension Fields:

- Key Usage
- Constraints
- Extended Key Usage
- Subject Key Identifier
- Authority Key Identifier
- Subject Alt. Names
- Certificate Policies
- CRL Dist. Endppoints
- ESCT List
- Certificate Authority Information ACcess

X509v3 Validation

Other validation issues of certificates for specific validation requirements

- **Subject Name** (fields and attributes)
  - Not only abstract UIDs, URIs, URLs, eMail addresses, …
  - Extended with X500 distinguished name attributes and classification categories as well as alternative names
- **Issuer name**
  - Issuer/CA Distinguished names with X500 attributes
- **Certif. policies, policy mappings and key policies**
  - Allowing for specific validation to a given policy
  - Setting constraints for limitation/contention of the damage from faulty or malicious Cas

X509v3

Other validation issues of certificates for specific validation requirements

- **Inclusion of KeyIDs for Subject and Authority,as Key Selectors**
- **Information on CRL distribution points or for OnLine Status verification points (OCSP) from CA issuers**
- **Gradual adoption of OID standardization**

- **Fingerprints with Dual Secure Hashing Functions for Integrity:**
  - Current use of SHA-256 and SHA-1

Extended validation (EV) Certificates

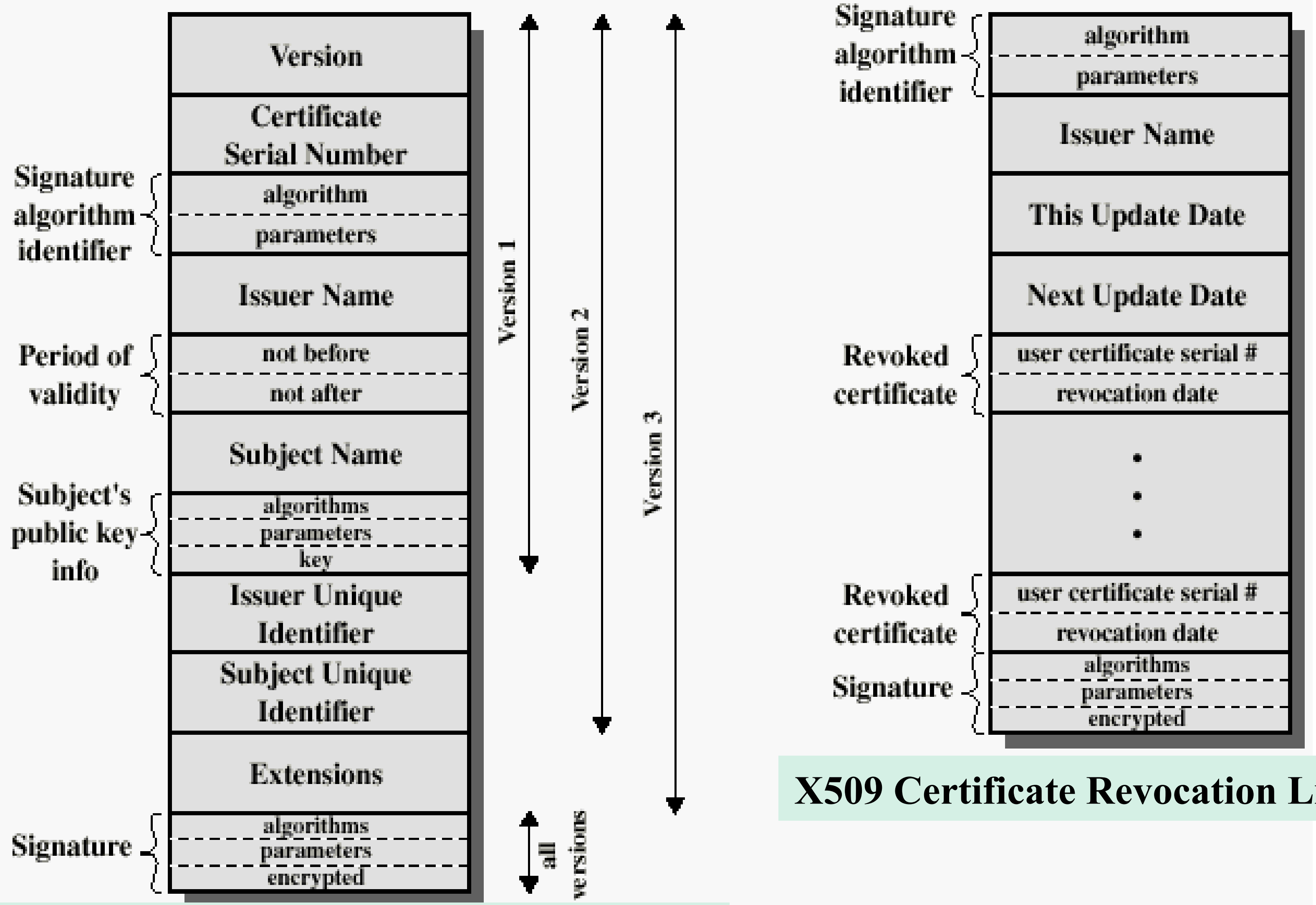
Relevance of
The CA/RA
Delegation Models

- Introduced by the CA/Browser forum
  - http://www.cabforum.org/, http://en.wikipedia.org/wiki/Extended_Validation_Certificate
  - CAs + Relying Party Application Software Suppliers
- Objective: inclusion of standardized procedures for verifying and expressing awareness of the certificate holder and validity (initially motivated by SSL certificates)
- Additional layer of protection: promotion of good practice, guidelines, accurate verification processes for issuing X509v3 SSL certificates
  - **Verifying the legal, physical and operational existence of the entity**
  - **Verifying that the identity of the entity matches official records**
  - **Verifying that the entity has exclusive right to use the domain specified in the EV Certificate**
  - **Verifying that the entity has properly authorized the issuance of the EV Certificate**

Outline

- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X509 Certificates and CRLs



X509 Certificate Revocation List

X509 certificate (fields in different versions)

Revocation of Certificates: why ? when ?

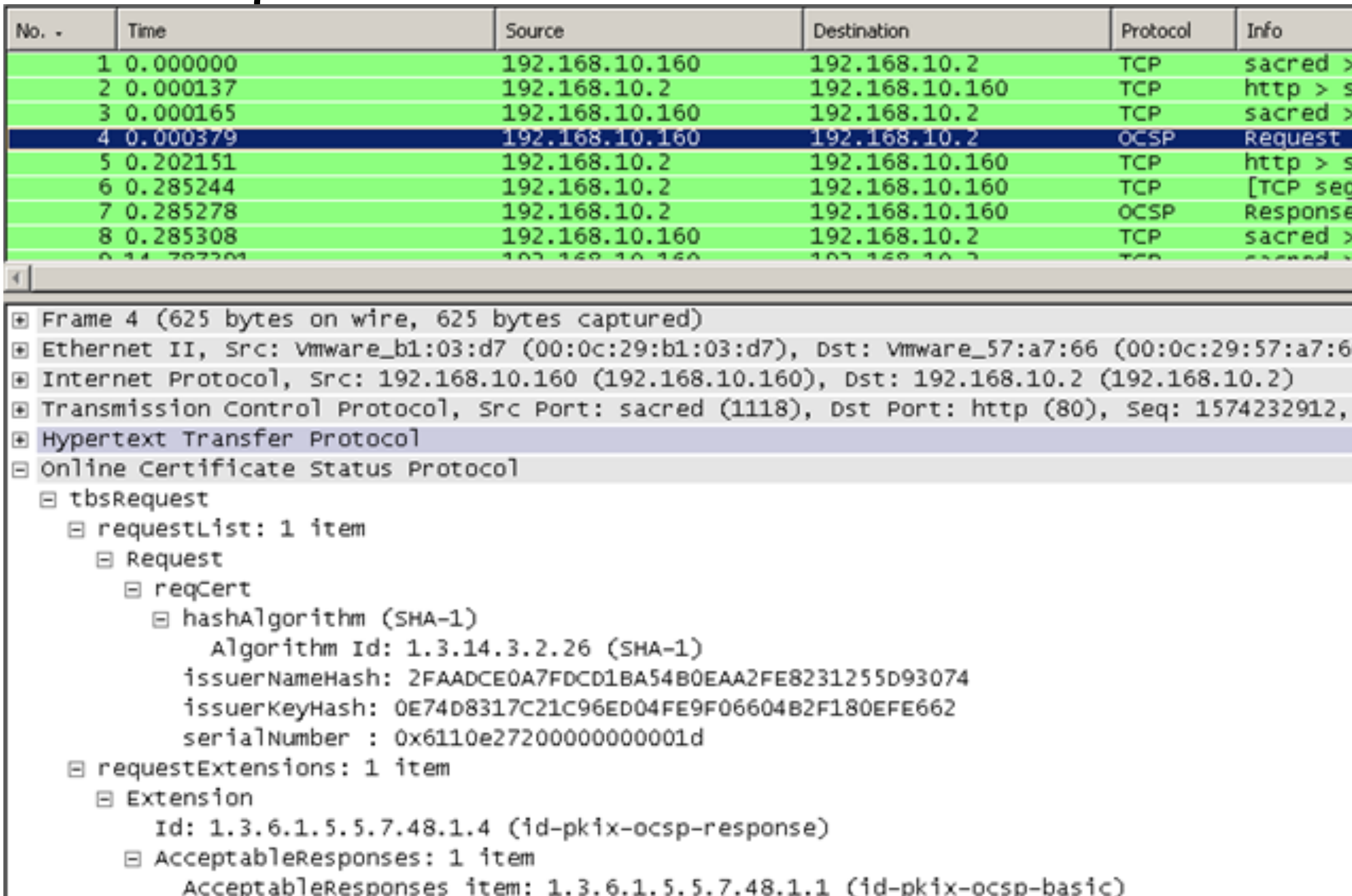
- **Reasons for revocation:**
  - User's private key is assumed to be compromised.
  - User is no longer certified by this CA.
  - CA's certificate is assumed to be compromised.
    - CA's private keys compromised
- **Certificates should not be validated**
  - After the expiration
    - Requires the issuing of a new certificate just before the expiration of the old one
    - The new certificate can be issued by a different CA
  - If the end use is not according with the content (policies, information extensions)
  - If it is in a "current" certification revocation list (CRL) issued by the CA that issued the certificate
  - If not validated in a synchronous "on line" verification process

Management of CRLs

- Maintained by each CA (or CRL delegation end-points)
  - As a list of revoked (but not expired) certificates issued by that CA, including
    - End-user certificates, Possible reverse certificates
- CRLs Managed by the final users (end-user responsibility)
  - Checked from a directory, every time a certificate is received
    - Supported by OnLine Revocation Protocol
    - CRL endpoint implementing the OCSP protocol
  - Checked from a local cache, periodically updated (ex., Incremental, Time-Controlled, Serial Number Controlled)
    - White Lists: White CRLs
    - Black Lists: CRLs
    - Full-Lists vs. Incremental Lists
    - Time-controlled vs. Version-Controlled

OCSP - Online Certificate Status Protocol

- A Request/Response Protocol, usually supported in HTTP
  - **OCSP Request**



OCSP - Online Certificate Status Protocol

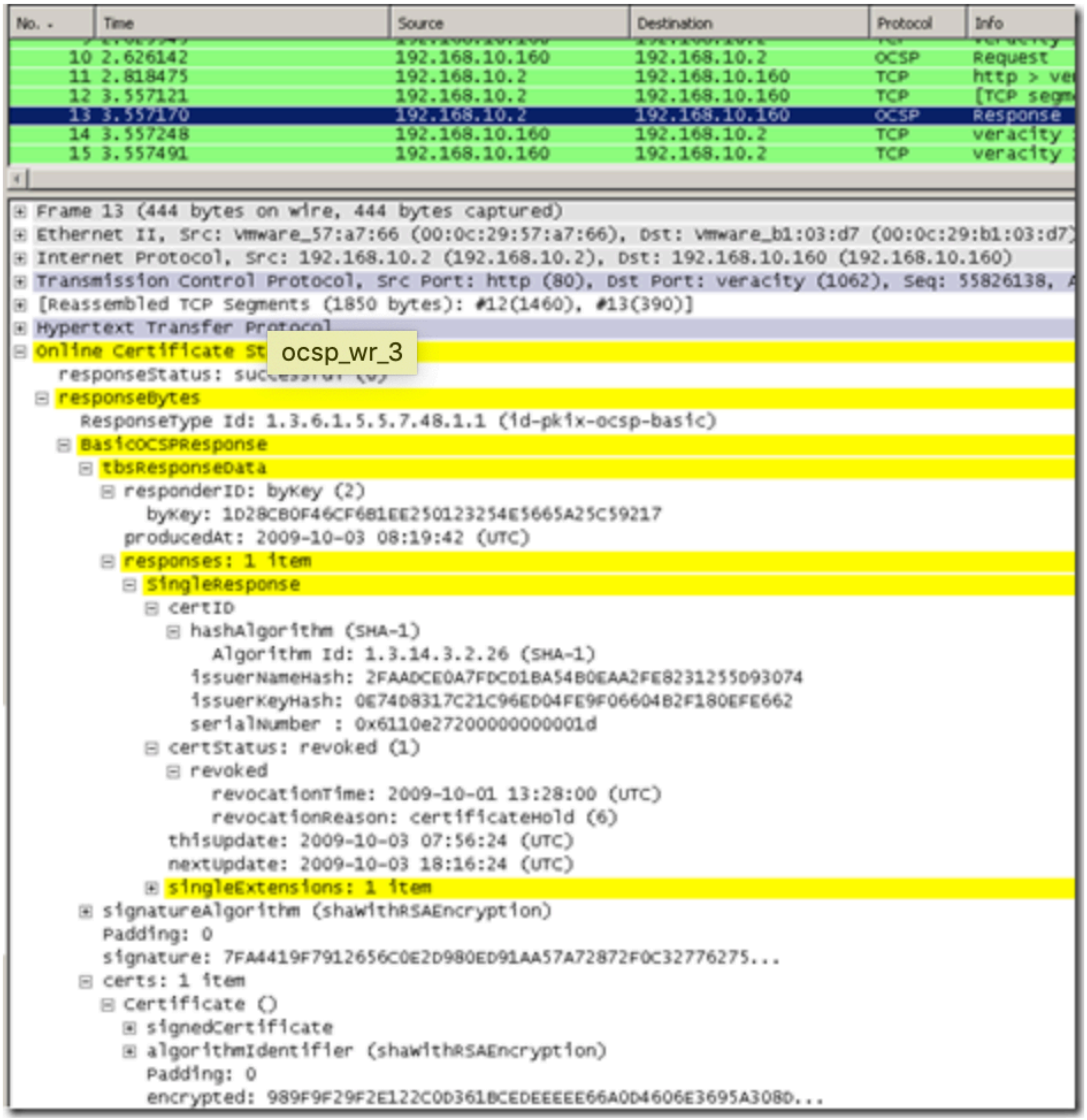
- OCSP Response

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.10.160	192.168.10.2	TCP	sacred >
2	0.000137	192.168.10.2	192.168.10.160	TCP	http > sa
3	0.000165	192.168.10.160	192.168.10.2	TCP	sacred >
4	0.000379	192.168.10.160	192.168.10.2	OCSP	Request
5	0.202151	192.168.10.2	192.168.10.160	TCP	http > sa
6	0.285244	192.168.10.2	192.168.10.160	TCP	[TCP segm
7	0.285278	192.168.10.2	192.168.10.160	OCSP	Response
8	0.285308	192.168.10.160	192.168.10.2	TCP	sacred >

```
⊞ Frame 7 (367 bytes on wire, 367 bytes captured)
⊞ Ethernet II, Src: Vmware_57:a7:66 (00:0c:29:57:a7:66), Dst: Vmware_b1:03:d7 (00:0c:29:b1:03:d7
⊞ Internet Protocol, Src: 192.168.10.2 (192.168.10.2), Dst: 192.168.10.160 (192.168.10.160)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: sacred (1118), Seq: 2186065053,
⊞ [Reassembled TCP Segments (1773 bytes): #6(1460), #7(313)]
⊞ Hypertext Transfer Protocol
⊟ Online Certificate Status Protocol
    responseStatus: successful (0)
  ⊟ responseBytes
      ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
    ⊟ BasicOCSPResponse
      ⊞ tbsResponseData
      ⊞ signatureAlgorithm (shaWithRSAEncryption)
        Padding: 0
        signature: 0E5230CC19E6370E39F1F3FA90A797E100D1DC7B5201F82B...
      ⊞ certs: 1 item
```

OCSP - Online Certificate Status Protocol

- **OCSP Response**



Outline

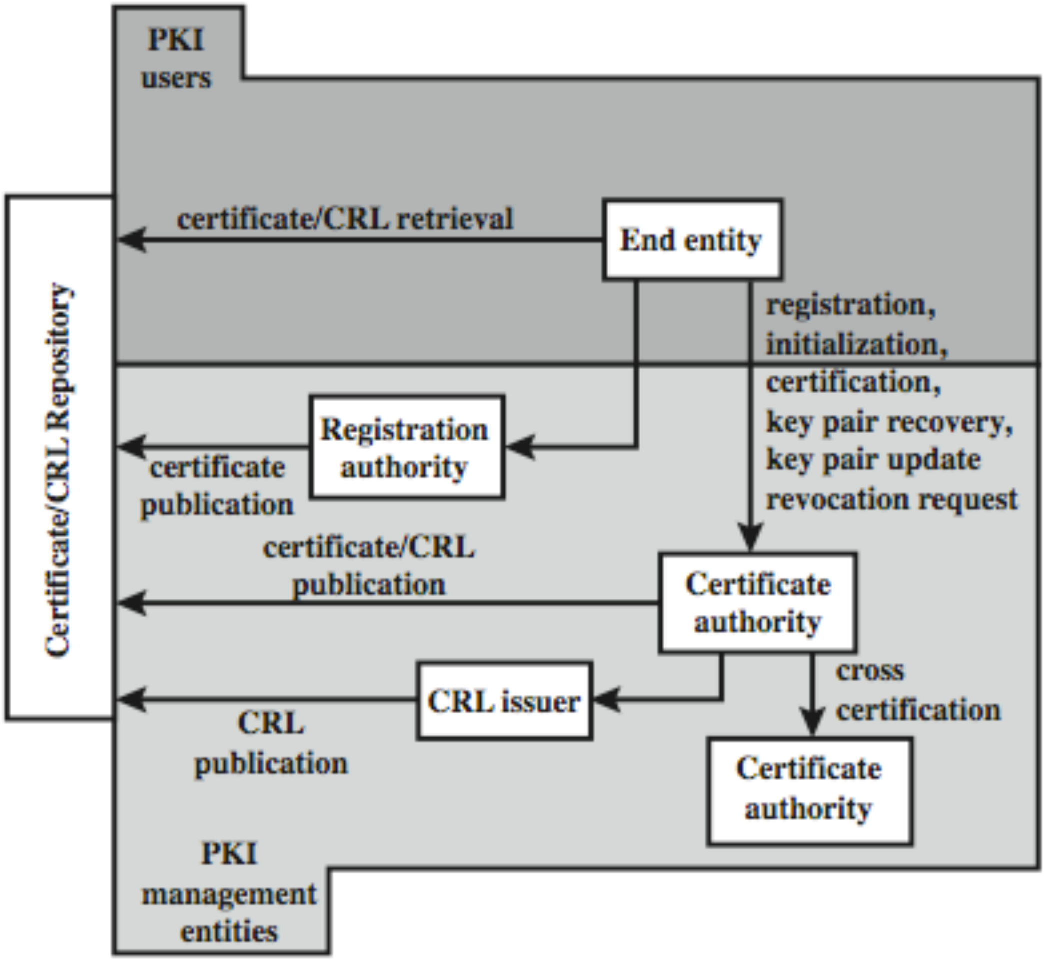
- **Key Management for Public Key Cryptography**
- **X509 Authentication**
  - X509 Certificates
  - Authentication procedures with X509
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Public Key Infrastructure

- PKI is the set of: HW, SW, People, Rules, Procedures, Policies and Protocols, needed to create, manage, store, distribute and revoke digital certificates
- Objective: to enable secure, convenient and efficient acquisition of public keys, promoting strict and well-known specifications
- Coordination from the IETF X509 (PKIX) WG
- Standardization base for compatibility purposes on the above issues

PKIX Architectural model and framework

- Key Elements
- Management Functions (APIs):
  - Registration
  - Initialization
  - Certification
  - Key-Recovering
  - Key-Update
  - Revocation Request
  - Cross Certification
- Management Protocols



PKIX Management Functions

- Registration
  - Enrollments from users to CAs (directly or through RAs)
  - Offline and Online procedures for mutual authentication
- Initialization
  - Initialization and installation of trusted CA certificates
- Certification
  - Registration of CSRs to obtain CA issued Certificates in standard formats (ex., PKCS#12, PEM, DER, BASE 64)
- Key Pair Recovery
  - Restoring encryption/decryption keys
- Key Pair Update
  - Regular updates and issuing of new certificates
- Revocation request
  - Regular updates and issuing of new certificates
- Cross certification
  - Exchanged signed CA public keys, between CAs

More extensible trust model

- Different entities involved, acting with different roles in a distributed way: **CAs, RAs, CRL Issuers, CRs**
  - Difference between:
    - **CA**: Certification authorities (Cert. ISSUING)
      - Different level CAs: aggregated in a direct certification chain
        - » Root CA, Level 2 CA, Level 3 CA, etc
        - » Model practically used in "well-known CA companies" or "CA delegation companies"
    - **R**: Registration authorities (REGISTRATION, ENROLLMENT DELEGATION)
    - **CRL Issuers**: (Issuers of CRLs)
    - **CRs or Certification Repositories** (DISTRIBUTION, for on demand REQUEST-REPLY

PKIX Management Protocols

- Standard protocols between PKIX entities supporting PKIX management functions

  Ex:
  - X509 Internet Public Key Infrastructure – Online certification status protocol (OCSP) RFC 6960
    - Update for previous RFC 5912, Obsoletes: RFCs 2560, 6277
  - CMP - Certificate Management Protocol: RFC 4210 (2015)
  - CMC – Certificate Management Messages over CMS: RFC 5272 > updated by recent RFC 6402 proposal
  - CMS – Cryptographic Message Syntax: RFC 5652 (obs. 3852)

See the standardization process from the
X509 PKIX IETF WG, … as time goes by ☺
http://datatracker.ietf.org/wg/pkix/

Programming support: ex., JAVA PKI API
http://docs.oracle.com/javase/6/docs/technotes/guides/security/certpath/CertPathProgGuide.html

Formats

At its core an X.509 certificate is a digital document that has been encoded and/or digitally signed according to RFC 5280 (PKIX).

See also (simple sumamry): https://en.wikipedia.org/wiki/X.509

- CSR: Certificate Signed Request
- DER Encoding: Binary based ASN.1
- PKCS#12, X509v3, BASE64 format encodings
- PKCS#7 format: CRLs - Certificate Revocation Lists:

Formats

- Encoding Conventions vs. file extensions:
- .pem – (
  Privacy-enhanced Electronic Mail) Base64 encoded DER certificate, enclosed between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----"
- .cer, .crt, .der – usually in binary
  DER form, but Base64-encoded certificates are common too (see .pem above)
- .p7b, .p7c –
  PKCS#7 SignedData structure without data, just certificate(s) or CRL(s)
- .p12 –
  PKCS#12, may contain certificate(s) (public) and private keys (password protected)
- .pfx – PFX, predecessor of PKCS#12

Conversions / Management of Formats

:-)))

- openssl x509 -outform der -in certificate.pem -out certificate.der
- openssl crl2pkcs7 -nocrl -certfile certificate.cer -out certificate.p7b -certfile CACert.cer
- openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.crt -certfile CACert.crt
- openssl x509 -inform der -in certificate.cer -out certificate.pem
- openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer
- openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer
- openssl pkcs12 -export -in certificate.cer -inkey privateKey.key -out certificate.pfx -certfile CACert.cer
- openssl pkcs12 -in certificate.pfx -out certificate.cer -nodes

Management / Generation of Certification Chains

Can use openssl tool (ex: Root > A > …)

- Root certification level:
  - openssl req -new -newkey rsa:1024 -nodes -out ca.csr -keyout ca.key
  - openssl x509 -trustout -signkey ca.key -days 365 -req -in ca.csr -out ca.pem
- "A" level:
  - openssl genrsa -out key_A.key 1024
  - openssl req -new -key key_A.key -out csr_A.csr
  - openssl x509 -req -days 365 -in csr_A.csr -CA CA_certificate_you_created.crt \ -CAkey CA_key_you_created.key -set_serial 01 -out crt_A.crt
- … and so on …

Complexity management issues (and usually flaws)

- Architectural weaknesses
- Problems involving certificate authorities
- Implementation issues
- Cryptographic weaknesses

Suggested Readings

Suggested Readings:

W. Stallings, Network Security Essentials – Applications and Standards, Chap 4., sections 4.5 – X509 and 4.6 - PKI