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

Mestrado Integrado em Engenharia Informática
MSc Course: Informatics Engineering
1º Semestre, 2019/2020

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

Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X509 Authentication

- Based on Algorithms and Constructions for Digital Signatures of Identity Claims (Asymmetric or Public-Key Cryptography) and trusted X509 certificates)
- Supported in Authentication Protocols

Signer (as the **authentication claimant of a certain digital identity claim**)

- Digital identity as unique identifier (UID)
- Must keep Private Key w/ required security assumptions
- Need that correspondent public-key must be known by the verifier (as the Authenticator peer)
- Control of the keypair generation process

Authenticator (as the verifier of the claimed identity signatures):

- Need to know/obtain public key of the claimant UID in a trusted way, to verify the signed authentication claim
- For X509 Authentication, trust assumptions are based on obtaining and managing X509 certificates (as trusted public key certificates)

Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Use of Public Key Crypto requires Secure and Trusted Key-Management

- **Generation control of keypairs**
- **Careful confinement, management and use (processing) in secure environments**
  - **Management of Private Keys** (in private-key rings)
  - **Public keys**: can be distributed, disseminated and publicly disclosed
    - Management as "public-key rings"
    - Trusted association to the correct UIDs of principals
    - Validation requires a trusted verification of such associations, as "verifiable" and "certified" associations

- Another issue: management of keys and certificates require the use of **standard and interoperable representation formats**
  - **Private and public keys or related parameters**
  - **Public key certificates / trusted management of public keys**

Management of Key Rings by Principals

Usually in Files

Trusted Associations
<subjectIDs, PublicKeys>

- As files, different formats
- As public keystores managing <subjectID$_i$, PublicKey$_i$> associations
  Ex: java keystores, PEM files, etc
- As trusted stores containing public key certificate stores and formats (ex., X509v3, PEM, DER. PKCS#12, etc.)

Usually in Protected (encrypted) Files

- As protected files w/ different formats
- As private keystores
  - java keystores wi/ different representations, ex: PEM, DER, PKCS#8

Management of Key Rings by Principals

Usually in Files

Trusted Associations
<subjectIDs, PublicKeys>

Public Key Ring

- As files, different formats
- As public keystores managing <$subjectID_i$, $PublicKey_i$> associations
  Ex: java keystores, PEM files, etc
- As trusted stores containing public key certificate stores and formats (ex., X509v3, PEM, DER. PKCS#12, etc.)

Usually in Protected (encrypted) Files

Master Keys
- Generated from secret seeds or passphrases
- Symmetric Encryption
- PWD-based Encryption

- As protected files w/ different formats
- As private keystores
  - java keystores wi/ different representations, ex: PEM, DER, PKCS#8

Protection of Private Keys

- **Private Keys**: must be protected from exposition risks, avoiding:
  - **Storage exposition**
    - Use of secure storage (encrypted)
      - Encrypted in disks or other storage devices
      - But where are the protection encryption keys ?
      - What if Protection Keys are "lost" ? Recovery-Mechanism
    - Ex., Keystores, protected by PBE and/or Symmetric Encryption

  - **Memory exposition** (when transferred to, managed and processed in memory) must be in memory w/ minimal exposure - only when required !

    - **Better:** stored and processed in locked "devices" or "appliances" where it may be impossible (or unlikely) the access by no-authorized parties (w/ cryptographic operations possibly performed in those devices)
      - Never exposed outside these devices !
      - Require crypto operations supported and processed "inside"
      - Access-control via authentication and cryptographic APIs

HSMs (Hardware Security Modules): Ex. of manufacturers, IBM, Safenet, nShield, …

https://www.ibm.com/security/cryptocards/hsms

IBM Crypto Express Modules

IBM PCIe
Crypto Coprocessor

Safenet
LAN-Based HSMs

Safenet
USB-Based HSM

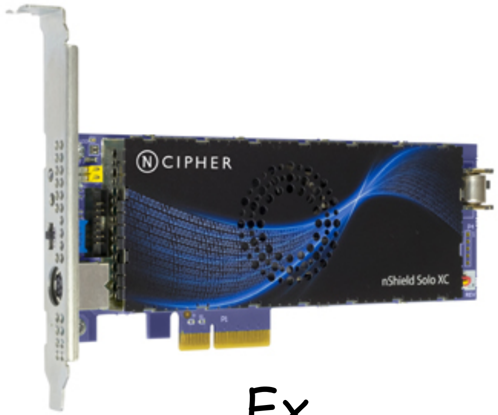
Safenet
HSM Backup Appliance

Safenet
PCIs-Based HSM

https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/

HSMs (Hardware Security Modules): Ex., nShield, ...

Ex.
nShield Connect
(Net Appliance)



Ex.
nShield Solo
(PCIe enabled)

Ex.
nShield Edge
(USB enabled)

https://www.ncipher.com/products/general-purpose-hsms

HSM Typical Features

- High performance cryptographic operations
- Compliance:
  - Security: FIPS 140.2 Levels 2 and 3, USGv6, Com. Criteria EAL4
  - Ex., Safety and environmental standards
- Supported cryptographic APIs (CAPIs): (the external surface)
  - PKCS#11
  - OpenSSL
  - Java JCE
  - Microsoft CAPI
  - CNG API
- OS and Virtualization compliance
- Reliability MTBF Metrics (~100000 hours)
- Security/Robustness:
  - Products w/ broad acceptance and evaluation
  - But ....

https://cryptosense.com/blog/how-ledger-hacked-an-hsm

HSMs can improve considerably the performance of cryptographic operations

Ex., Compare w/ openssl performance in your computer ;-): openssl speed rsa ecc

nShield Connect Models	500+	XC Base	1500+	6000+	XC Mid	XC High
RSA Signing Performance (tps) for NIST Recommended Key Lengths						
2048 bit	150	430	450	3000	3500	8600
4096 bit	80	100	190	500	850	2025
ECC Prime Curve Signing Performance (tps) for NIST Recommended Key Lengths						
256 bit	540	680	1260	2400	5500	14,400

Devices for personal use

https://www.yubico.com/products/yubihsm/

Ex., YubyKey Series
USB-A, USB-C
Lightening, NFC

https://www.schneier.com/blog/archives/2019/07/yubico_security.html

Smartcards, Smartcard Readers

Cardomatic Smartcard-HSM
USB Stick

https://www.cardomatic.de/SmartCard-HSM-USB-Stick/

USB

USB +
Local Auth.
And Access Control
Pin/Pwd

USB +
Local Auth
and Access Control
Biometry

Interaction w/ Smartcards and other cryptographic devices

- Interface (via reader) by sending commands / receiving results: **APDUs or App. Protocol Data Units)**
  - APDUs are standardized messages (msg in / msg out)
- Note: APDUs are standardized structures but the content may be different depending on specific implementations
  - Many Smartcard manufacturers, variety of implementations and programming support
  - Applications (and programmers) don't use directly (in general) APDUs (considered a low level abstraction)

- Use of **more high-level abstractions or programming interfaces**
  - **Crypto APIs**
  - Provide standard generic primitives allowing the manipulation of objects in the smartcard, cryptographic and key-management operations
  - Examples:
    - **PKCS#11 (Crypto API 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 a physical device (such as a smartcard) regarded as a "cryptographic token"
  - Implements an Object-Oriented Interface, through Middleware (libraries) provided by manufacturers
    - Also 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.
  - 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/JCE APIs **can, without modification, take advantage of algorithms offered by underlying PKCS#11 implementations, such as, for example:**
  - Cryptographic Smartcards,
  - HSMs or Hardware cryptographic accelerators
  - High performance software implementations.

PKCS#11 in Java

- A Java PKCS#11 Crypto Provider can be installed or used as any other crypto provider: use the device as a "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)

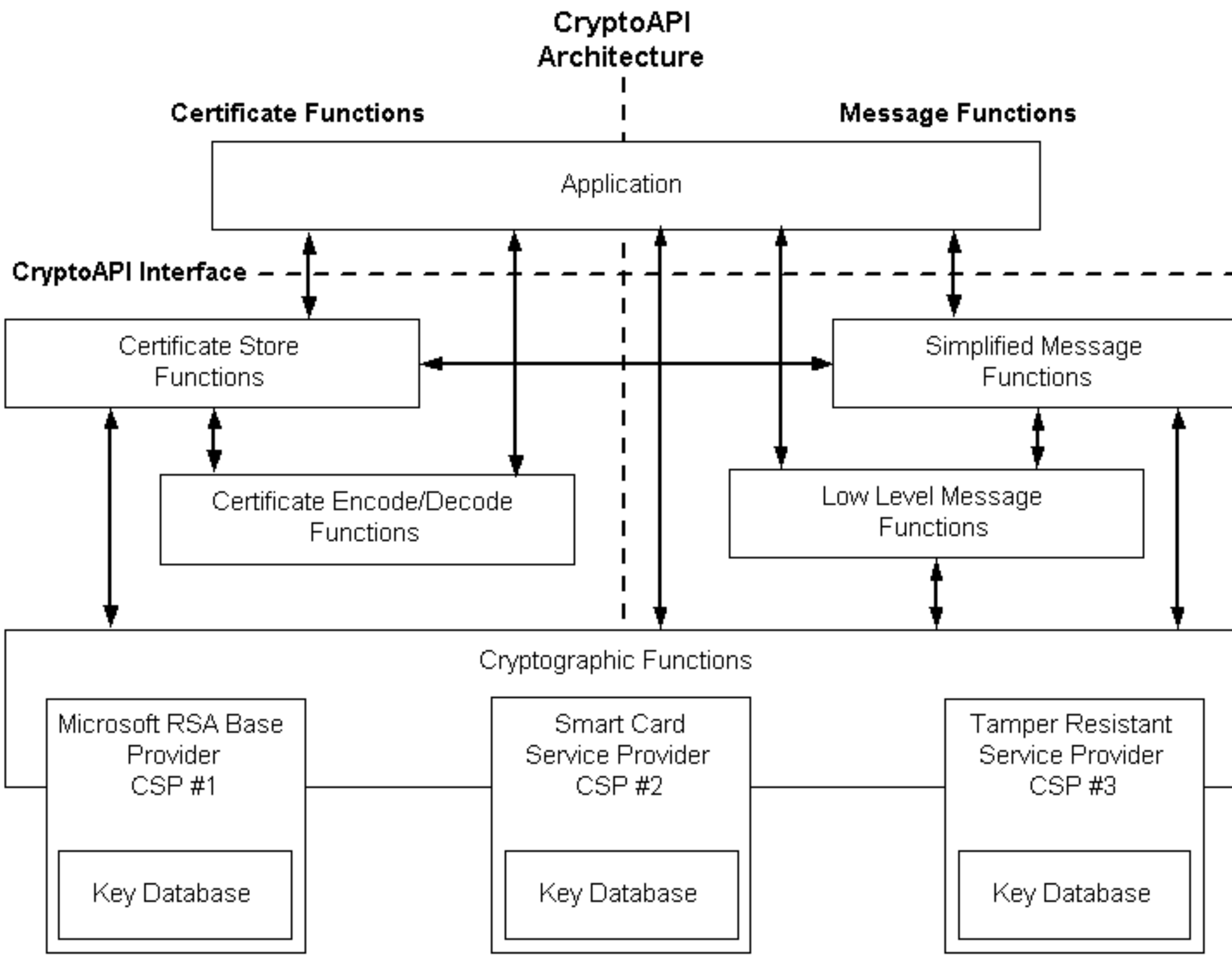
- High-Level Middleware Integration, including Smartcard interoperability for Microsoft Windows OS
- Architecture based on a generic module (providing an external API) and specific CSP (*Cryptographic Service Providers*), each one provided for specific physical devices
  - One CSP can or cannot use the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

See more in:

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

Microsoft CryptoAPI System Architecture CryptoNG API (aka CNG) and CAPICOM

https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptography--cryptoapi--and-capicom



Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X.509 Standardization

X509: a standard framework, part of the ITU-T X500 standardization effort, initially targeted for:

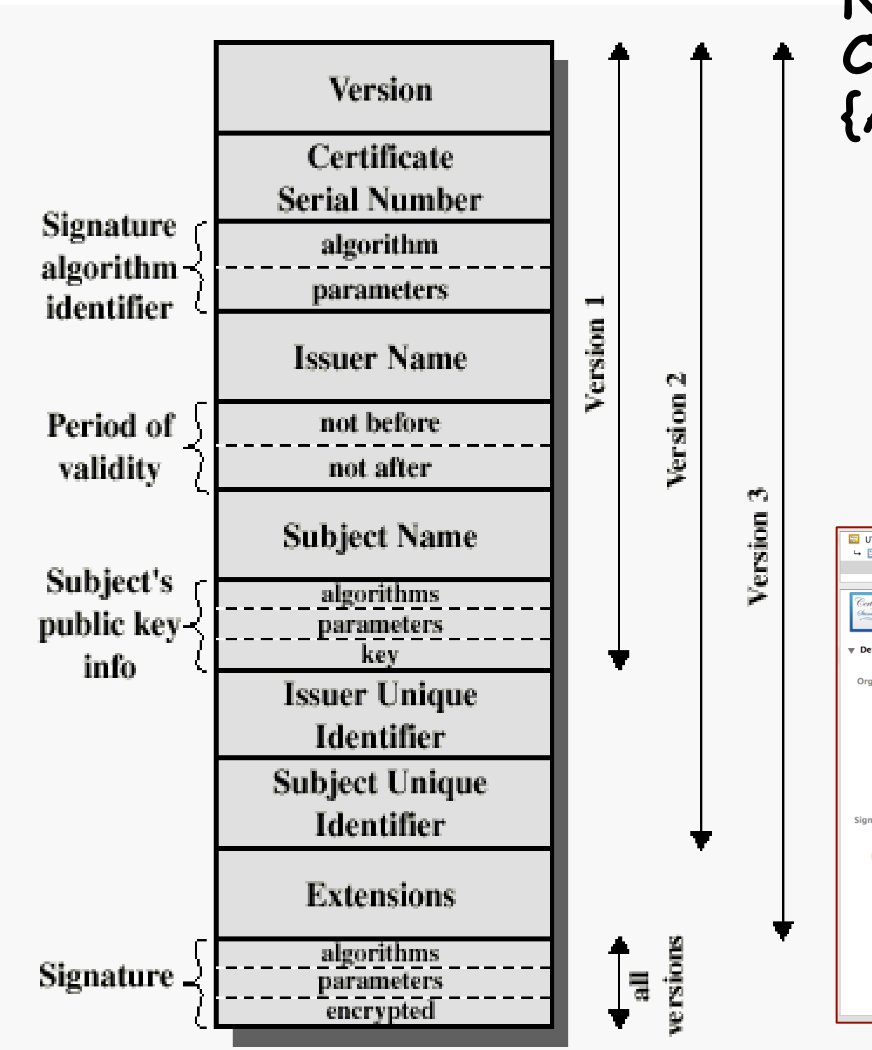
- Provision of **authentication services by X500 directory service**
- Standard representation of keys and public key certificates (formats and their attributes and data representation types), as well as recommended cryptography (algorithms and parameters)
  - Currently: X509v3 Certificates and X509v3 EV (Extended Validation) Certificates
  - Canonical Encoding Standardization
- Framework to address PKI systems (**P**ublic **K**ey **I**nfrastructures)
  - Processes, entity roles, interfaces)
  - Life cycle management of certificates: generation, enrollment, certification requests, certificate issuing, validation, revocation

---

Standardization: 1988, 1993 (v1), 1995 (v2), 2000 (v3), …

IETF RFC 2459 (Jan 1999) …... RFC 8399 (May/2018)

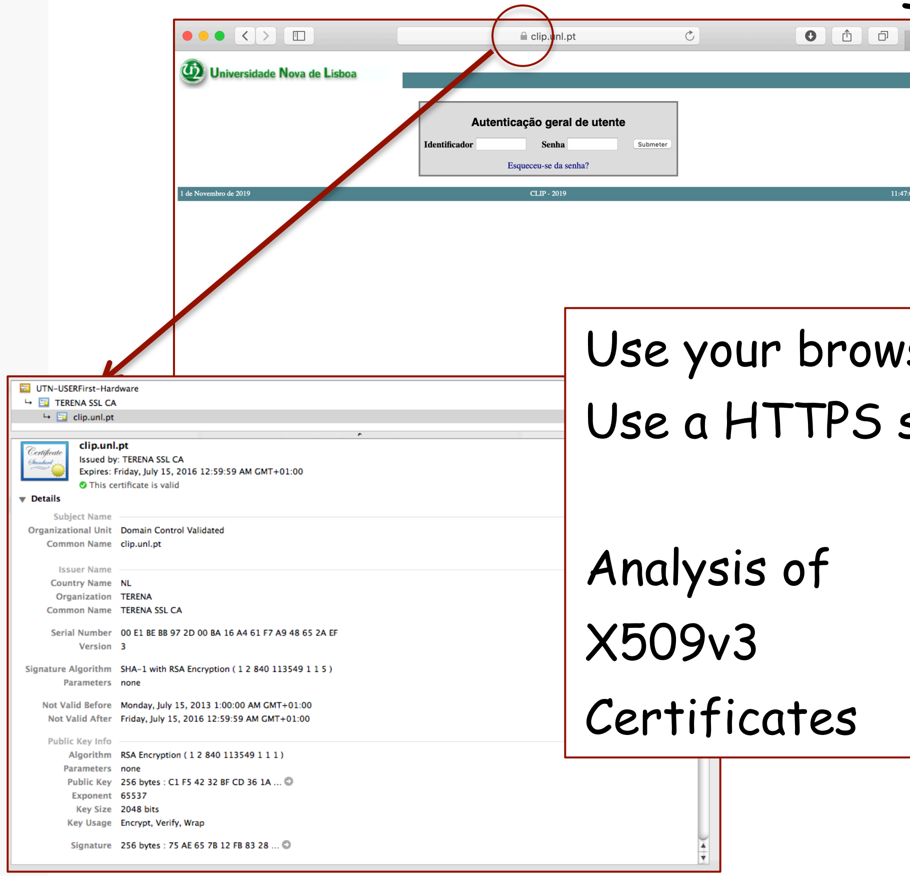
X.509 v3 Certificate: Structure, Attributes, Extensions, Classifiers



Notation:

CA <<A>> =

$\{A, V, SN, AI, CA, TA, KpubA\}_{SigCA}$



Use your browser
Use a HTTPS site

Analysis of X509v3 Certificates

X509 certificate (Extended attributes: improved in different versions)

X.509 Certificates

Each certificate contains:

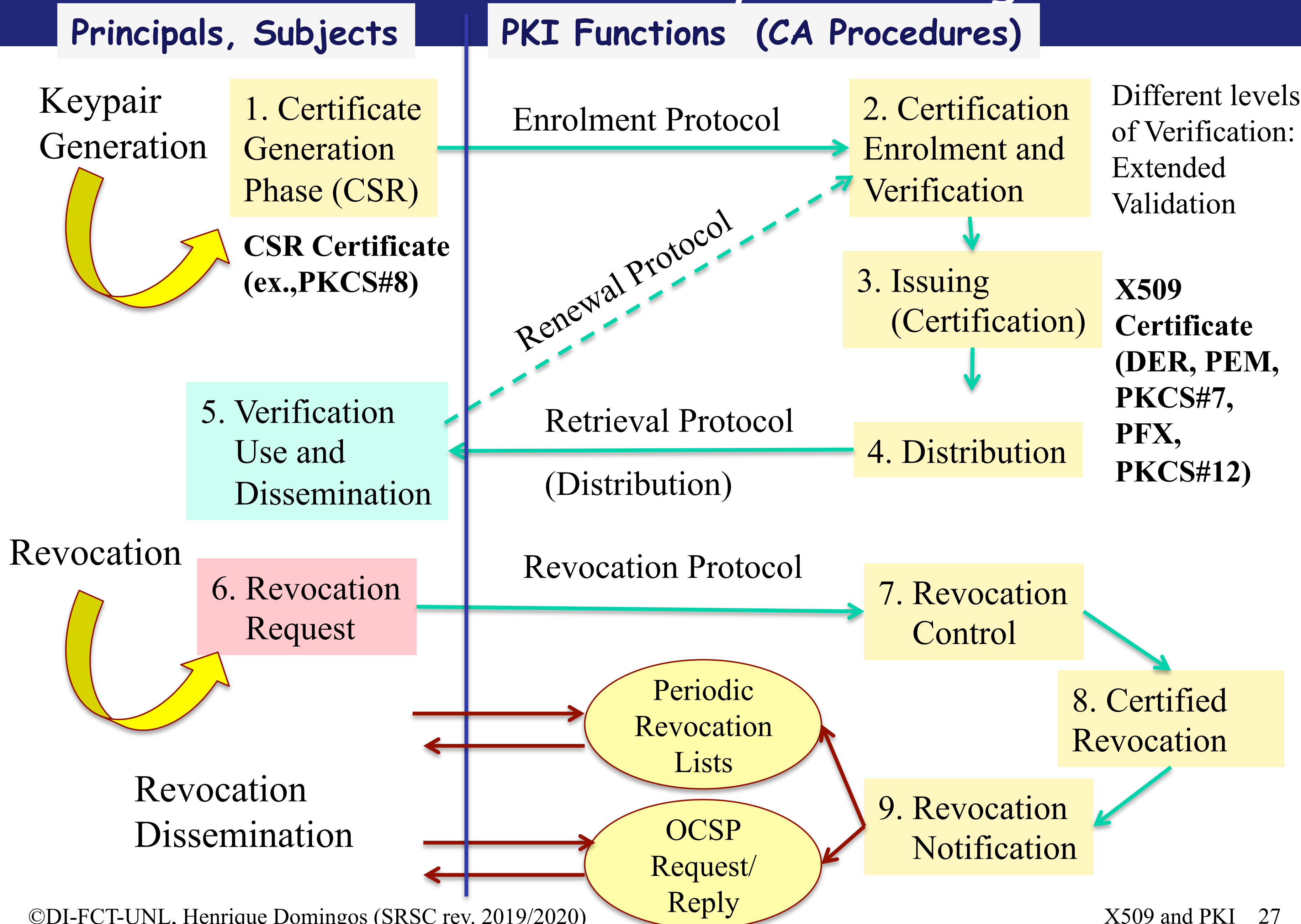
- The public key of a distinguished subject name (principal, user)
  - Subject name, Subject's public key information fields
- Other attributes with additional information as a list of other (field, value) pairs
  - Issuer UID, serial number, version, validity information, relevant information of cipher-suites 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

Discussion: see the different fields, policies and extended attributes in current X509v3 Certificates

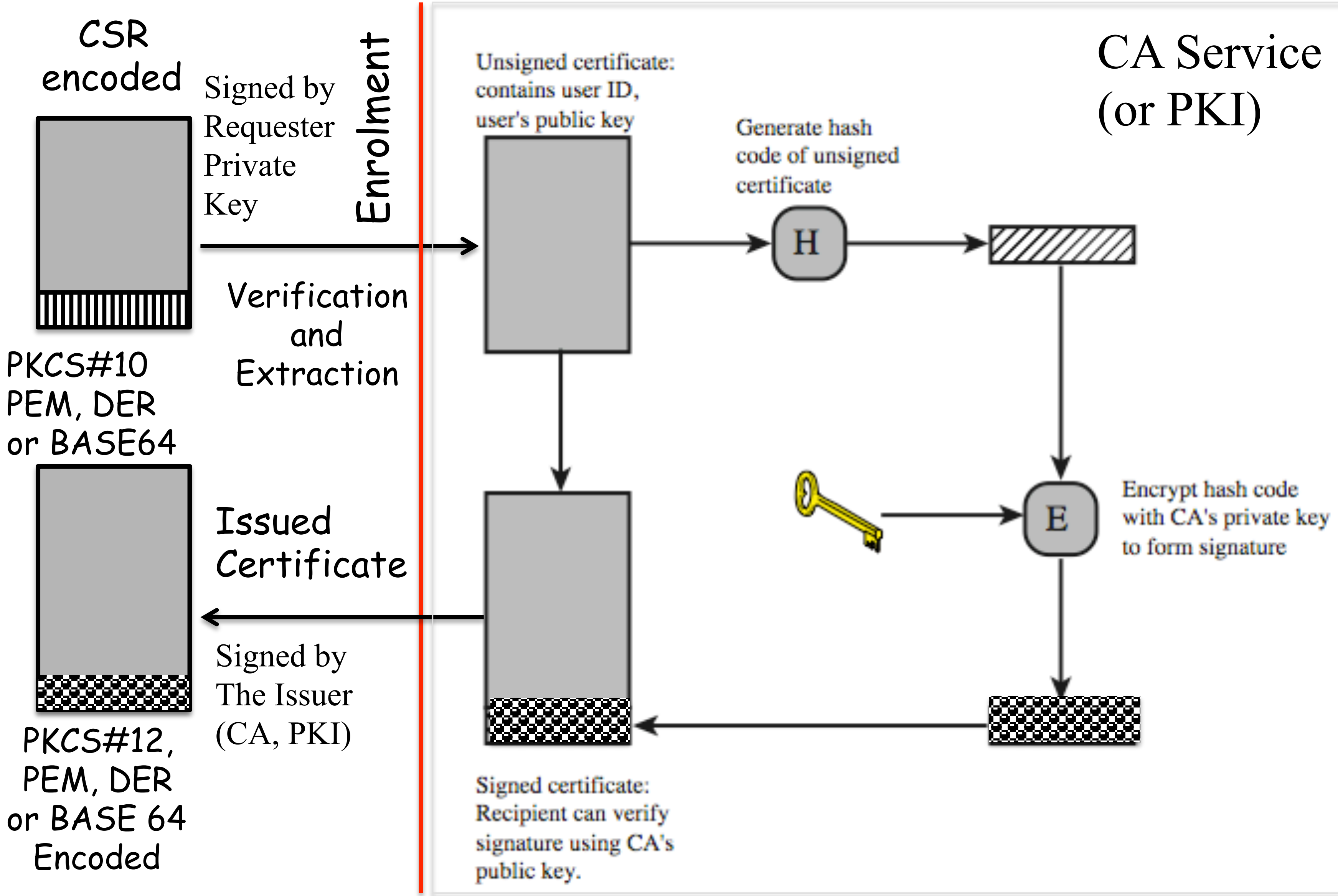
Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X509 Certificates: Life Cycle Management



CSR Validation, extraction and signing (Issuing of Certificates)



Obtaining a User's Certificate

- Certificates: issued by CAs (Functions on PKIs)
  - Any user with access to the public key of the CA can recover and validate the certified user public key
  - Users can exchange certificates and certification chains for verification
    - Can use direct or reverse chains for verification
  - Certificates are public and unforgeable (signed by the issuer CA).
    - Possible to send/distribute/disseminate them in protocols or placed in public directories or repositories
    - Note: having a certificate is not a proof of authentication
      - Need a digital signature, exhibiting the public key certificate to validate the signature

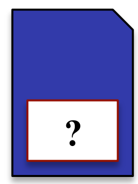
Typical life cycle management

Principals (Sujects):

Generate Keypairs (RSA, DSA)

Generation of Self-Signed Pub-Key Certificate

Only usable by principals accepting it (in their trusted cert stores)



Generation of CSR Certificate

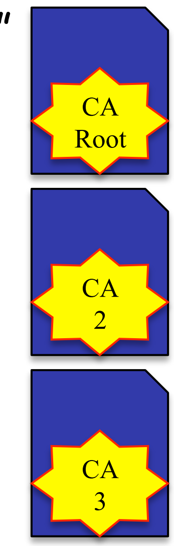
Secure storage & management Of Keypair and Private Key !

Enrolment Process for certification

Certification Authority (or PKI solution)

Has a "well-known" disseminated Root Public key Certificate

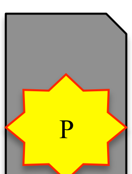
Has Issued Intermediate CA certificates (in a chain)



Validation of enrolment and CSR Certificates

Issues generated certified (signed) X509v3 certificates (in a certain chain)

X509v3 issued certificate

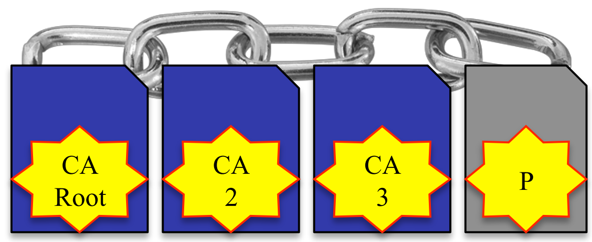


Receives their Issued X509v3 certificates

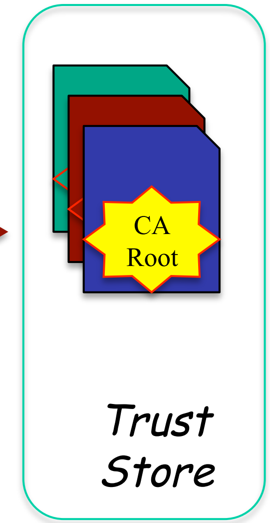
Ready for use when presented in their certification chain

Certification Chains

Principal A



Principal B



Can Verify the Rest of the Chain (Attributes and Chained Signatures)

YES NO

Outline

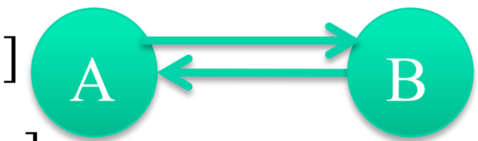
- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Summary of Base Authentication Procedures

One-way authentication and Key dist.

A[{ta, ra, IdB}Kab, Sig_{KprivA} (signData), $\{Kab\}_{KpubB}$]

Two-way (mutual) authentication and Key dist.



A [{ta, ra, IdB}Kab, Sig_{KprivA} (signData), $\{Kab\}_{KpubB}$]

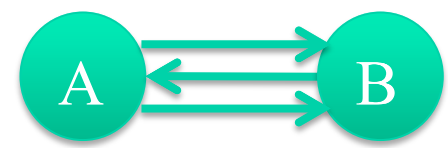
B [{tb, rb, IdA}Kba, Sig_{KprivB} (signData), {Kba}KpubA]

Three-way (Mutual) authentication and Key Dist.

A[{ta, ra, IdB}Kab, Sig_{KprivA} (signData), {Kab}KpubB]

B[{tb, rb, IdA}Kba, Sig_{KprivB} (signData), {Kba}KpubA]

A{rb}



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) establishes in addition:
  - 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 a *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 no need of 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

Authentication Procedures
Example of concretizations

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 ?

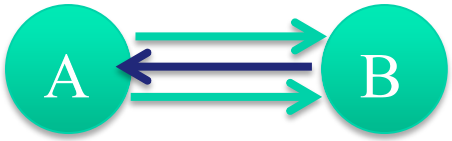
"On the Fly" validation of trust chains

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

> Advantages ? Drawbacks ?

Base Authentication variants (Variant 1)

One-way authentication and Key dist.



A: I am A

B: Authentication challenge **Cb** for the claimer

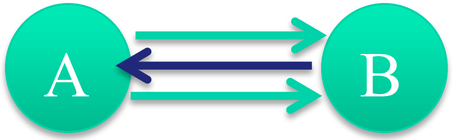
A[{ta, ra, **Cbr**, IdB}Kab, Sig_{KprivA}(signData), $\{Kab\}_{KpubB}$]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

Base Authentication variants (Ex., Variants 1)

One-way authentication and Key dist.



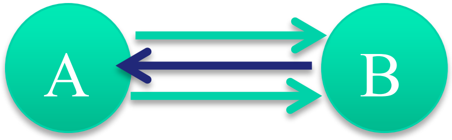
A: I am A, <my ciphersuite proposal>
B: Challenge **Cb**, <my ciphersuite choice>
A [{ta, ra, **Cbr**, IdB}Kab, Sig_{KprivA}(signData), {Kab}KpubB]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

Base Authentication Variants (Ex., Variants 2)

One-way authentication and Key dist.



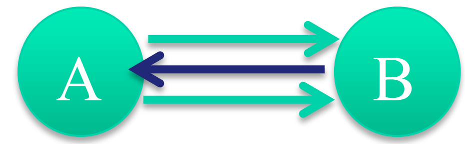
A: I am A, <my ciphersuite proposal>, $CERT_A$
B: Challenge **Cb**, <my ciphersuite choice>, $CERT_B$
A[{ta, ra, **Cbr**, IdB}Kab, Sig_{KprivA}(signData), {Kab}KpubB]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

Base Authentication Procedures (Ex., Variants 3)

One-way authentication and Key dist.



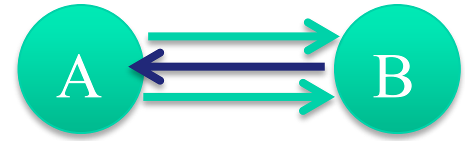
A: I am A, <my ciphersuite proposal>, <Certification Chain>
B[**Cb** Challenge, <my ciphersuite choice>, <Certification Chain>]
A[{ta, ra, **Cbr**, IdB}Kab, Sig_{KprivA}(signData), {Kab}KpubB]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

Base Authentication Procedures (Ex., Variants 4)

One-way authentication and Key dist.



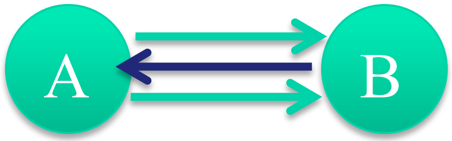
A: I am A, <my ciphersuite proposal>, <Certification Chain>
B: **Cb** Challenge, <my ciphersuite choice>, Sig_{KprivB}(signData), <Cert Chain>
A[{ta, ra, **Cbr**, IdB}Kab, Sig_{KprivA}(signData), $\{Kab\}_{KpubB}$]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

Base Authentication Procedures (Ex., Variants 5)

One-way authentication and Key dist.



A: I am A, <my ciphersuite proposal>, <Certification Chain>

B: **Cb**, <my ciphersuite choice>, Sig_{KprivB}(DHpubB, SignData), <Cert Chain>

A[{ta, ra, **Cbr**, IdB}Ks, Sig_{KprivA}(DHpubA, signData)]

Two-way (mutual) authentication and Key dist.

Three-way (Mutual) authentication and Key Dist.

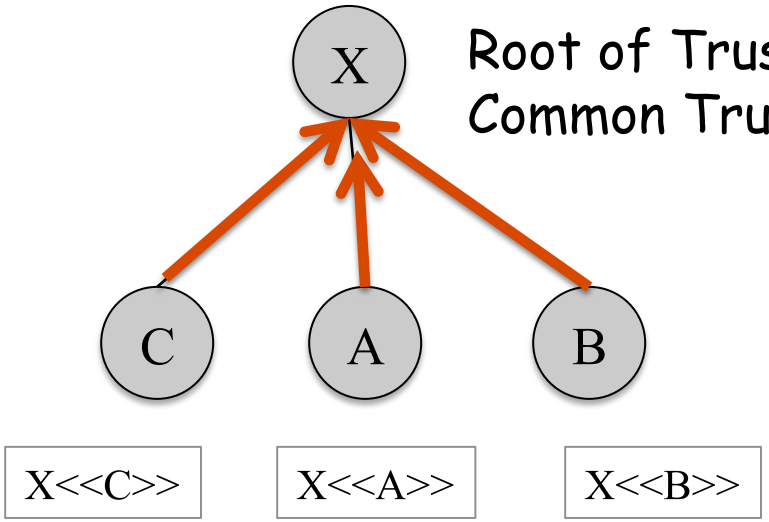
Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Trust and Validation Chains

Common trust based validation

- When all users subscribe to the same **Root Of Trust** X
- Ex., Model for a small community of users (non-scalable, centralized-root trust)
- Any user A transmits directly the certificate to any other (B, C)



Root of Trust:
Common Trust (ex., Common CA)

What if we have more than one RoT (or CA)

No common trust verification conditions

- Model for a large community of users (scalable model)
- Users need to have Public Keys of all the CAs ?
- It may be more practical to consider that
  - There will be several Roots of Trust (CAs),
  - But each of which securely provides its public key to some fraction of the users
  - Then we can use cross-certification links in a certification hierarchy

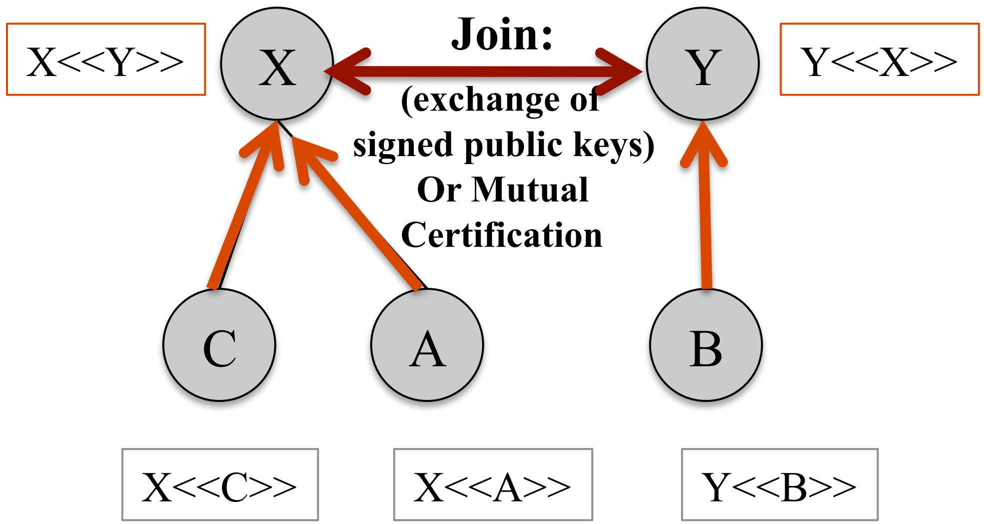
Notation for a Public Key Certificate:

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

Y<<X>> means: Certificate of entity **X** issued by **Y**

Verification of certificates => imply that the verifiers previously obtained, in a trusted way, the CA public key

Solution for no Common Trust: Peering



- **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)

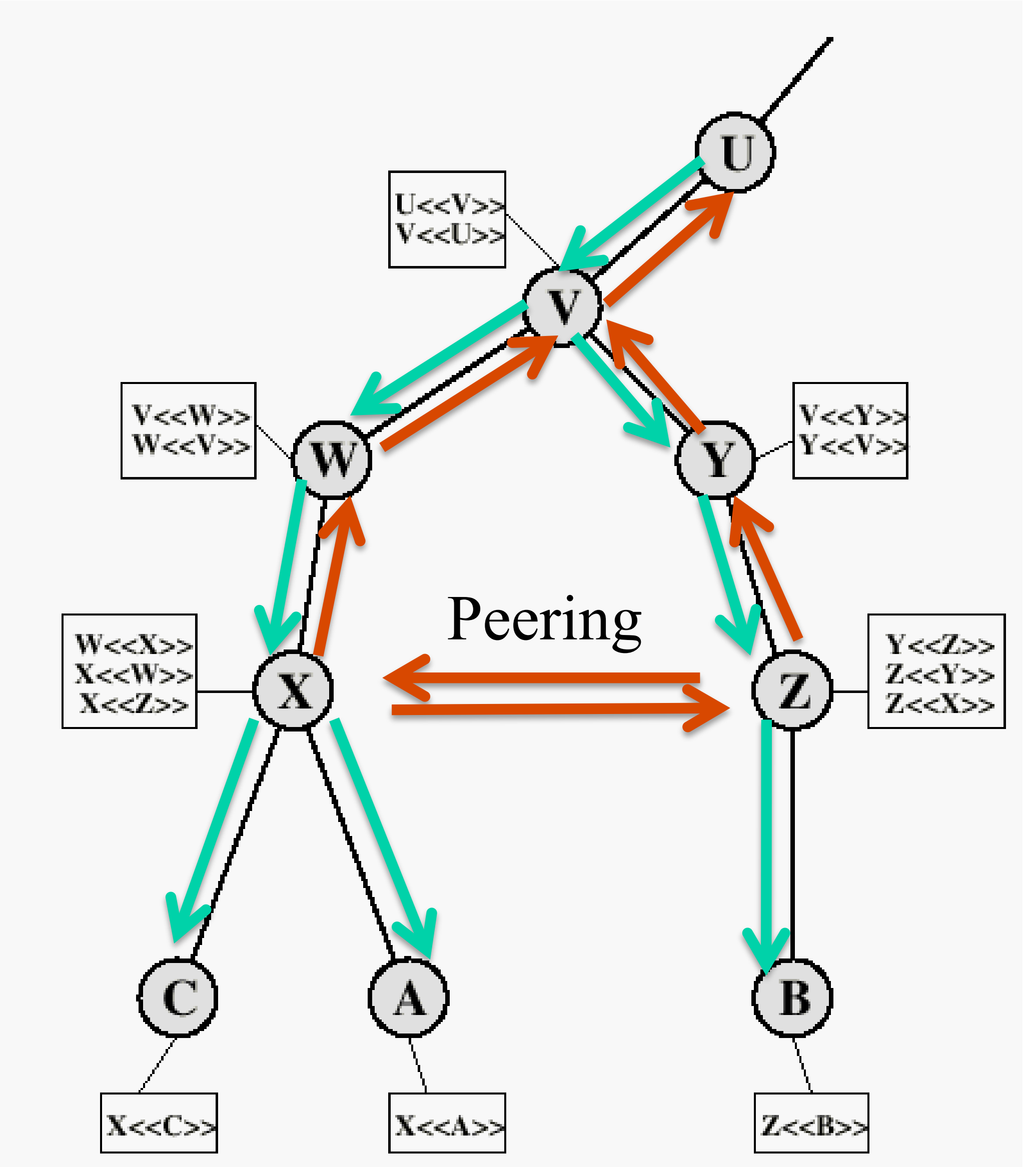
X.509 CA Hierarchy and Chains

- 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
  - the authentication processing supported with a direct certification chain validation

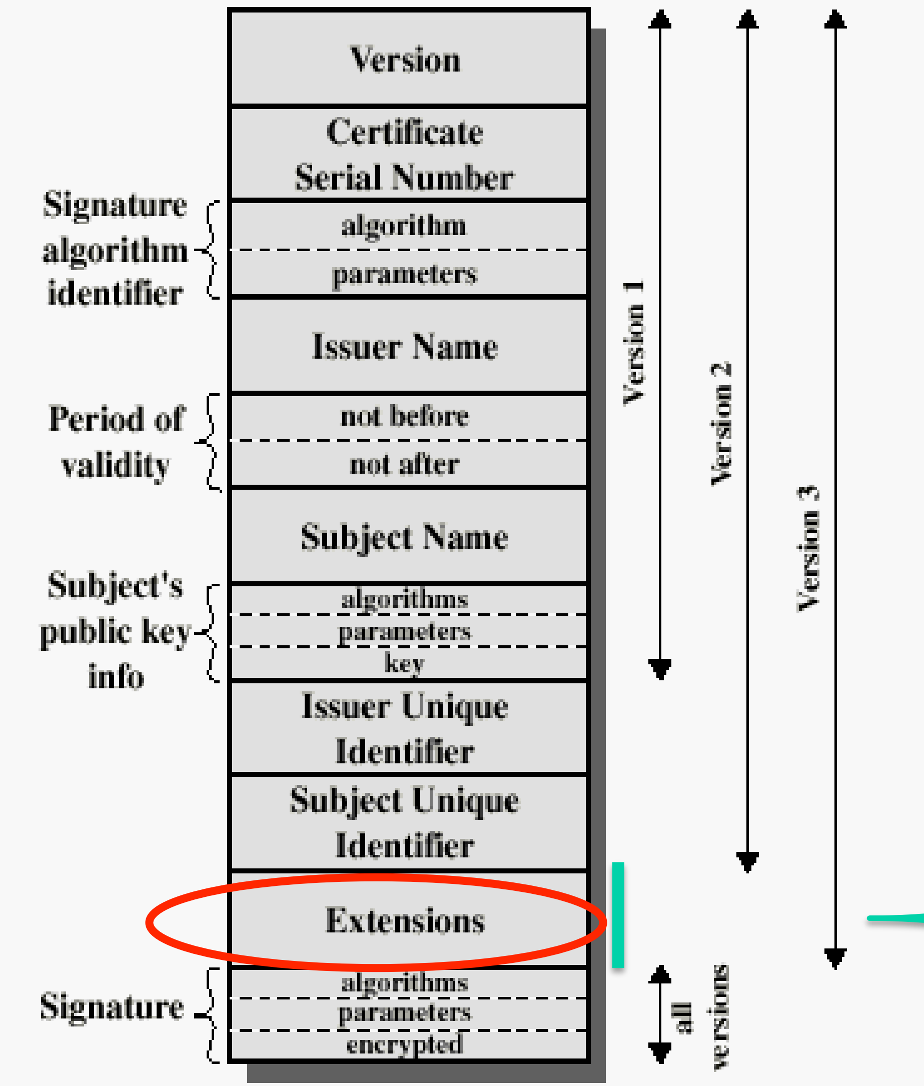
- Ex., see the CA's Root Certificates in your Java installation
  - Find **cacerts** in your /……/**jre/lib/security hierarchy**

- See the certification chain in a TLS (HTTPS) connection:
  - Can use your Browser
  - Or can use openssl
    - `openssl s_client -connect www.feistyduck.com:443`

Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **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

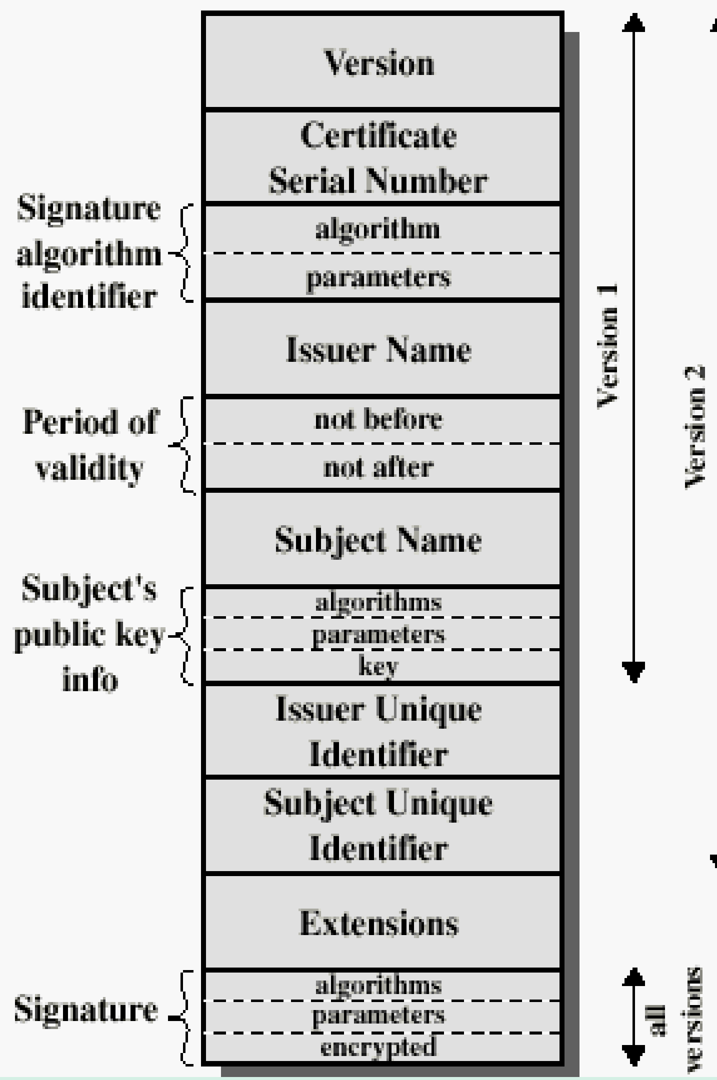
- Introduced by the CA/Browser forum
  - http://www.cabforum.org/, http://en.wikipedia.org/wiki/Extended_Validation_Certificate
  - CAs + Relying Party Application Software Suppliers
- Objective: inclusion of standardized procedures for verifying and expressing awareness of the certificate holder and validity (initially motivated by SSL - TLS 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**

Relevance of The CA/RA Delegation Models

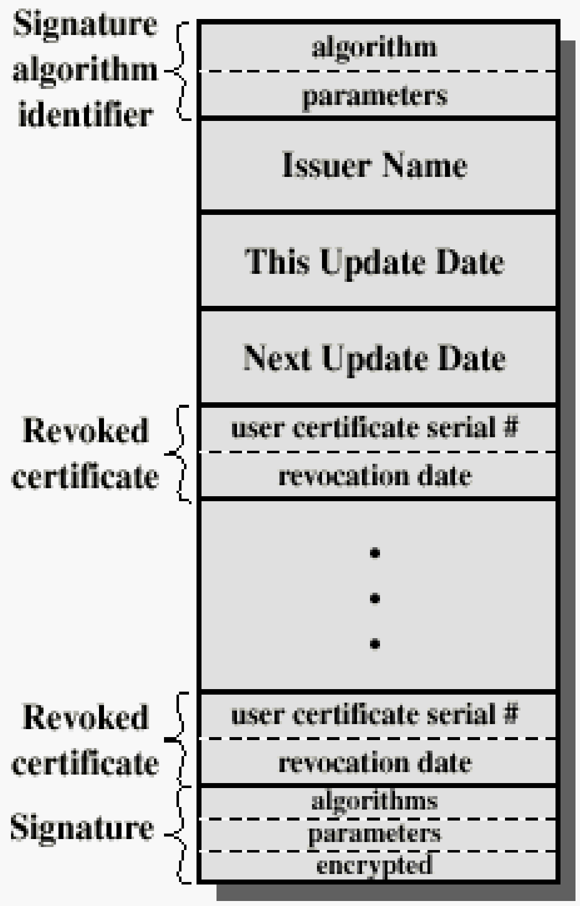
Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

X509 Certificates and CRLs



X509 certificate (fields in different versions)



X509 Certificate Revocation List

Revocation of Certificates: Why, When, How

- **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 (specific attributes, policies, extensions)
  - If it is in a "current" certification revocation list (CRL) issued by the CA that issued the certificate
  - If not validated by synchronous "on line" verification process
    - Via OCSP Protocol

Management of CRLs

- Maintained by each CA (or CRL issuers' end-points)
- Usually provided in DER or PEM Formats
  - A list of revoked (not expired) certificates issued by that CA, including
    - End-user certificates
    - Possible reverse certificates
- CRLs must be managed by final users (user responsibility)
  - Checked from a directory, every time a certificate is received
    - CRL endpoints (in issued X509 certificates)
- Checked from a local cache, periodically updated (ex., Incremental, Time-Controlled, Serial Number Controlled)
    - **Black Lists: CRLs**
    - **Full-Lists vs. Incremental Lists**
    - **Time-controlled vs. Version-Controlled**
    - Also possible: White Lists as White CRLs

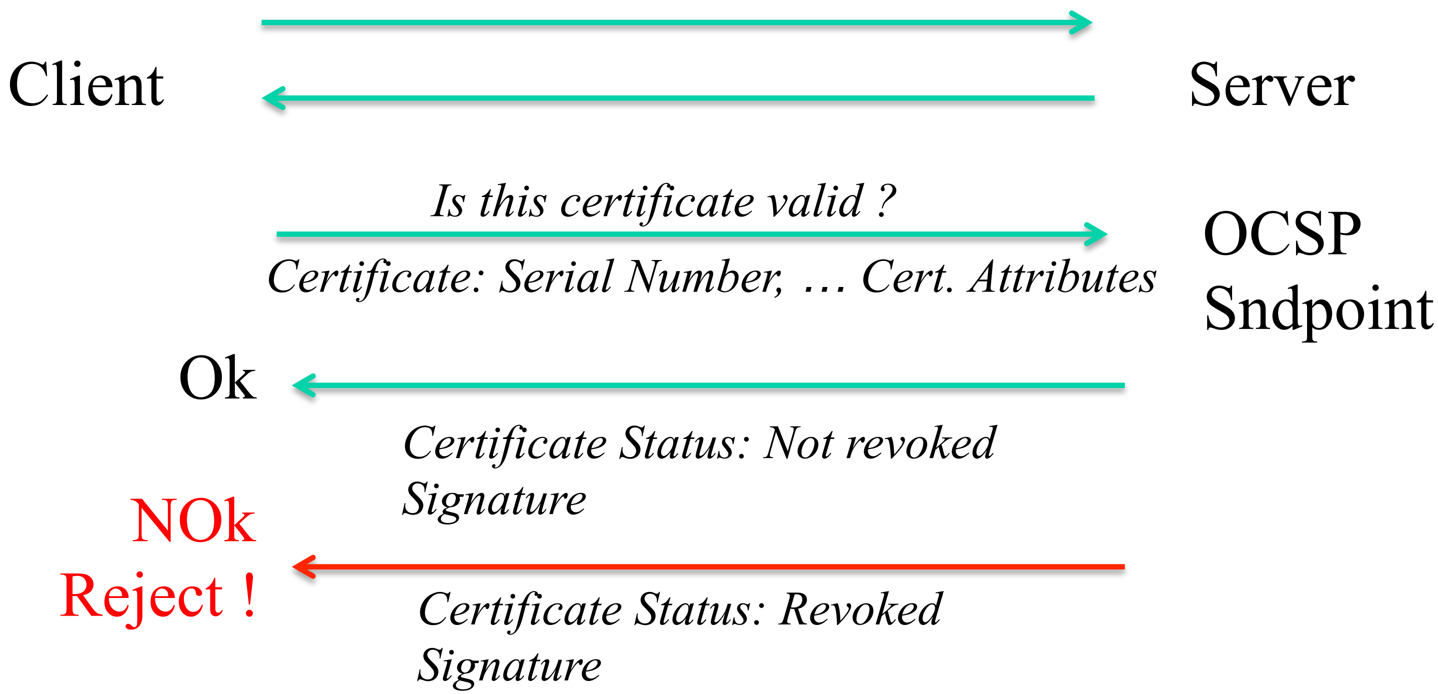
See a CRL, as usually issued by CAs

- Download the current CRL from the CRL endpoint of a given (issued) certificate
- Inspect the CRL (example w/ keytool and openssl):

```
keytool -printcrl –file <obtainedcrl>
openssl crl –inform DER –text –noout –in <obtainedcrl>
```

Revocation control w/ the OCSP Protocol

- OCSP – On Line Certificate Status Protocol
  - Client/Server Request/Reply Protocol
  - OCSP Endpoints provided by CAs
    - OCSP Endpoint Attribute in issued X509 Certificates



OCSP (example with openssl)

- Given a certificate (ex.): certificte.pem as a chained certificate
- Verify the OCSP endpoint attribute (typically a given URL)
- Verification of all certificates in the chain
- Use of openssl:

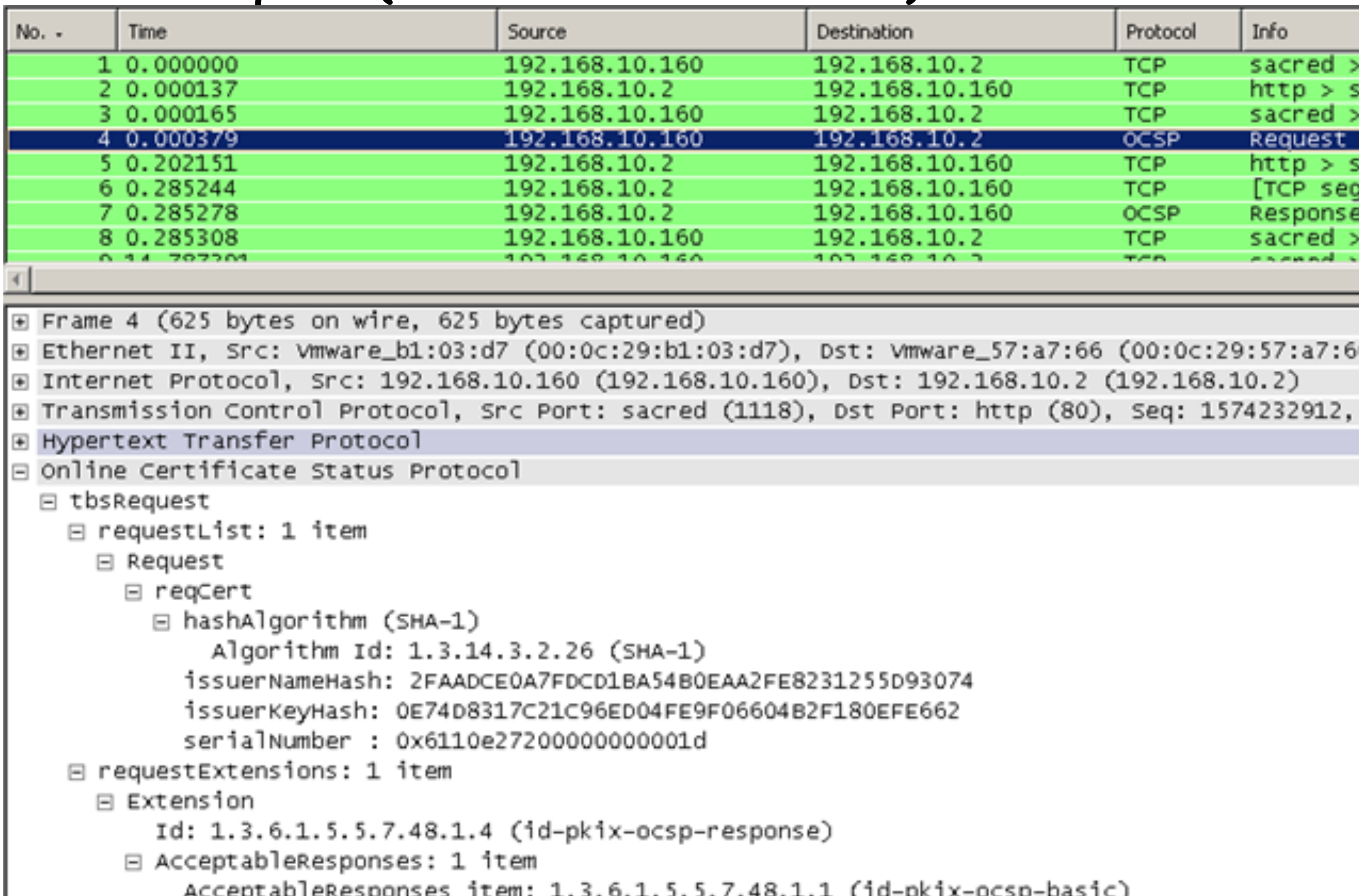
openssl ocsp -issuer certificate.pem -cert sslcert.pem -url <http://OCSP-URL> -text -CAfile CAchainfile.pem

```
WARNING: no nonce in response
Response verify OK
sslcert.pem: good
        This Update: Mar 13 17:13:19 2012 GMT
        Next Update: Mar 20 17:13:19 2012 GMT
```

```
WARNING: no nonce in response
Response verify OK
sslcert.pem: revoked
        This Update: Mar 16 16:18:11 2012 GMT
        Next Update: Jun 11 00:52:47 2012 GMT
        Reason: keyCompromise
        Revocation Time: Mar 16 16:16:56 2012 GMT
```

OCSP - Online Certificate Status Protocol

- A Request/Response Protocol, usually supported in HTTP
  - **OCSP Request (with the wireshark tool)**



OCSP - Online Certificate Status Protocol

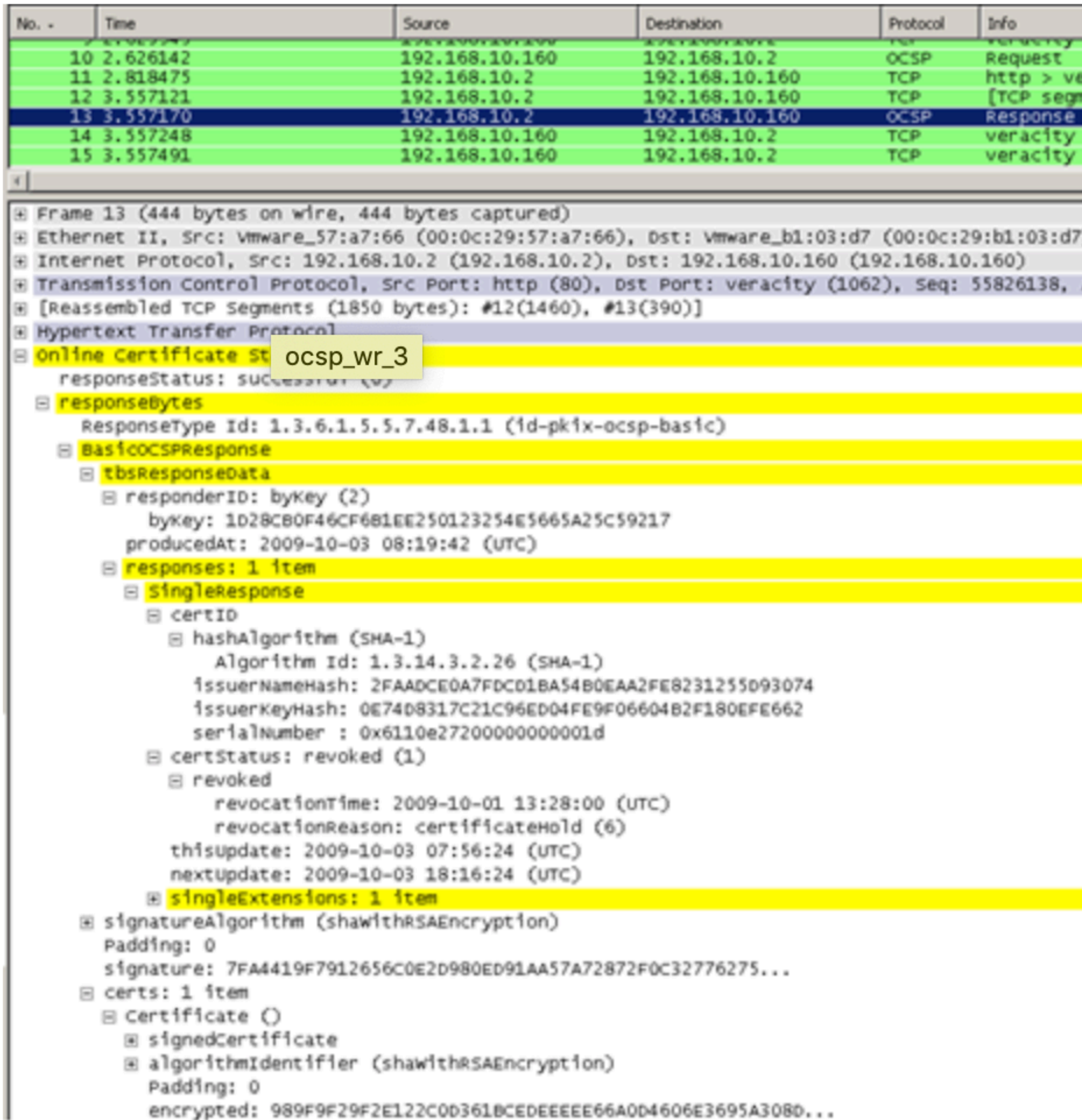
- **OCSP Response (with the wireshark tool)**

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

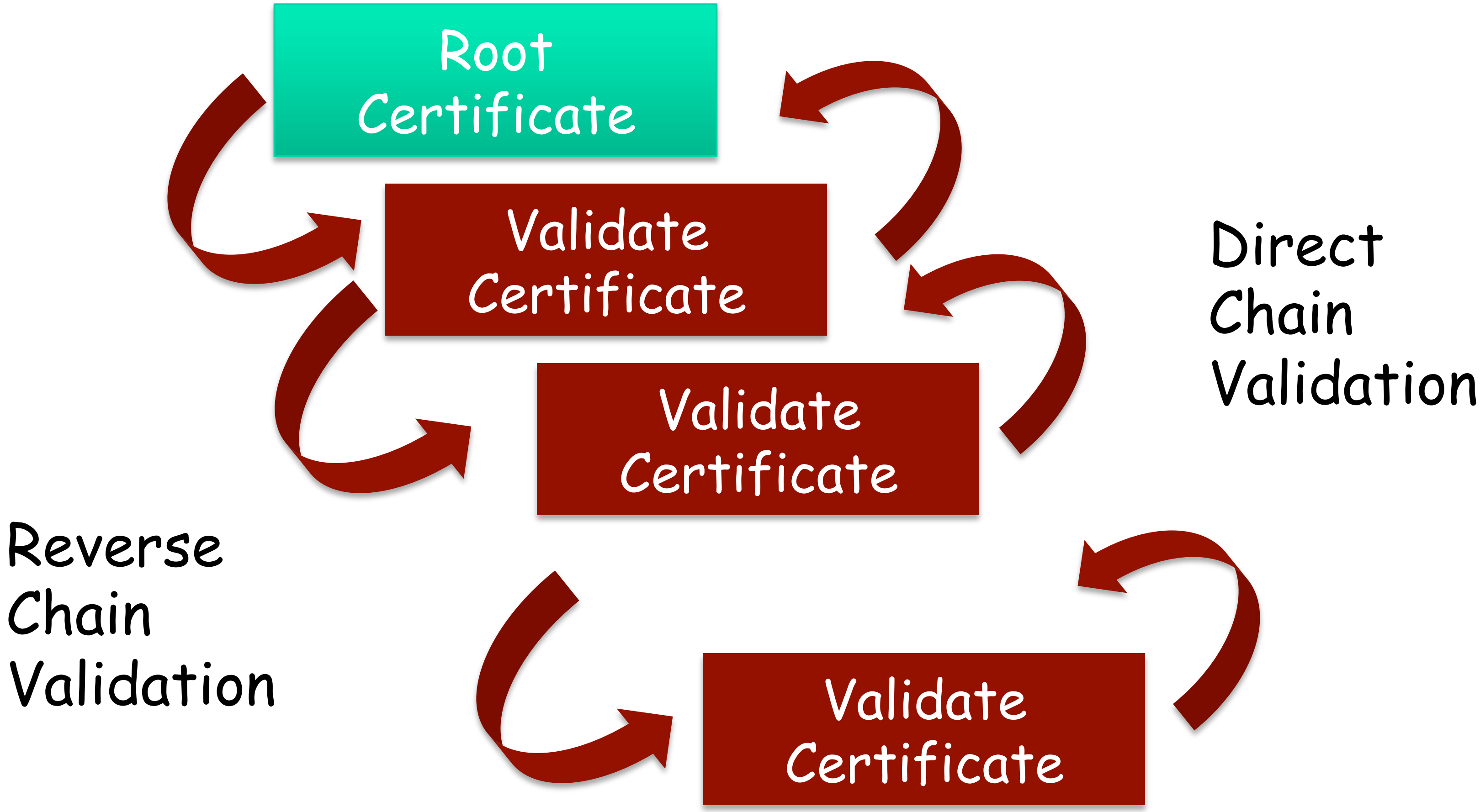
- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Validation can be complex, in a long tail

- Validation of different attributes
  - Subject Name Attributes:
    - Names, DNS names
  - Issuer Name Attributes
    - O, OU, Cname, … Validity
- Validation of critical fields and attributes
  - Keysizes, Key usage, …
  - Extensions: critical attributes and other possible required attributes
    - key usage policy
    - Verification of selected extensions
    - Timestamping
    - CRL endpoints => Look to the more recent issued CRL
    - OCSP endpoints => Possibly validate on the OCSP endpoint
    - …
    - Integrity Fingerprints
- Basic constraints
  - Certificate authority
- Validation of signatures

Validate a Certificate

Chain Validation can be more complex yet in a more long tail (direct and/or reverse)



Programming support: ex., JAVA PKI API
http://docs.oracle.com/javase/8/docs/technotes/guides/security/certpath/CertPathProgGuide.html

Complexity management issues (and usually flaws)

- Architectural weaknesses
- Errors and issues involving certificate authorities and/or management of PKIs
  - Ex., Verification problems in enrolment processes
- Implementation issues
- Cryptographic weaknesses

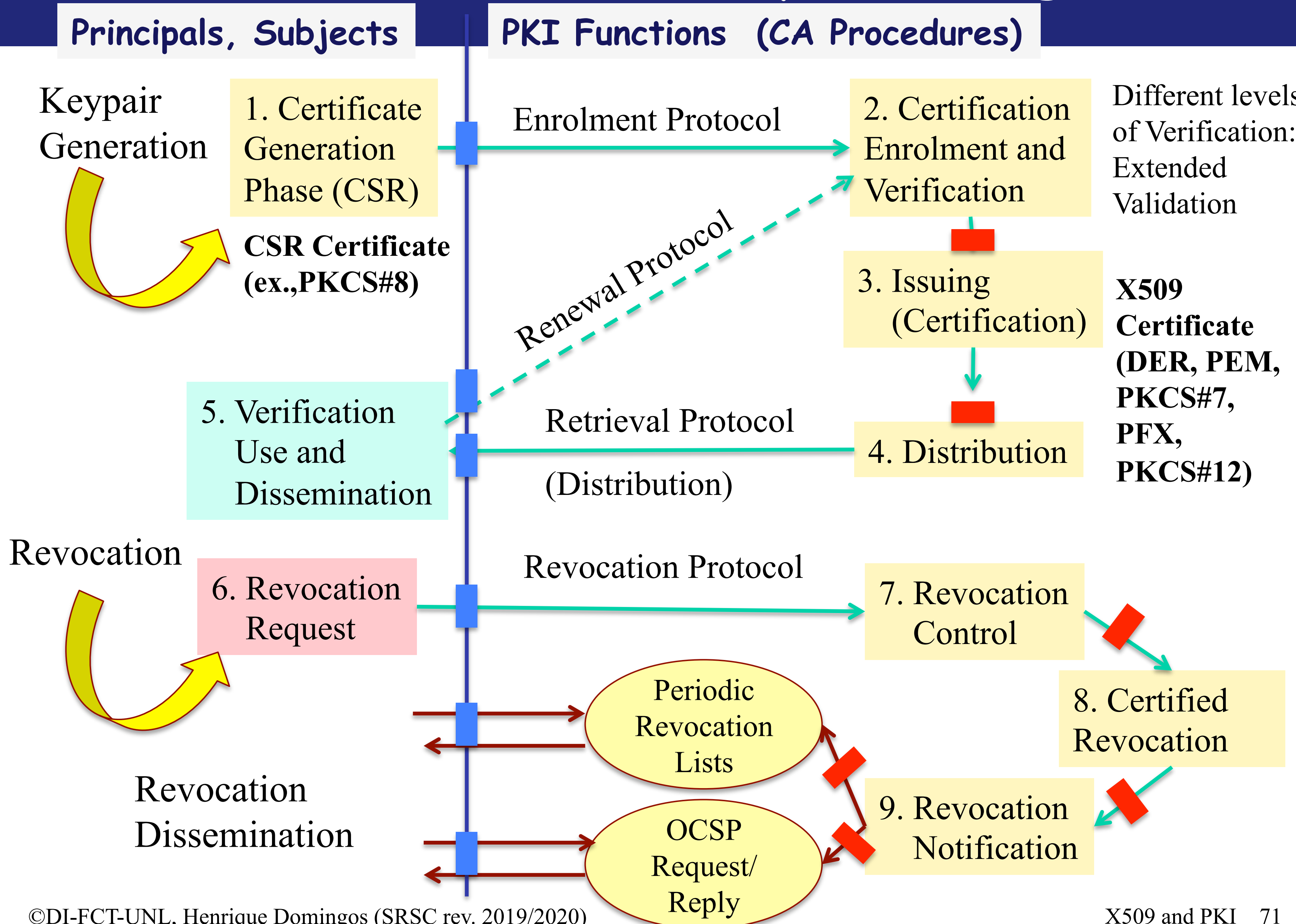
SW Certificates/Certification/Validation weaknesses

- Incorrect verification
- Incomplete verification or limited chain levels
- Implementation Bugs

Outline

- **X509 Authentication**
  - X509 Authentication and Key Management Issues
- **X509 Certificates**
  - X509 and X509 v3 Certificates
  - Life-Cycle Management of X509 Certificates
  - Authentication procedures
  - Forward and reverse certification chains
  - X509 v3 Extensions
  - Revocation
  - The possible long tail of certification chains
- **PKI - Public Key Infrastructure**
  - PKI Standardization and PKIX Management

Remember the X509 Life Cycle Management



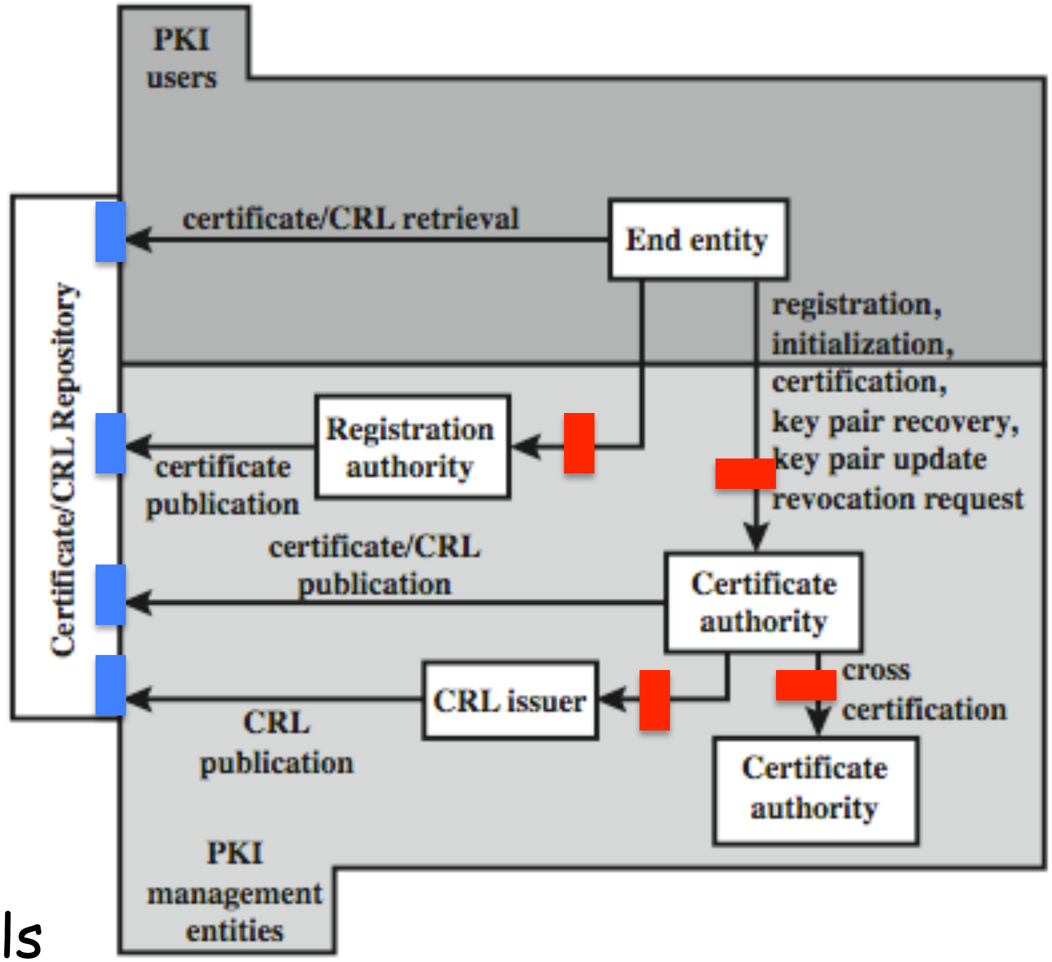
PKI - Public Key Infrastructure

- A Standard Framework Model
  - a set of: HW, SW, People, Rules, Procedures, Policies and Protocols, needed to create, manage, store, distribute and revoke digital certificates
- Objective: enable secure, convenient and efficient acquisition of public keys, promoting strict and well-known specifications
- Coordination by the IETF X509 (PKIX) WG
- Standardized base for compatibility purposes on the above issues in building PKI Platforms
  - Solutions that can also be used by CAs (Certification Authorities) and Ras (Registration Authorities or CA Registrars)

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

Scale and 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:
  - **OCSP**: 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, …
http://datatracker.ietf.org/wg/pkix/

Formats

Certificates has been encoded and/or digitally signed in different formats (defined in RFC 5280 - PKIX) .

See also, for ex: https://en.wikipedia.org/wiki/X.509

Encodings:

- PKCS#10 CSR: Certificate Signed Request format
- PKCS#12, X509v3, PEM, ASN.1, DER or BASE64 encodings
- PKCS#7 format: CRLs - Certificate Revocation Lists

Management of CRLs

- Download and verification
- Can use keytool, KeyStoreExplorer or openssl tools
- Programatically (ex., JAVA, CRL Class, X509CRL SubClass)

https://docs.oracle.com/javase/7/docs/api/java/security/cert/CRL.html

More on 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

Conversions available in some existent tools
See: openssl and keytool:-)))

Example w/ openssl:

- 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-Cycle of Keypairs Public-Key certificates Generation and Management

- See Lab materials (Labs 4.1 and 4.2):
  - Use of keytool
  - Use of openssl
  - Generation-Cycle of:
    - Keypairs
    - Management in Keystores (in different formats)
      - Java Keystores
      - Canonical file formats: PEM, PKCS#12
    - How to generate, manage and use certification chains:
      - CA (root level): Intermediate level … : Leaf level

- See also Lab materials (Lab 5)

Suggested Readings

Suggested Readings:

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