DIGITAL SIGNATURES

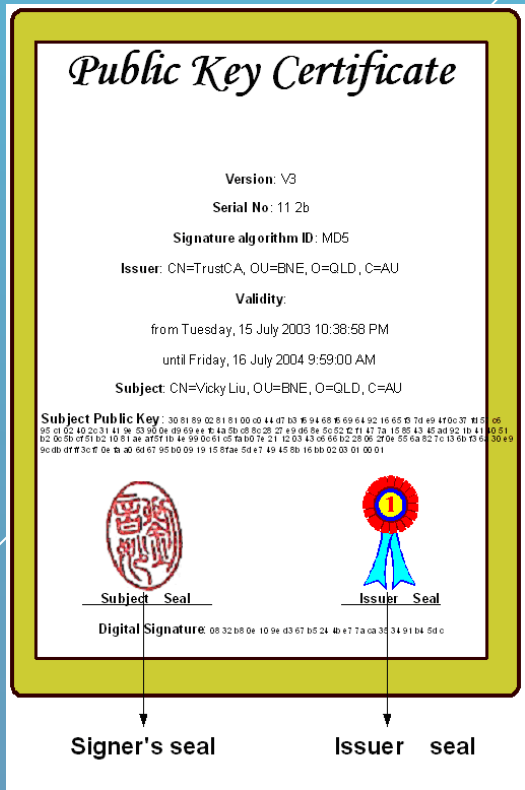
Mukesh Chinta

Asst Prof, CSE, VRSEC

What is a Digital Signature?

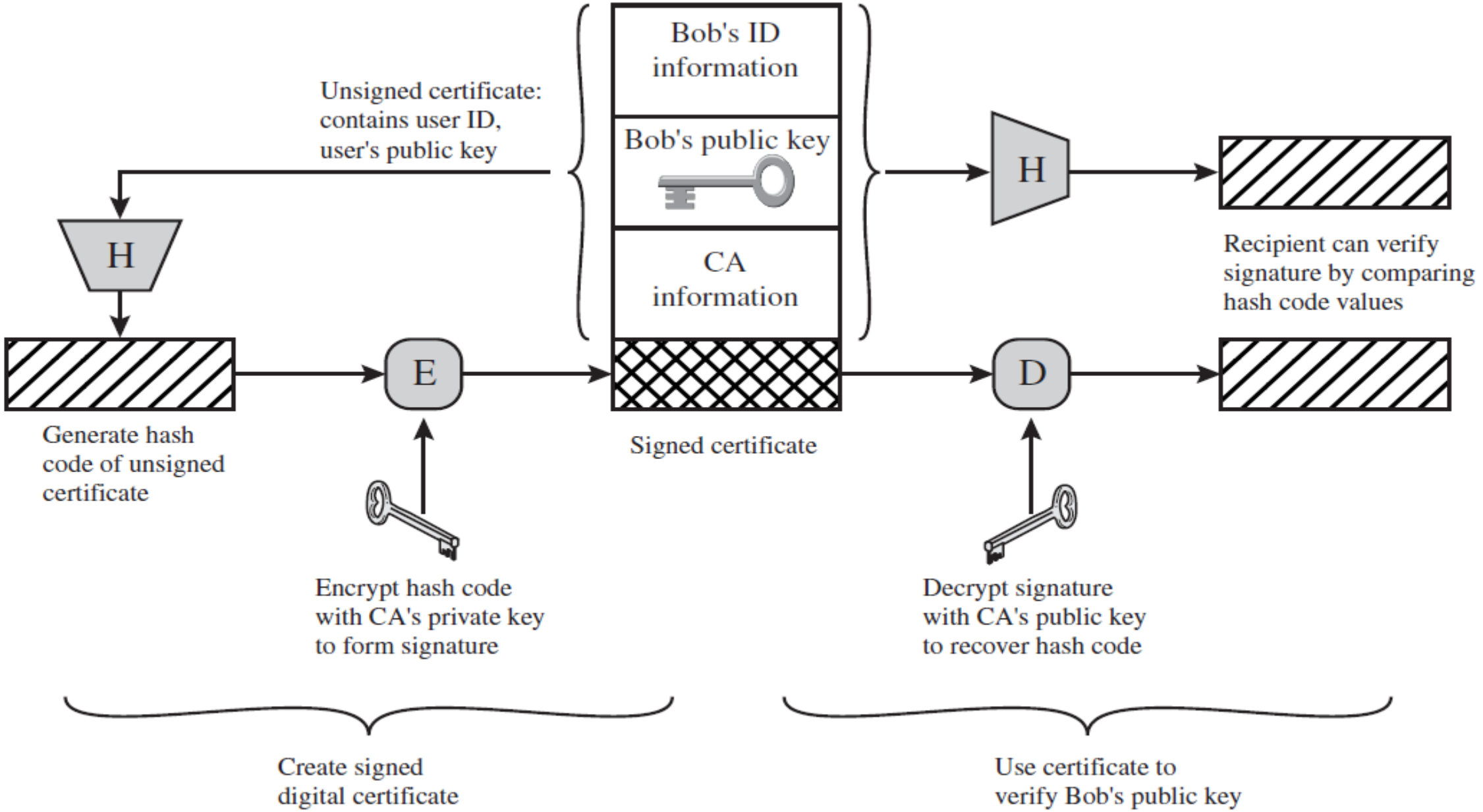
- A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document, and possibly to ensure that the original content of the message or document that has been sent is unchanged.
- Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of other users.
- The main difference from a handwritten signature is that digital signature of a message is intimately connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always looks the same.

X.509 Certificates

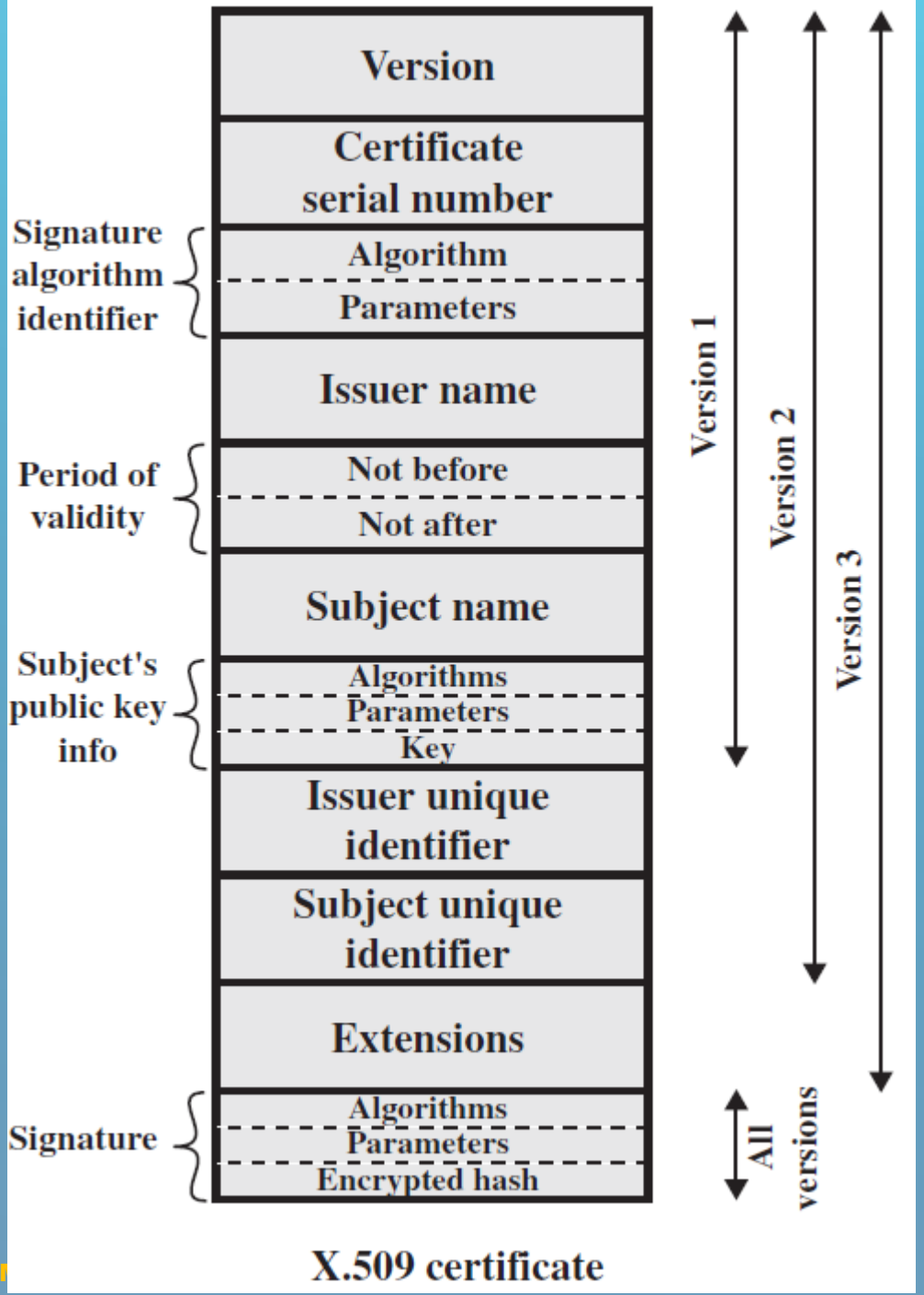


X.509 SCHEME

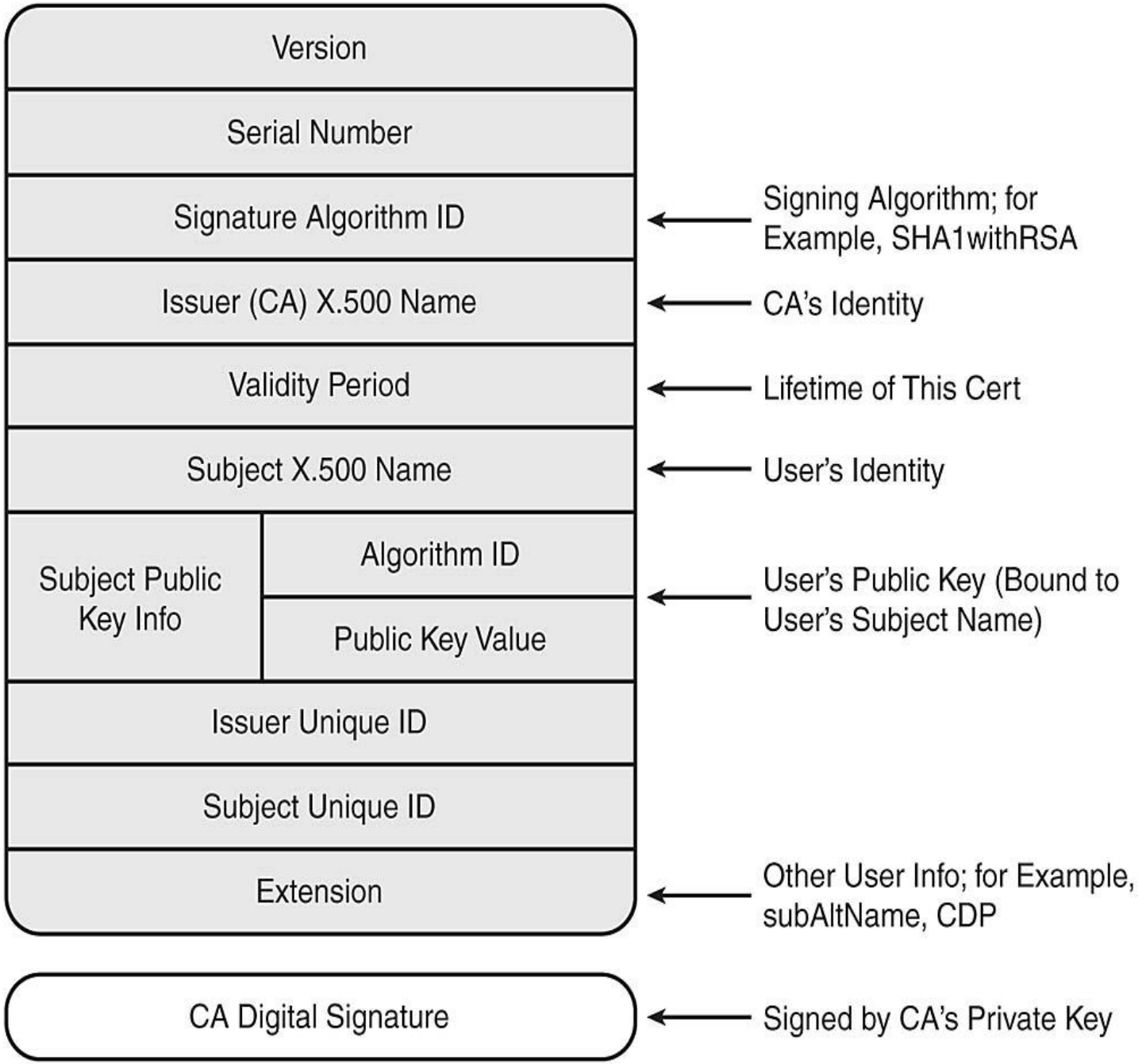
- An X.509 certificate is a digital certificate that uses the widely accepted international X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the user, computer or service identity contained within the certificate.
- An X.509 certificate binds a name to a public key value. The role of the certificate is to associate a public key with the identity contained in the X.509 certificate.
- To prevent impersonation, all certificates must be signed by a *certification authority* (CA). A CA is a trusted node that confirms the integrity of the public key value in a certificate.
- A CA signs a certificate by adding its *digital signature* to the certificate. A digital signature is a message encoded with the CA's private key. The CA's public key is made available to applications by distributing a certificate for the CA. Applications verify that certificates are validly signed by decoding the CA's digital signature with the CA's public key.
- X.509 was initially issued in 1988 and was subsequently revised in 1993. A third version was issued in 1995 and was revised in 2000.
- The X.509 standard is based on use of public-key cryptography and digital signatures. It does not dictate the use of a specific algorithm but recommends RSA.



Public-Key Certificate Use



X.509 certificate



- **Version**: Differentiates among successive versions of the certificate format; the default is version 1.
- **Serial number:** An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier**: An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier

The standard uses the following notation to define a certificate:

$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$

where

$Y\langle\langle X\rangle\rangle$ = the certificate of user X issued by certification authority Y

$Y\{I\}$ = the signing of I by Y. It consists of I with an encrypted hash code appended

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

T^A = period of validity of the certificate

OBTAINING A CERTIFICATE

- Any user with access to CA can get any certificate from it
- No party other than the CA can modify a certificate without being detected

Because certificates cannot be forged, they can be placed in a public directory without much effort to safeguard them.

CA HIERARCHY

- If both users share a common CA then they are assumed to know its public key and a user can directly transmit his/her certificate to others.
- For a large community of users, its not feasible that all users subscribe to the same CA. Hence CA's must form a hierarchy.
- Use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (reverse)
- Each client trusts parents certificates
- Enable verification of any certificate from one CA by users of all other CAs in hierarchy

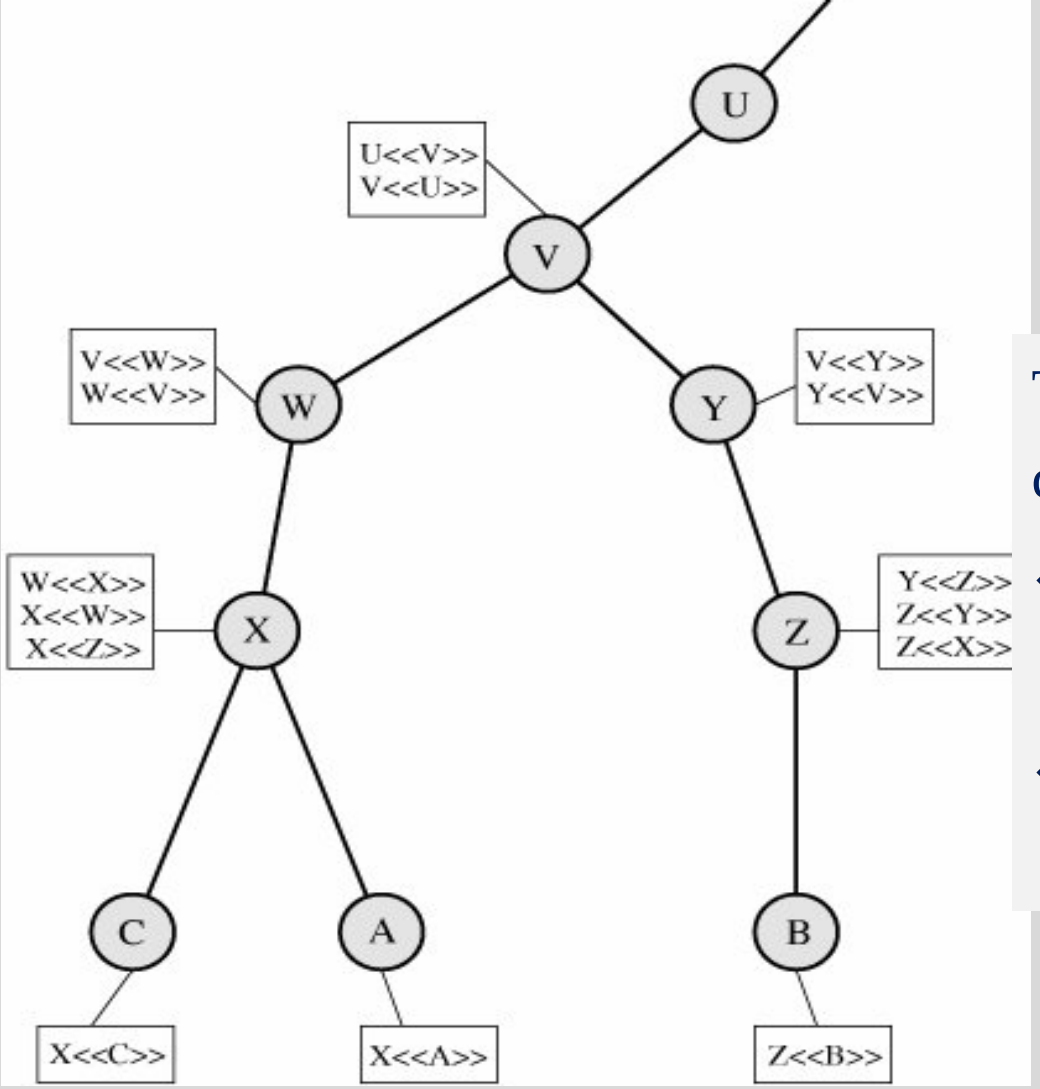
A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2. If A does not securely know the public key of X_2, then B's certificate, issued by X_2, is useless to A. A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

- Step 1: A obtains from the directory the certificate of X_2 signed by X_1. Because A securely knows X_1's public key, A can obtain X_2's public key from its certificate and verify it by means of 's signature on the certificate.
- Step 2: A then goes back to the directory and obtains the certificate of B signed by X_2. Because A now has a trusted copy of X_2's public key, A can verify the signature and securely obtain B's public key. A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

$$X1\langle\langle X2\rangle\rangle X2\langle\langle B\rangle\rangle$$

In the same fashion, B can obtain A's public key with the reverse chain:

$$X2\langle\langle X1\rangle\rangle X1\langle\langle A\rangle\rangle$$



The directory entry for each CA includes two types of certificates:

- Forward certificates: Certificates of X generated by other CAs
- Reverse certificates: Certificates generated by X that are the certificates of other CAs

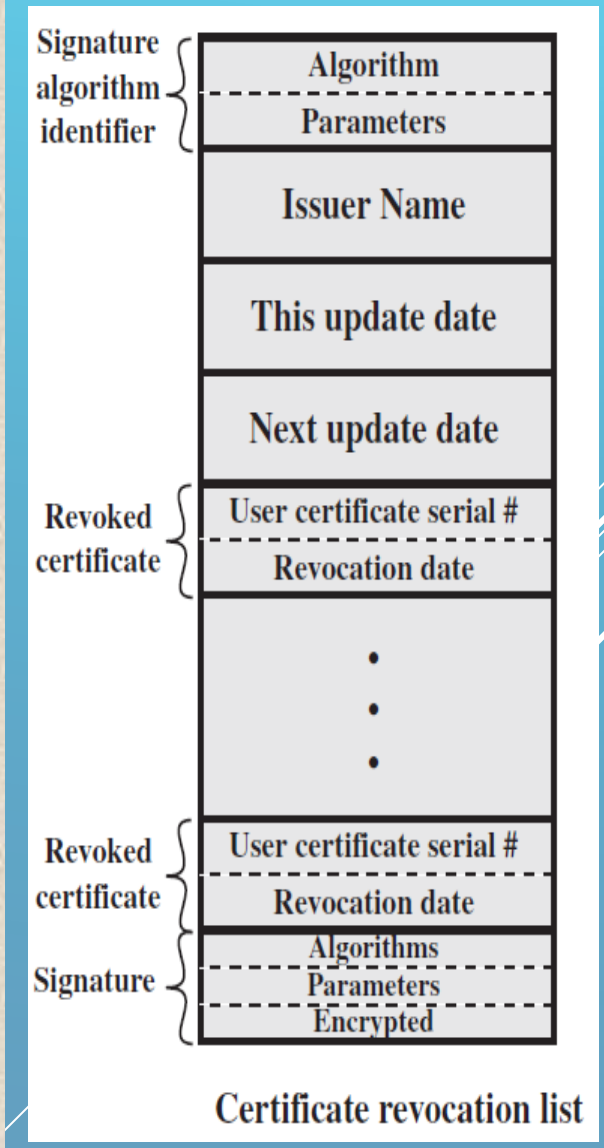
Track chains of certificates:

A acquires B certificate using chain: X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

B acquires A certificate using chain: Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

CERTIFICATE REVOCATION

- Each certificate has a period of validity upon later a new certificate is issued.
- A need may arise to revoke before expiry, eg:
  1. User's private key is compromised
  2. User is no longer certified by this CA
  3. CA's certificate is compromised
- CA's maintain list of revoked certificates
  - *Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate*
- When a user receives a certificate in a message, the user must determine whether the certificate has been revoked.



Certificate revocation list

X.509 VERSION 3

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. Few limitations of Version 2 are:

- The Subject field is inadequate to convey the identity of a key owner to a public-key user.
- The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internet-related identification
- There is a need to indicate security policy information. This enables a security application or function, such as IPSec, to relate an X.509 certificate to a given policy.
- There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
- It is important to be able to identify different keys used by the same owner at different times to support key life cycle management.

X.509V3 EXTENSIONS

X.509 version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an extension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored or not.

The certificate extensions fall into three main categories:

- key and policy information,
- subject and issuer attributes
- certification path constraints.

A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.

KEY AND POLICY INFORMATION

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy. A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements

- Authority key identifier: Identifies the public key to be used to verify the signature on this certificate or CRL.
- Subject key identifier: Identifies the public key being certified. Useful for subject key pair updating.
- Key usage: Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used
- Private-key usage period: Indicates the period of use of the private key corresponding to the public key
- Certificate policies: Certificates may be used in environments where multiple policies apply.
- Policy mappings: Used only in certificates for CAs issued by other CAs. Policy mappings allow an issuing CA to indicate that one or more of that issuer's policies can be considered equivalent to another policy used in the subject CA's domain.

CERTIFICATE SUBJECT AND ISSUER ATTRIBUTES

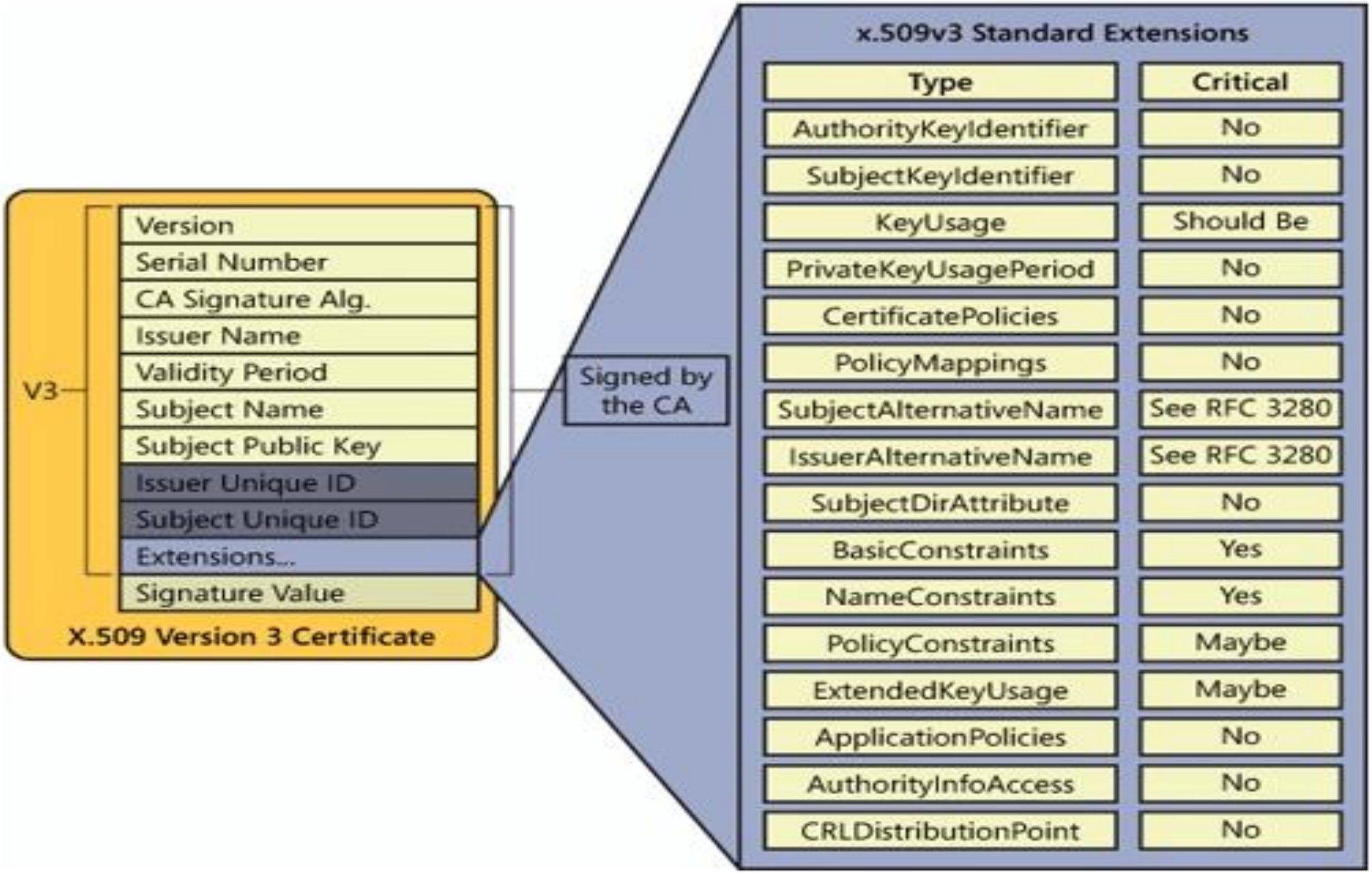
These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity.

- Subject alternative name: Contains one or more alternative names, using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI, and IPSec, which may employ their own name forms.
- Issuer alternative name: Contains one or more alternative names, using any of a variety of forms.
- Subject directory attributes: Conveys any desired X.500 directory attribute values for the subject of this certificate.

CERTIFICATION PATH CONSTRAINTS

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs. The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.

- Basic constraints: Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.
- Name constraints: Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.
- Policy constraints: Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.


V3
Version
Serial Number
CA Signature Alg.
Issuer Name
Validity Period
Subject Name
Subject Public Key
Issuer Unique ID
Subject Unique ID
Extensions...
Signature Value
X.509 Version 3 Certificate
Signed by the CA
x.509v3 Standard Extensions
Type
Critical
AuthorityKeyIdentifier
No
SubjectKeyIdentifier
No
KeyUsage
Should Be
PrivateKeyUsagePeriod
No
CertificatePolicies
No
PolicyMappings
No
SubjectAlternativeName
See RFC 3280
IssuerAlternativeName
See RFC 3280
SubjectDirAttribute
No
BasicConstraints
Yes
NameConstraints
Yes
PolicyConstraints
Maybe
ExtendedKeyUsage
Maybe
ApplicationPolicies
No
AuthorityInfoAccess
No
CRLDistributionPoint
No

X.509 AUTHENTICATION PROCEDURES

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications. All these procedures make use of public-key signatures.

1. **One-Way Authentication**

   One way authentication involves a single transfer of information from one user (A) to another (B). Note that only the identity of the initiating entity is verified in this process, not that of the responding entity. At a minimum, the message includes a timestamp, a nonce, the identity of B and is signed with A's private key. The message may also include information to be conveyed, such as a session key for B.

2. **Two-Way Authentication**

   Two-way authentication thus permits both parties in a communication to verify the identity of the other, thus additionally establishing the above details. The reply message includes the nonce from A, to validate the reply. It also includes a timestamp and nonce generated by B, and possible additional information for A

3. **Three-Way Authentication**

   It includes a final message from A to B, which contains a signed copy of the nonce, so that timestamps need not be checked, for use when synchronized clocks are not available.

1 message (A->B) used to establish

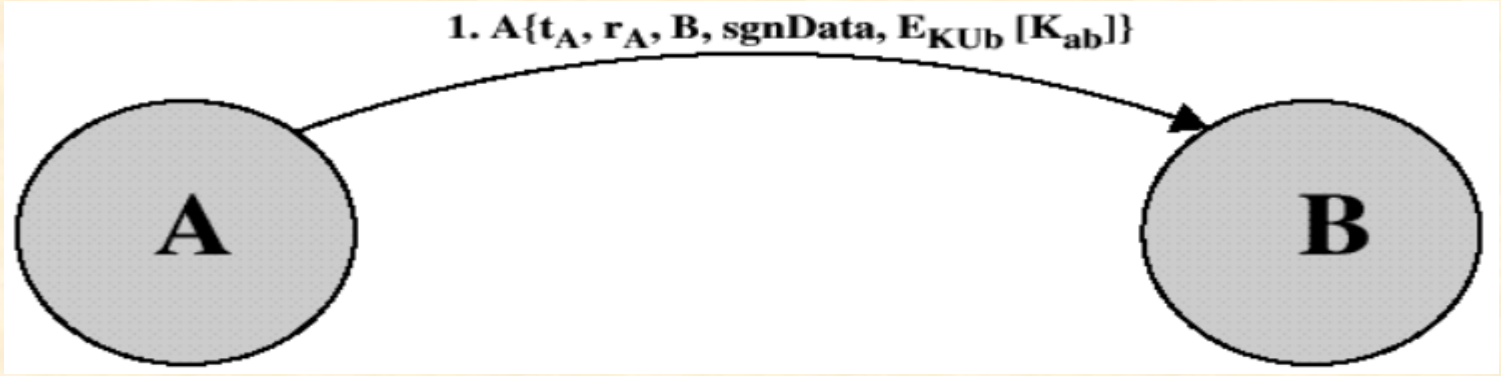
- the identity of A and that message is from A
- message was intended for B
- integrity & originality of message

Ta-timestamp rA=nonce B =identity
sgnData=signed with A's private key

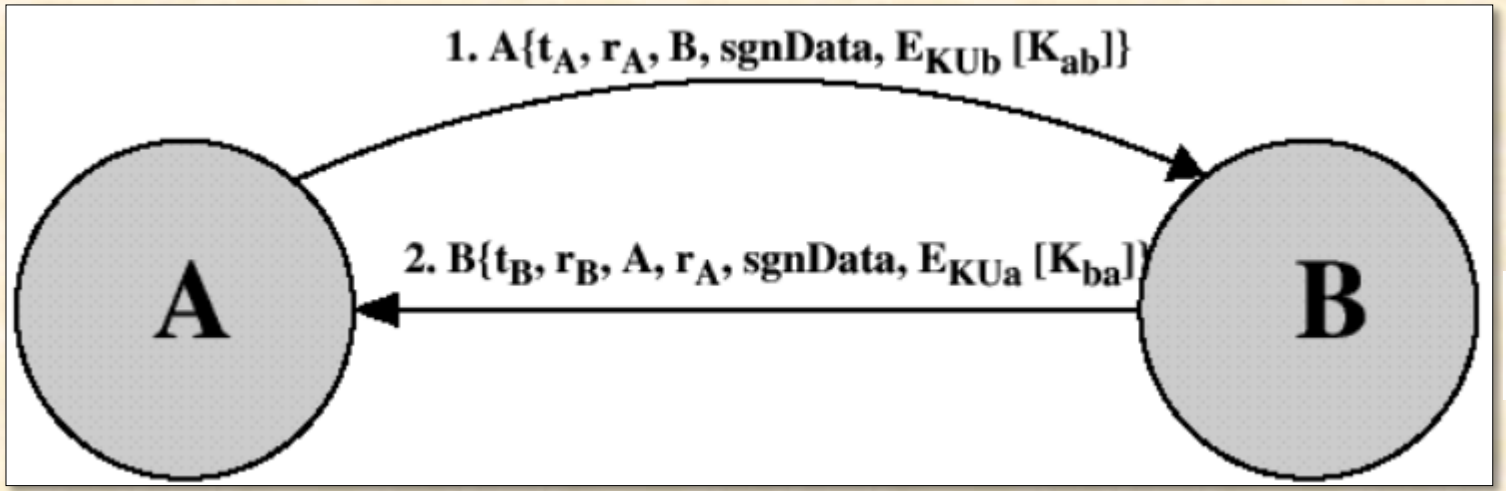
2 messages (A->B, B->A) which also establishes in addition:

- the identity of B and that reply is from B
- that reply is intended for A
- integrity & originality of reply

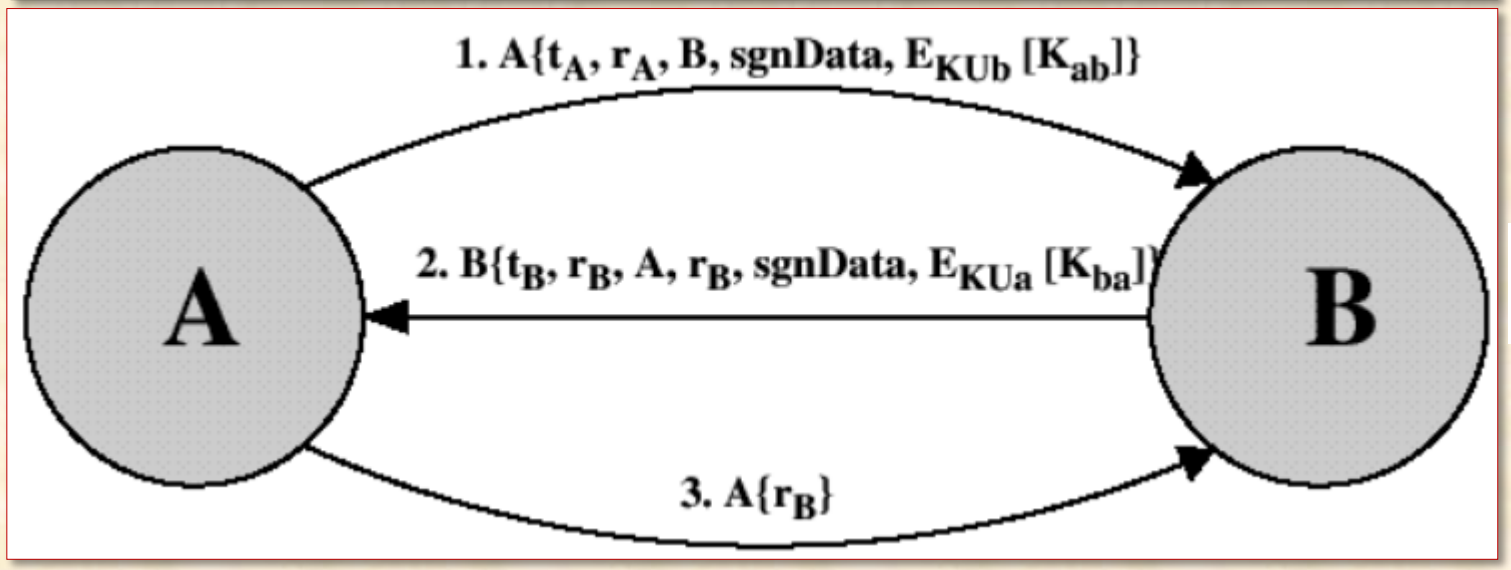
- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks



One-way Authentication



Two-way Authentication



Three-way Authentication

PUBLIC-KEY INFRASTRUCTURE (PKI)

- RFC 2822 (*Internet Security Glossary*) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.

- The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.

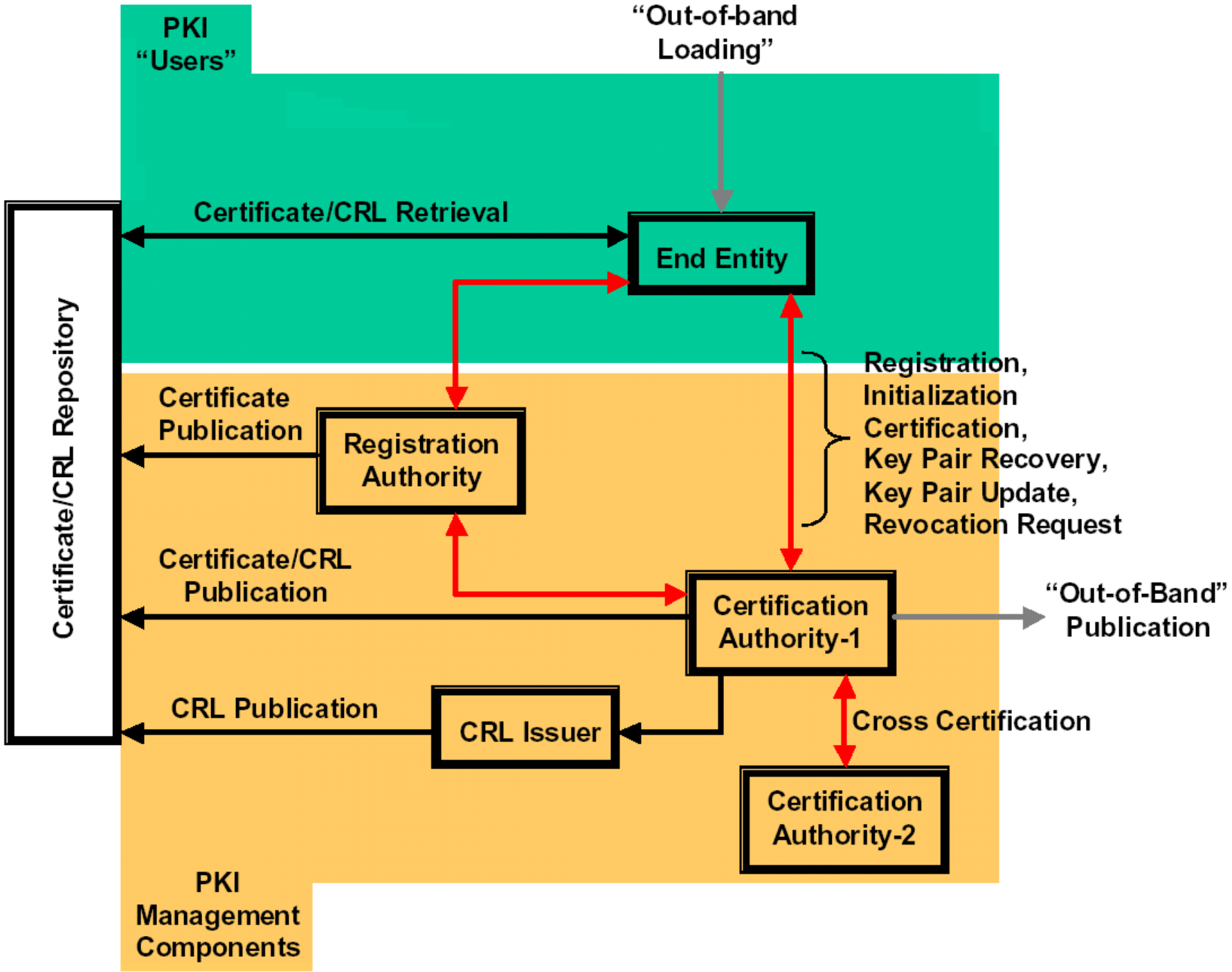
PKIX/PKCS

There are two main standards that help implement PKI on a practical level on the Internet. Both are based on the X.509 certificate standard and establish complimentary standards for implementing PKI. Public Key Infrastructure X.509 (PKIX) and Public Key Cryptography Standards (PKCS) intertwine to define the most commonly used set of standards.

1. PKIX was produced by the Internet Engineering Task Force (IETF). It defines standards for interactions and operations for the four component types - the user (end-entity), *certificate authority* (CA), registration authority (RA), and the repository for certificates and *certificate revocation lists* (CRLs).

2. PKCS, a product of RSA Security, defines many of the lower-level standards for message syntax and cryptographic algorithms.

PKIX Architectural Model



- End entity: A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate. End entities typically consume and/or support PKI-related services.

- Certification authority (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.

- Registration authority (RA): An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the End Entity registration process, but can assist in a number of other areas as well.

- CRL issuer: An optional component that a CA can delegate to publish CRLs.

- Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by End Entities.

PKIX Management protocols are the protocols that are required to support on–line interactions between PKI user and management entities. The possible set of functions that can be supported by management protocols is

- Registration of entity, that takes place prior to issuing the certificate
- Initialization, for example generation of key–pair
- Certification, the issuance of the certificate
- Key–pair recovery, the ability to recover lost keys
- Key–pair update, when the certificate expires and a new key–pair and certificate have to be generated
- Revocation request, when an authorized person advices the CA to include a specific certificate into the revocation list
- Cross-certification, when two CAs exchange information in order to generate a cross–certificate

Public Key Cryptography Standards

PKCS # 1	The RSA encryption standard. This standard defines mechanisms for encrypting and signing data using the RSA public key system.
PKCS # 3	The Diffie-Hellman key-agreement standard. This defines the Diffie-Hellman key agreement protocol.
PKCS # 5	The password-based encryption standard (PBE). This describes a method to generate a Secret Key based on a password.
PKCS # 6	The extended-certificate syntax standard. This is currently being phased out in favor of X509 v3.
PKCS # 7	The cryptographic message syntax standard. This defines a generic syntax for messages which have cryptography applied to it.
PKCS # 8	The private-key information syntax standard. This defines a method to store Private Key Information.
PKCS # 9	This defines selected attribute types for use in other PKCS standards.
PKCS # 10	The certification request syntax standard. This describes a syntax for certification requests.
PKCS # 11	The cryptographic token interface standard. This defines a technology independent programming interface for cryptographic devices such as smartcards.
PKCS # 12	The personal information exchange syntax standard. This describes a portable format for storage and transportation of user private keys, certificates etc.
PKCS # 13	The elliptic curve cryptography standard. This describes mechanisms to encrypt and sign data using elliptic curve cryptography.
PKCS # 14	This covers pseudo random number generation (PRNG). This is currently under active development.
PKCS # 15	The cryptographic token information format standard. This describes a standard for the format of cryptographic credentials stored on cryptographic tokens.