CHAPTER THREE

Digital Signatures and Digital Certificates

Outline

- Message authentication
- Security requirements
- Digital Signature Using Public Key
- Digital Signature Using Message Digest
- Digital certificates
- Distribution of public keys
- Direct key exchange protocols
- Authenticating users and their public keys with certificates signed by Certificate Authorities (CA)

Message Authentication

- Up till now, we have been concerned with protecting message content (i.e secrecy) by encrypting the message.

- Will now consider:
  - how to protect message integrity (ie protection from modification), as well as
  - confirming the identity of the sender.

- generally this is the problem of message authentication,
- In eCommerce applications it is more important than secrecy.

Message Authentication…

- Message authentication is concerned with:
  - Protecting the integrity of a message
  - Validating identity of originator
  - Non-repudiation of origin (dispute resolution)

- Electronic equivalent of a signature on a message.
- First, we will consider the security requirements
- Then two alternative functions used:
  - message encryption
  - hash function

Security Requirements

- In the context of communications across a network, the following attacks can be identified:
- Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.
- Traffic analysis: Discovery of the pattern of traffic between parties.
- Masquerade: Insertion of messages into the network from a fraudulent source.
  - This includes the creation of messages by an opponent that are supposed to come from an authorized entity.

Security Requirements

- Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.
- Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
- Timing modification: Delay or replay of messages.
- Source repudiation: Denial of transmission of message by source.
- Destination repudiation: Denial of receipt of message by destination.

Security Requirements

- disclosure
- traffic analysis

Belongs to message confidentiality, and are handled using the encryption techniques already discussed.

- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

- The remaining requirements belong in the realm of message authentication.
- This addresses the issue of ensuring that a message comes from the assumed source and has not been altered.
- The use of a digital signature can also address issues of repudiation.
- It may also address sequencing and timeliness.

Message Encryption

- Message encryption by itself also provides a measure of authentication.

- If symmetric encryption is used then:
  - receiver know sender must have created it
  - since only sender and receiver know the key used
  - Encryption of a message by a sender's private key also provides a form of authentication.

  - E.g. DES,3DES,...

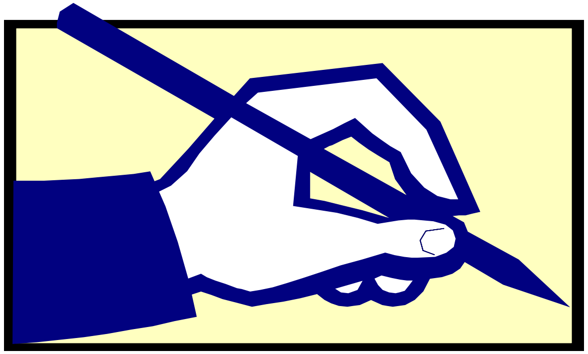
Message Encryption…

- If public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - however if
    - sender **signs** message using their private-key
    - then encrypts with recipients public key
    - have both secrecy and authentication
  - E.g. RSA

Authentication using session-key

- If a message is being encrypted using a session key known only to the sender and receiver, then the message may also be authenticated.
  - since only sender or receiver could have created it
  - any interference will corrupt the message
  - but this does not provide non-repudiation since it is impossible to prove who created the message
- E.g. DH

Digital Signature



- Digital signatures allow the world to verify I created a piece of data
  - e.g. email, code

- Digital signatures are created by encrypting a hash of the data with my private key
- The resulting encrypted data is the signature
- This hash can then only be decrypted by my public key

Why Digital Signatures?

- To provide Authenticity, Integrity and Non-repudiation to electronic documents

- To use the Internet as the safe and secure medium for any data exchange between two users

Digital Signature using pubic key cryptography (RSA)

- RSA may be used directly as a digital signature scheme
  - given an RSA scheme {(e,n), (d,n)}
- To **sign** a message, compute:
  - $s = m^d (\text{mod } n)$
- To **verify** a signature, compute:
  - $m = s^e (\text{mod } n) = m^{e.d} (\text{mod } n)$
- Thus know the message was signed by the owner of the public-key.
- More commonly use a **hash function** to create a separate Message Digest (MD) which is then signed.

Hash Function Properties

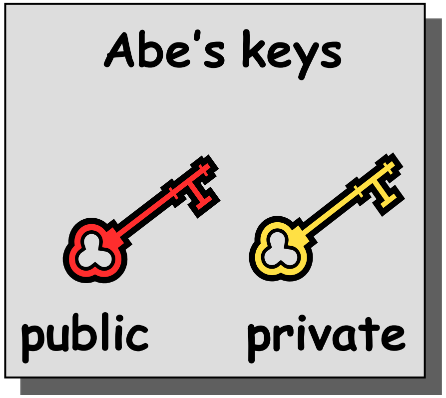
- a Hash Function produces a fingerprint of some file/message/data

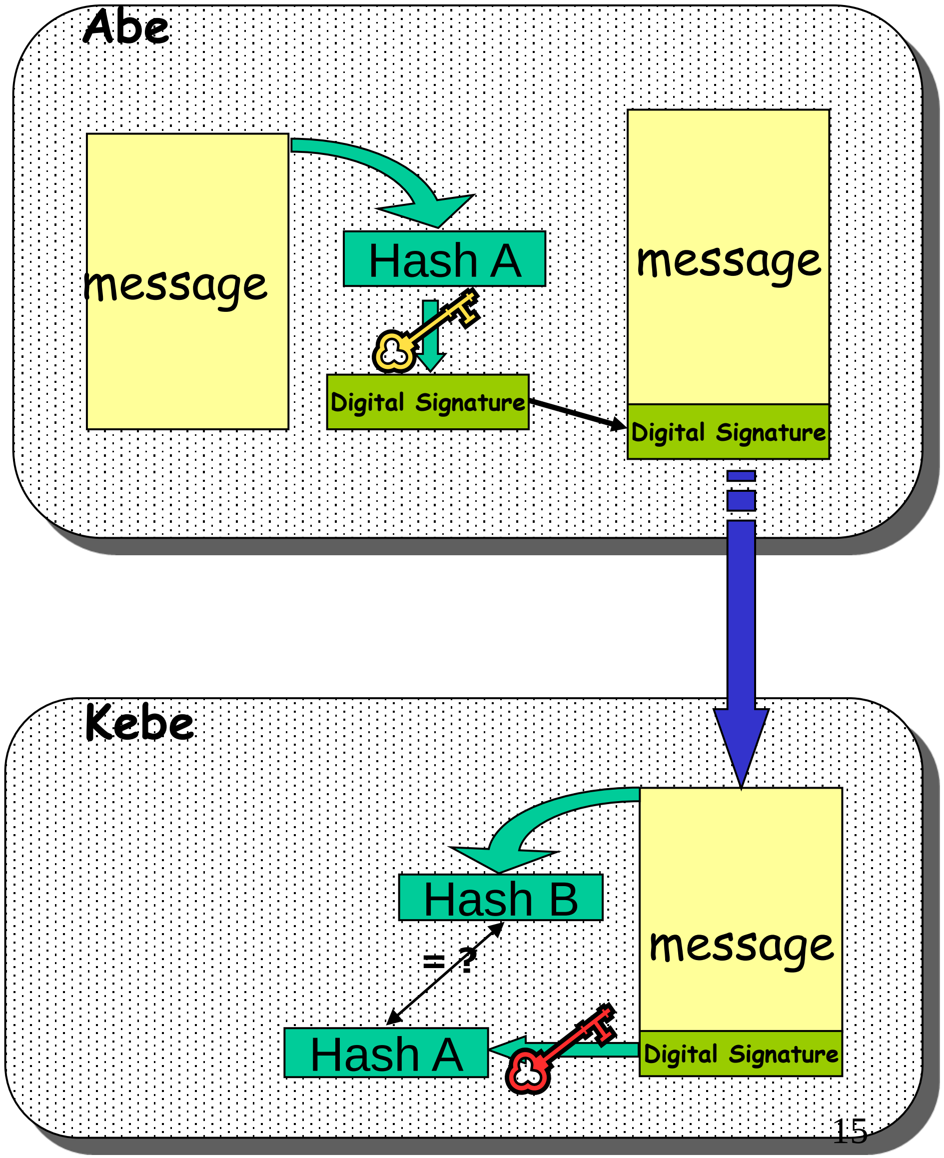
  h = H(M)

  – condenses a variable-length message M
  – to a fixed-sized fingerprint h
    - the length of h must be at least 128 bits.
    - given M, it **must be easy** to calculate H(M) = h
    - given h, it **must be difficult** to calculate $M = H^{-1}(h)$
    - given M, it **must be difficult** to find M' such that H(M) = H(M')
- Examples:
  - **SNEFRU**: hash of 128 or 256 bits;
  - **MD4/MD5**: hash of 128 bits;
  - **SHA** : hash of 160 bits.

Digital Signatures – Authentication using hash function

- Abe calculates the *hash* of the message: *a 128 bit value based on the content of the message*
- Abe encrypts the hash using his *private* key: the encrypted hash is the *digital signature*.
- Abe sends the signed message to Kebe.
- Kebe calculates the hash of the message
- Decrypts A with Abe's *public* key.
- If hashes equal:
  1. hash A is from Abe's private key;
- 2. message wasn't modified;





Digital Certificates

- Abe's digital signature is useful to Kebe if:
  1. Abe's private key is not compromised – keep these safe!!!
  2. Kebe has Abe's public key

- How can Kebe be sure that Abe's public key is really Abe's public key and not someone else's?
  - A *third party* establishes the correspondence between public key and owner's identity.
  - Both Kebe and Abe trust this third party

The "third party" is called a *Certification Authority* (CA).

Key Distribution: Problems

- Distribution of a key is a difficult matter!

- For a symmetric cryptosystem, the initial key must be communicated along a secured channel(?)

- For public key, we need a body that certifies the public key is that of the party we need to communicate with

- Solution: Certification/Certificate Authority (CA) that signs (certifies) the public key

Key Management

- public-key encryption helps address key distribution problems
- have two aspects of this:
  - distribution of public keys
  - use of public-key encryption to distribute secret keys

Using public keys to exchange secrete session keys

- Public key cryptography fulfills an extremely important role in the overall design and operation of secure computer networks.
  - because it leads to superior protocols for managing and distributing secret session keys
  - that can subsequently be used for the encryption of actual message content using symmetric-key algorithms such as 3DES, AES, etc..

Using public keys to exchange secrete session keys…

- If two parties A and B are sure about each other's identity, and can be certain that a third party will not masquerade,
  - they can use a simple and direct key exchange protocol for exchanging a secret session key. (**Scenario 1)**
- In general, such protocols will not require support from any coordinating or certificating agencies.
- The key exchange protocols are more complex for security that provides a higher level of either one-sided or mutual authenti-cation between two communicating parties.
- These protocols usually involve Certificate Authorities.

Scenario 2

A direct key exchange protocol (Scenario 1)

- If each of the two parties A and B has full confidence that a message received from the other party is indeed authentic,
- the exchange of the secret session key for a symmetric-key based secure communication link can be carried out with a simple protocol such as the one described below:
  - Wishing to communicate with *B*, *A* generates a public/private key pair $\{PU_A, PR_A\}$ and
  - *A* transmits **an unencrypted message** to *B* consisting of PU_A and *A's* identifier, ID_A (which can be *A's* IP address).
  - Note that PU_A is party A's public key and PR_A the private key.

A direct key exchange protocol…

- Upon receiving the message from A, B generates and stores a secret session key K_S.
- Next, B responds to A with the secret session key K_S.
  - This response to A is encrypted with A's public key PU_A.
  - We can express this message from B to A as $E(PU_A, K_S)$.
- Obviously, since only A has access to the private key PR_A, only A can decrypt the message containing the session key.
- A decrypts the message received from B with the help of the private key PR_A and retrieves the session key K_S.
- A discards both the public and private keys, PU_A and PR_A, and B discards PU_A.
- Now A and B can communicate confidentially with the help of the session key K_S.

A direct key exchange protocol…

- However, this protocol is vulnerable to the **man-in-the-middle attack** by an adversary E who is able to intercept messages between *A* and *B*.

- This is how this attack takes place:
  - When *A* sends the very first unencrypted message consisting of PU_A and ID_A, E intercepts the message. (Therefore, *B* never sees this initial message.)
  - The adversary E generates its own public/private key pair $\{PU_E, PR_E\}$ and transmits $\{PU_E, ID_A\}$ to *B*.
  - Assuming that the message received came from *A*, *B* generates the secret key K_S, encodes it with PU_E, and sends it back to *A*.

A direct key exchange protocol…

- This transmission from *B* is again intercepted by E, who for obvious reasons is able to decode the message.
- E now encodes the secret key K_S with *A's* public key PU_A and sends the encoded message back to *A*.
- *A* retrieves the secret key and, not suspecting any foul play, starts communicating with *B* using the secret key.
- E can now successfully eavesdrop on all communications between *A* and *B*.

Certificate Authorities for authenticating your public key

- A **certificate** issued by a **certificate authority (CA)** authenticates your public key.
  - A certificate is your public key signed by the CA’s private key.
- A certificate assigned to a user consists of:
  - The user’s public key,
  - the identifier of the key owner,
  - a time stamp (in the form of a period of validity), etc.,
- The whole block encrypted with the CA’s private key.
- Encryption of the block with the CA’s private key is referred to as the CA having signed the certificate.

Certificate Authorities for authenticating your public key…

- We may therefore express a certificate issued to party *A* by

$$C_A = E\,(PR_{CA}, [T, ID_A, PU_A])$$

  - where PR_{CA} is the private key of the Certificate Authority,
  - T the expiration date/time for the *A's* public key PU_A that is being validated by the **CA**, and
  - ID_A the party *A's* identifier.

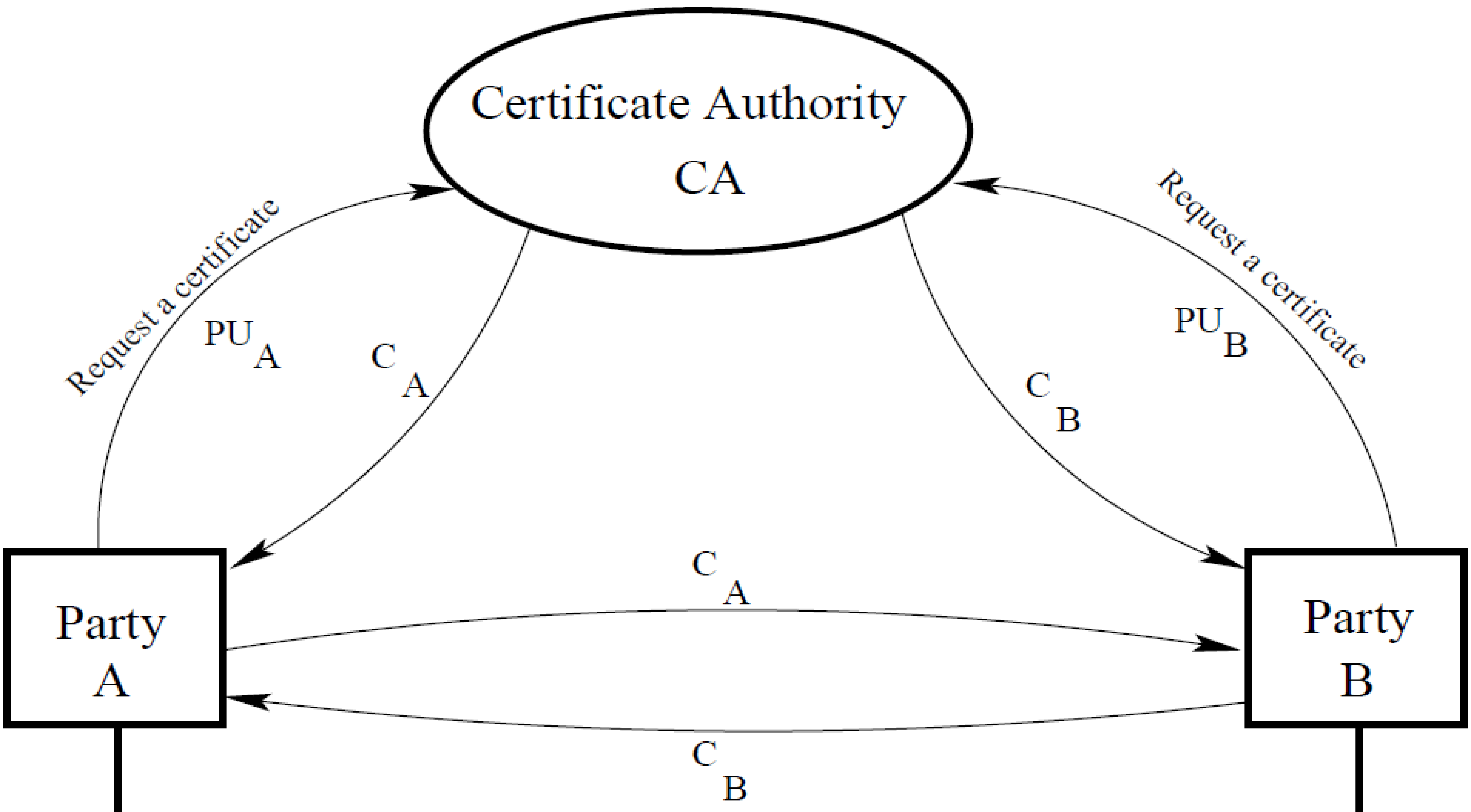
- Subsequently, when party *A* presents his/her certificate to party *B*, *B* can verify the legitimacy of the certificate by decrypting it with the CA's public key.
- **Successful decryption authenticates both:**
  - the certificate supplied by *A* and
  - *A's public key*.

Certificate Authorities for authenticating your public key…

- Having established the certificate’s legitimacy,
  - having authenticated A, and
  - having acquired A’s public key,
- B responds back to A with his own certificate.
- A processes B’s certificate in the same manner as B processed A’s certificate.
- This exchange results in A and B acquiring *authenticated public keys* for each other.
- NOTE
  - Each of the two parties A and B acquires the other party’s public key not directly but through the other party’s certificate.
  - For greater security, B can ask CA to verify that the certificate received from A is currently valid, that is, it has not been revoked.

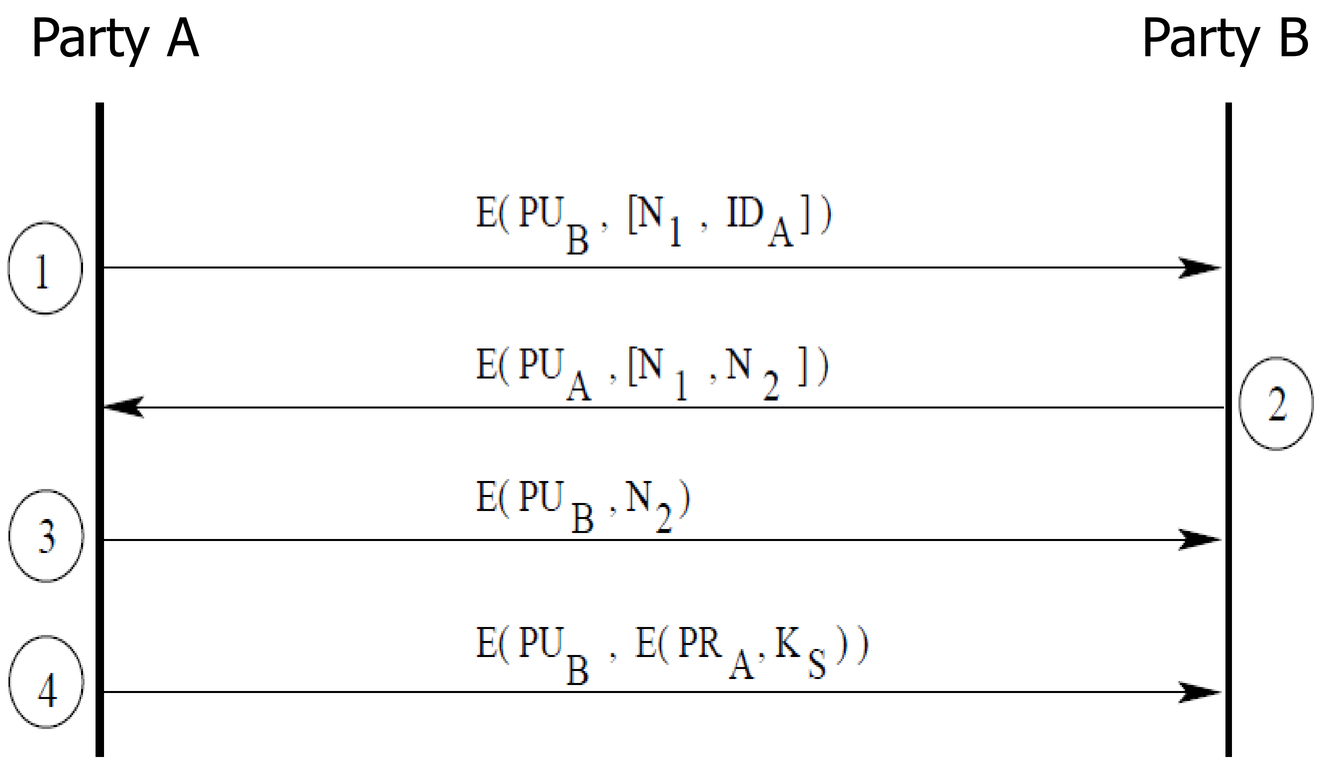
Parties A and B want to establish a secure and authenticated communication link

- Party A initiates a request for the link



Using Authenticated Public Keys to Exchange a Secret Session Key

- Having acquired the public keys, the two parties A and B then proceed to exchange a secret session key.



Using Authenticated Public Keys to Exchange a Secret Session Key…

- A uses B's public key PU_B to encrypt a message that contains A's identifier ID_A and a nonce N_1 as a transaction identifier.
- A sends this encrypted message to B.
- This message can be ex-pressed as

$$E\,(PU_B,\ [N_1,\ ID_A])$$

- B responds back with a message encrypted using A's public key PU_A, the message containing A's nonce N_1 and new nonce N_2 from B to A.
- The structure of this message can be expressed as

$$E\,(PU_A,\ [N_1,\ N_2])$$

Using Authenticated Public Keys to Exchange a Secret Session Key…

- Since only B could have decrypted the first message from A to B, the presence of the nonce N_1 in this response from B further assures A that the responding party is actually B
  - since only B could have decrypted the original message containing the nonce N_1.

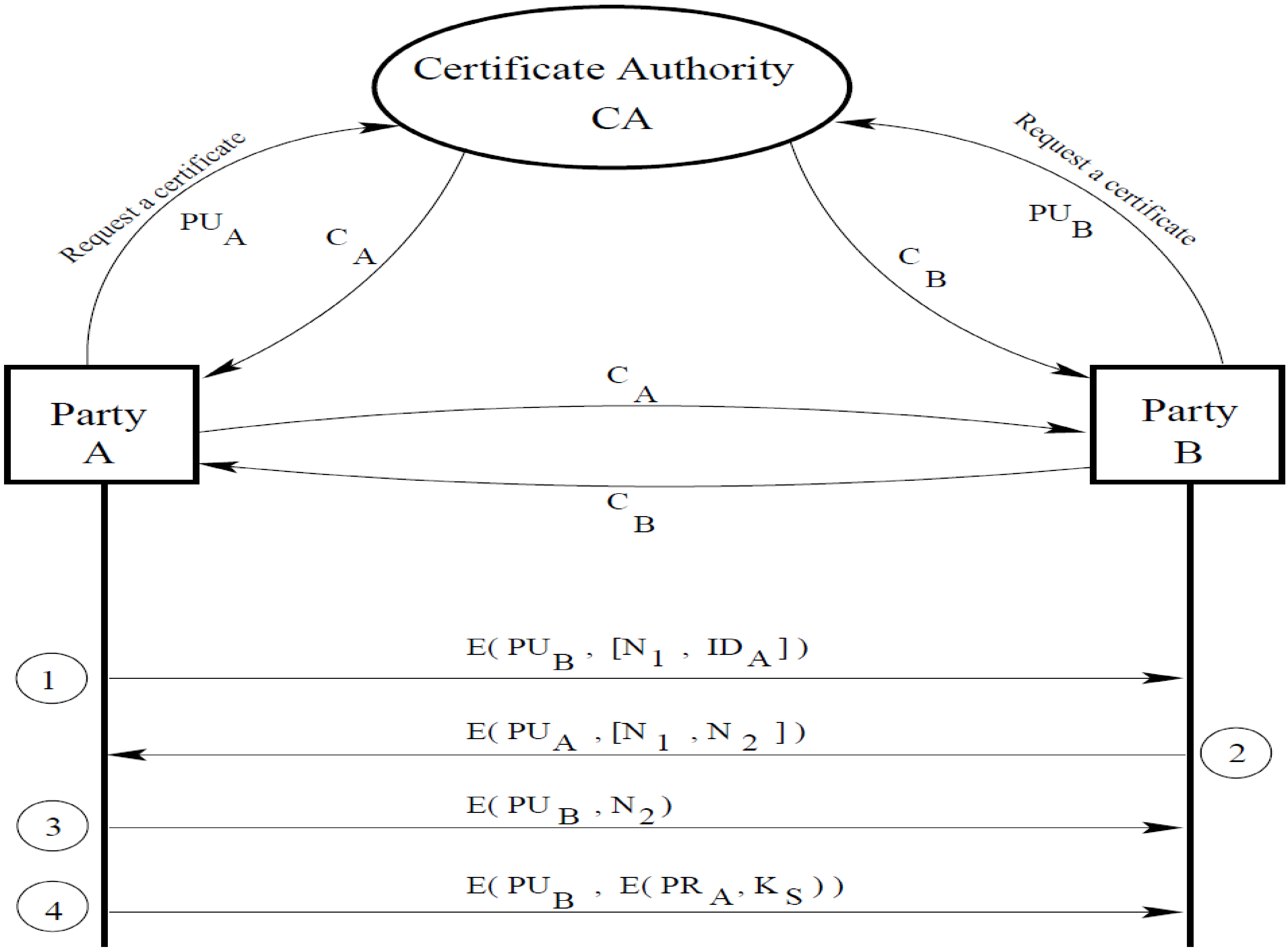
- A now selects a secret session key K_S and sends B the following message

$$M = E\,(PU_B,\ E\,(PR_A,\ K_S))$$

- **NOTE**
- A encrypts the secret key K_S with his/her own private key PR_A before further encrypting it with B's public key PU_B.
- Encryption with A's private key makes it possible for B to authenticate the sender of the secret key.

Using Authenticated Public Keys to Exchange a Secret Session Key…

- Further encryption with B’s public key means that only B will be able to read it.
- B decrypts the message first with its own private key PR_B and then recovers the secret key by applying another round of decryption using A public key PU_A.


Certificate Authority
CA
Request a certificate
PU A
C A
C B
PU B
Request a certificate
Party
A
Party
B
C A
C B
1
E(PU B , [N 1 , ID A])
2
E(PU A , [N 1 , N 2])
3
E(PU B , N 2)
4
E(PU B , E(PR A , K S))

The X.509 Certificate format standard for Public Key Infrastructure

- The set of standards related to the creation, distribution, use, and revocation of digital certificates is referred to as the **Public Key Infrastructure** (**PKI**).
- **X.509** is one of the PKI standards.
  - It is this standard that specifies the format of digital certificates.
- The X.509 standard is described in the IETF document RFC 5280, RFC 6818.
- The X.509 standard is based on a strict hierarchical organization of the CAs in which the trust can only flow downwards.
- The CAs at the top of the hierarchy are known as **root CAs**.
- The CAs below the root are generally referred to as intermediate-level CAs.

Role of a PKI

- Create…

  … Manage…

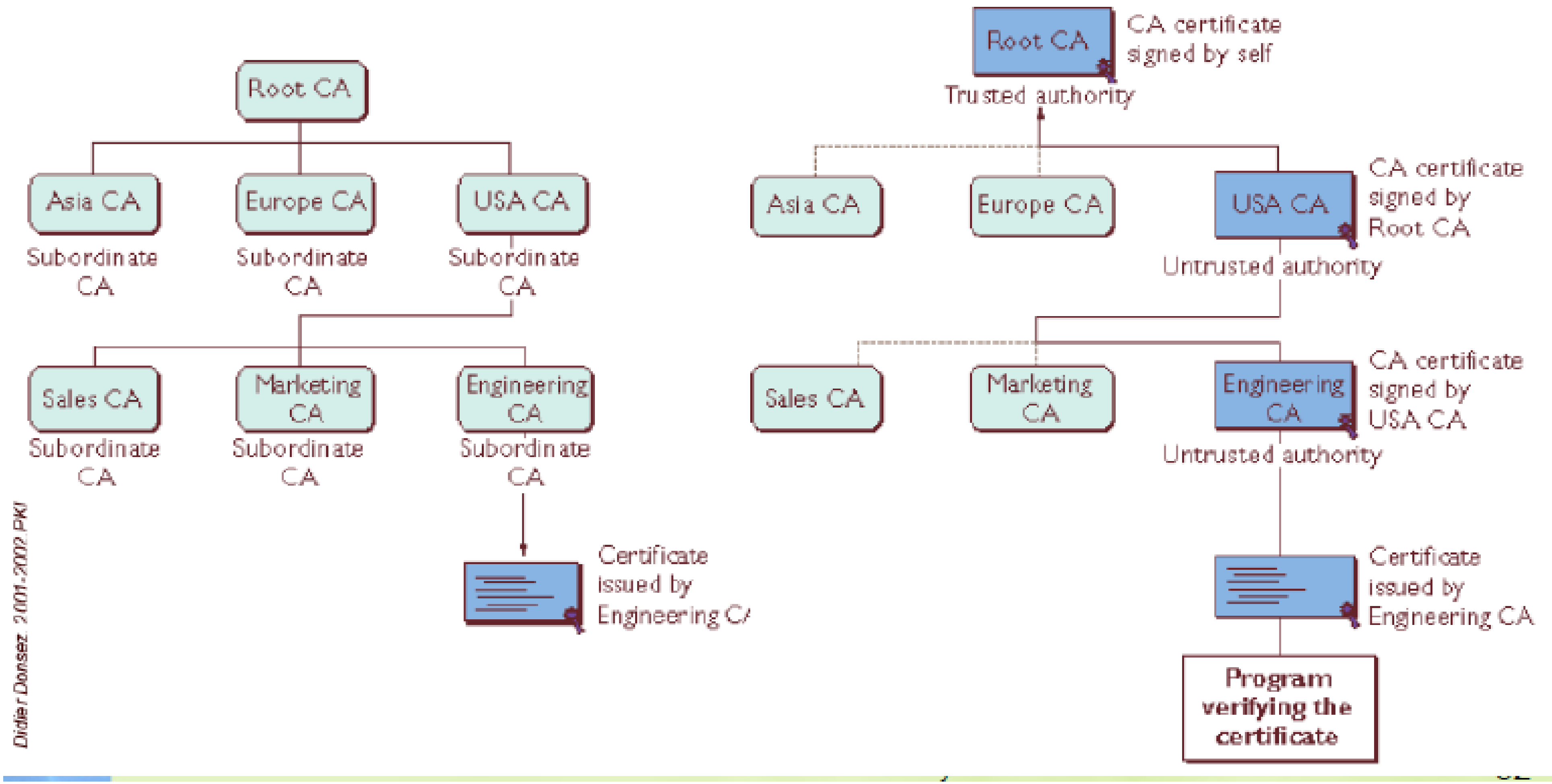
  … Store…

  … Distribute and revoke

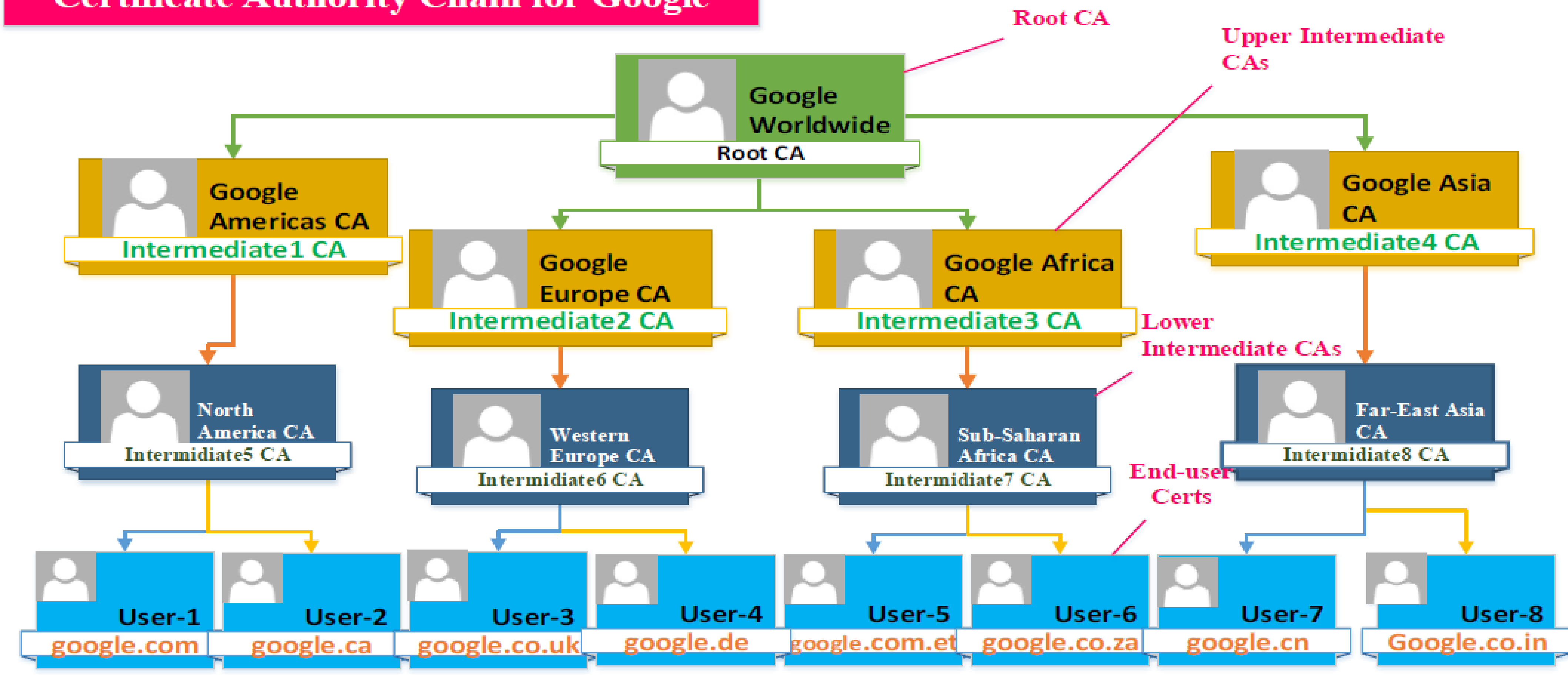
  certificates !

- For:
  - Authentication, Integrity, Confidentiality, Non-repudiation, (access control)

Delegation of authority

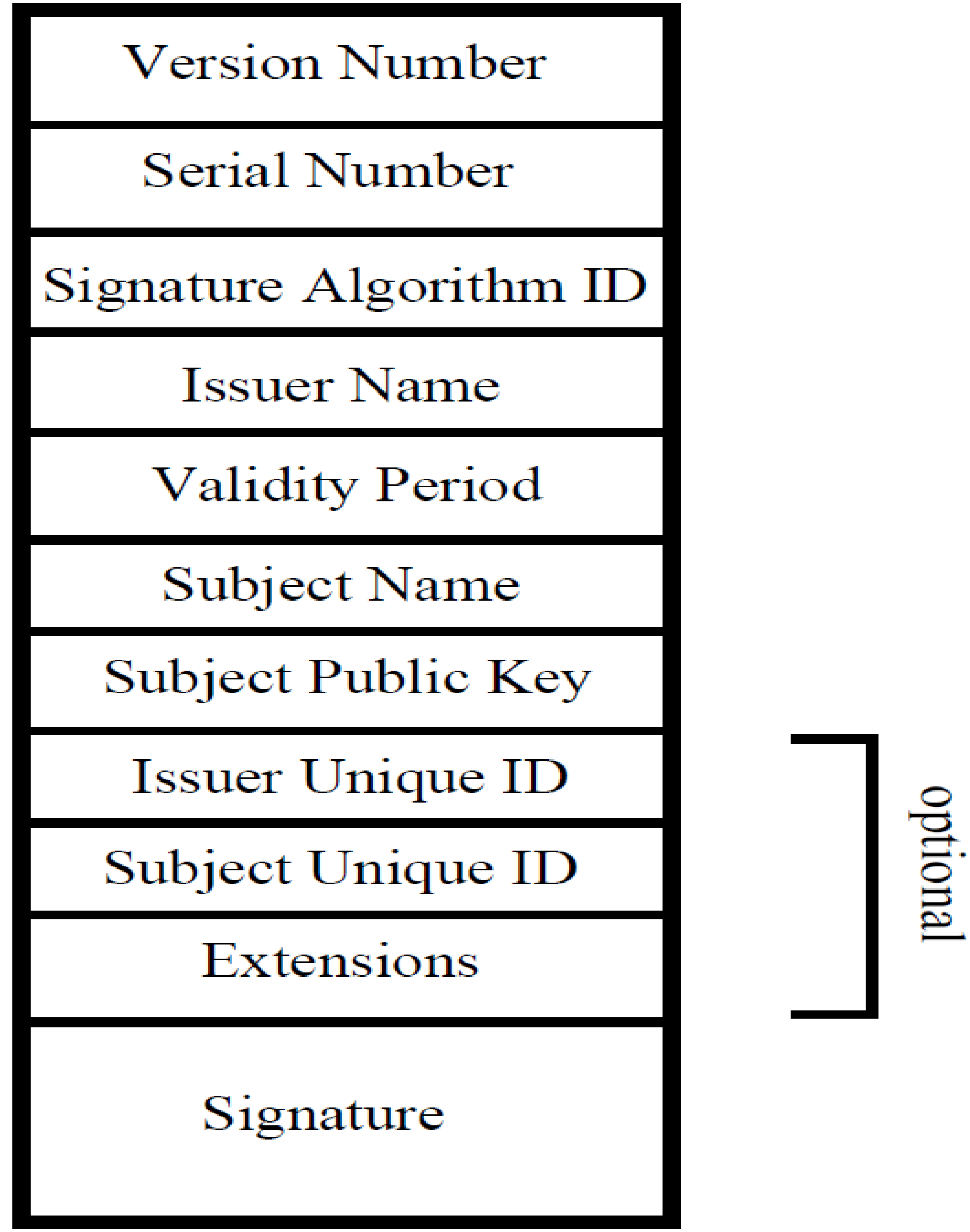


Certificate Authority Chain for Google



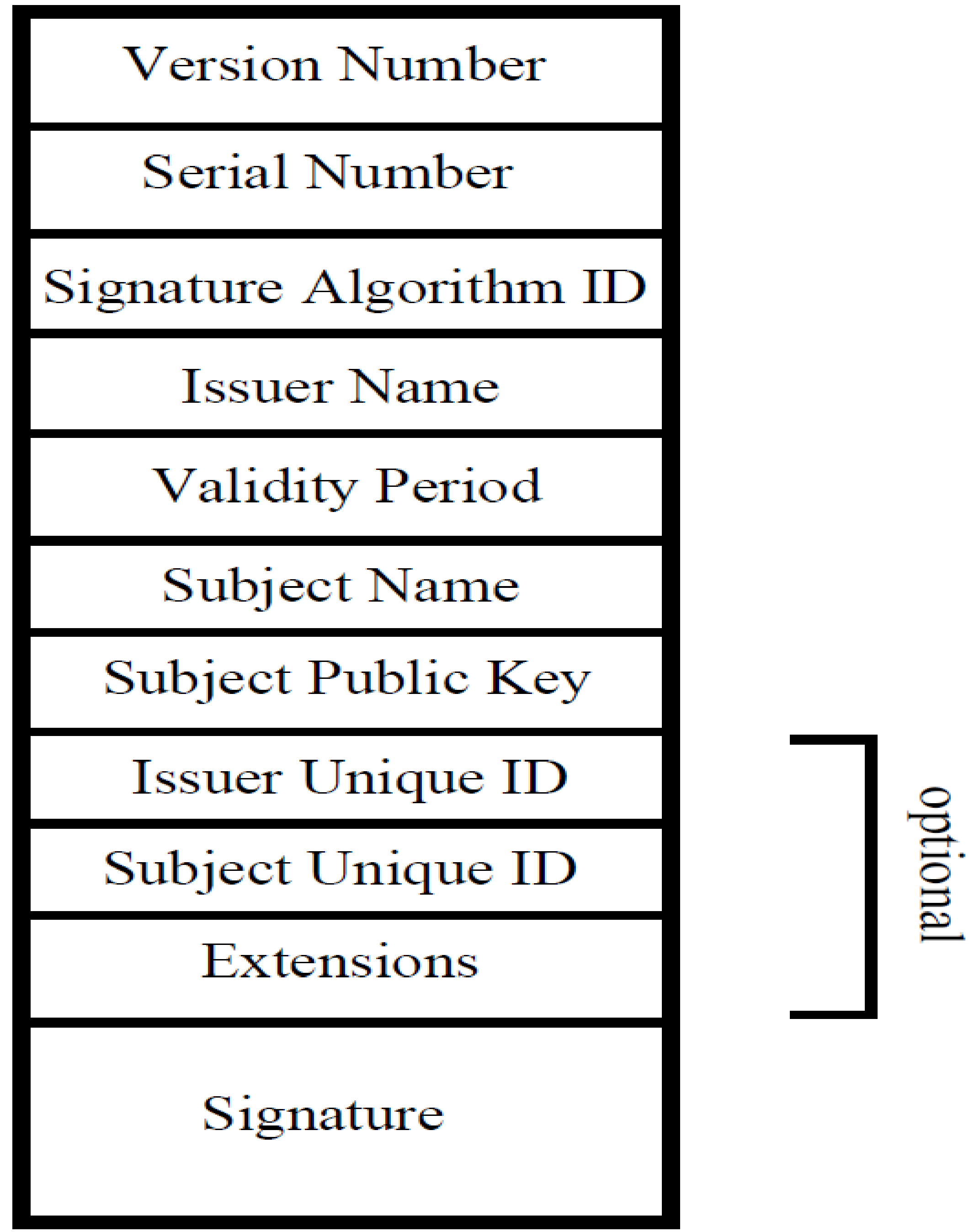
X.509 Certificate Format

- **Version Number**: describes the version of the X.509 standard to which the certificate corresponds.
- **Serial Number**: This is the serial number assigned to a certificate by the CA.
- **Signature Algorithm ID**: This is the name of the digital signature algorithm used to sign the certificate. (MD5,SHA)
- **Issuer Name**: This is the name of the Certificate Authority that issued this certificate.
- **Validity Period**: This field states the time period during which the certificate is valid.
- **Subject Name**: This field identifies the individual/organization to which the certificate was issued.



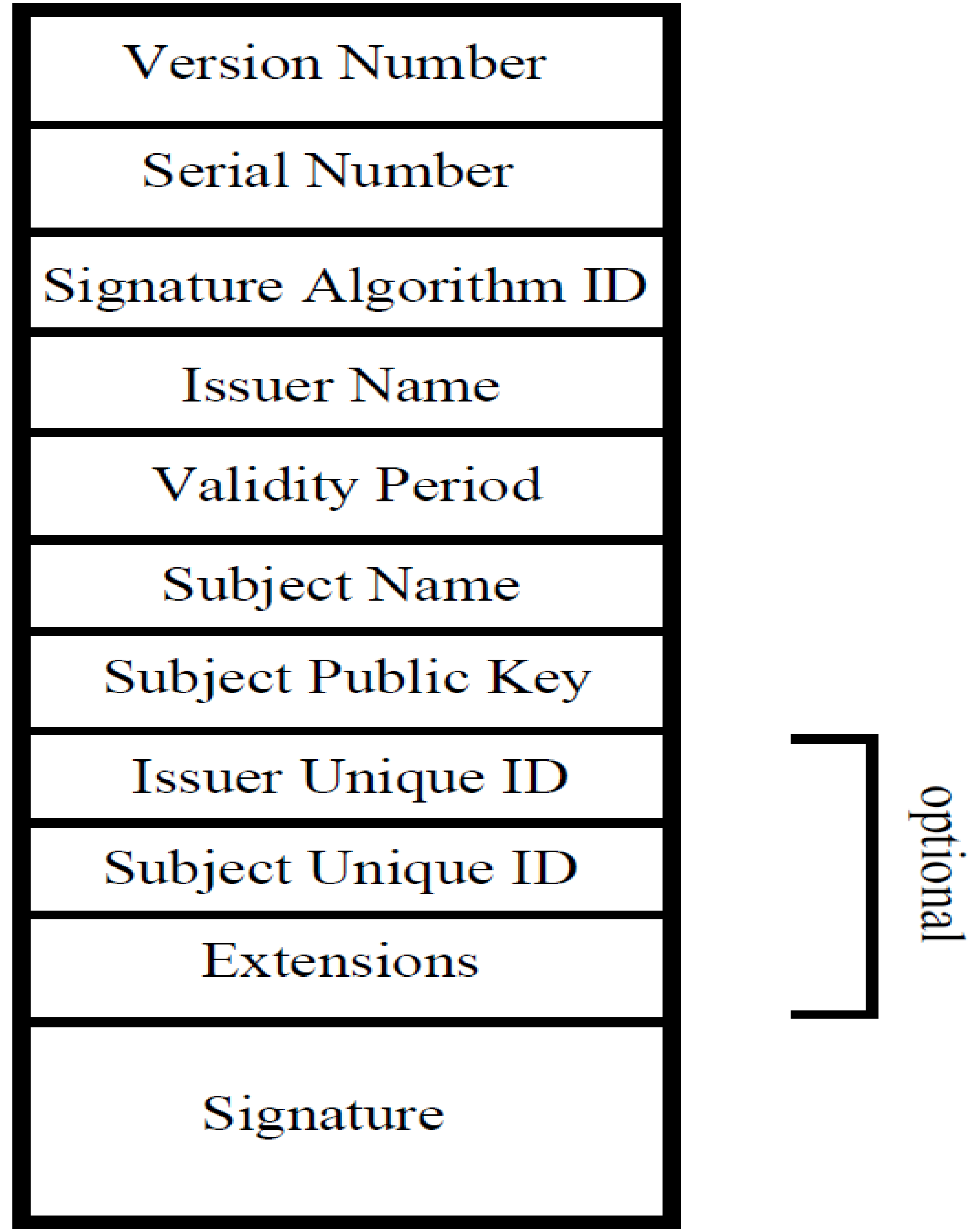
X.509 Certificate Format

- **Subject Public Key**: This field presents the public key that is meant to be authenticated by this certificate.
  - This field also names the algorithm used for public-key generation.
- **Issuer Unique Identifier:** (optional)With the help of this identifier, two or more different CA's can operate as logically a single CA.
  - The **Issuer Name** field will be distinct for each such CA but they will share the same value for the **Issuer Unique Identifier.**
- **Subject Unique Identifier**: (optional) With the help of this identifier, two or more different certificate holders can act as a single logical entity.



X.509 Certificate Format

- Each holder will have a different value for the **Subject Name** field but they will share the same value for the **Subject Unique Identifier** field.
- **Extensions**: (optional) This field allows a CA to add additional private information to a certificate.
- **Signature**: This field contains the signature of the CA that issued the certificate.
  - This signature is obtained by first computing a message digest from the rest of the fields with a hashing algorithm like SHA-1,
  - Then CA will encrypt MD using private key→ *Signature*



The X.509 Certificate format standard for Public Key Infrastructure

- The digital representation of an X.509 certificate, described in RFC 5280, is created by first using the following ASN.1 representation to generate a byte stream for the certificate and converting the byte stream into a printable form with Base64 encoding.

```
Certificate ::= SEQUENCE {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING  }

   TBSCertificate  ::=  SEQUENCE  {
        version         [0]  EXPLICIT Version DEFAULT v1,
        serialNumber         CertificateSerialNumber,
        signature            AlgorithmIdentifier,
        issuer               Name,
        validity             Validity,
        subject              Name,
        subjectPublicKeyInfo SubjectPublicKeyInfo,
        issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        extensions      [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version MUST be v3
        }

   Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

   CertificateSerialNumber  ::=  INTEGER
```

The X.509 Certificate format standard for Public Key Infrastructure

```
Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
```

The X.509 Certificate format standard for Public Key Infrastructure

- It is the hash of the bytestream that corresponds to what is stored for the field TBSCertificate that is encrypted by the CA’s private key for the digital signature that then becomes the value of the signatureValue field.

- You may read TBSCertificate as the “To Be Signed” portion of what appears in the final certificate.

- As to what algorithms are used for hashing and for encryption with the CA’s private key, that is identified by the value of the field signatureAlgorithm.

The X.509 Certificate format standard for Public Key Infrastructure

- Shown below is an example of a certificate in Base64 representation and it resides in a file whose name carries the ".pem" suffix.


```
-----BEGIN CERTIFICATE-----
MIIDJzCCApCgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBzjELMAkGA1UEBhMCWkEx
FTATBgNVBAgTDFdlc3Rlcm4gQ2FwZTESMBAGA1UEBxMJQ2FwZSBUb3duMR0wGwYD
VQQKExRUaGF3dGUgQ29uc3VsdGluZyBjYzEoMCYGA1UECxMfQ2VydGlmaWNhdGlv
biBTZXJ2aWNlcyBEaXZpc2lvbjEhMB8GA1UEAxMYVGhhd3RlIFByZW1pdW0gU2Vy
dmVyIENBMSgwJgYJKoZIhvcNAQkBFhlwcmVtaXVtLXNlcnZlckB0aGF3dGUuY29t
MB4XDTk2MDgwMTAwMDAwMFoXDTIwMTIzMTIzNTk1OVowgc4xCzAJBgNVBAYTAlpB
MRUwEwYDVQQIEwxXZXN0ZXJuIENhcGUxEjAQBgNVBAcTCUNhcGUgVG93bjEdMBsG
A1UEChMUVGhhd3RlIENvbnN1bHRpbmcgY2MxKDAmBgNVBAsTH0NlcnRpZmljYXRp
b24gU2VydmljZXMgRGl2aXNpb24xITAfBgNVBAMTGFRoYXd0ZSBQcmVtaXVtIFNl
cnZlciBDQTEoMCYGCSqGSIb3DQEJARYZcHJlbWl1bS1zZXJ2ZXJAdGhhd3RlLmNv
bTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA0jY2aovXwlue2oFBYo847kkE
VdbQ7xwblRZH7xhINTpS9CtqBo87L+pW46+GjZ4X9560ZXUCTe/LCaIhUdib0GfQ
ug2SBhRz1JPLlyoAnFxODLz6FVL88kRu2hFKbgifLy3j+ao6hnO2RlNYyIkFvYMR
uHM/qgeN9EJN50CdHDcCAwEAAaMTMBEwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG
9w0BAQQFAAOBgQAmSCwWwlj66BZ0DKqqX1Q/8tfJeGBeXm43YyJ3Nn6yF8Q0ufUI
hfzJATj/Tb7yFkJD57taRvvBxhEf8UqwKEbJw8RCfbz6q1lu1bdRiBHjpIUZa4JM
pAwSremkrj/xw0llmozFyD4lt5SZu5IycQfwhl7tUCemDaYj+bvLpgcUQg==
-----END CERTIFICATE-----
```

The X.509 Certificate format standard for Public Key Infrastructure

- Ordinarily you would request a CA for a certificate for your public key.
- But that does not prevent you from generating your own certificates for testing purposes.

```
openssl> req -new -newkey rsa:1024 -days 365 -nodes -x509 -keyout test.pem -out test.cert
```

- Where the first argument req to openssl is for generating an X509 certificate
- You can also use OpenSSL to make your own organization a CA.
- Visit http://sandbox.rulemaker.net/ngps/m2/howto.ca.html to find out how you can do it.

Digital Certificates

- Types of Digital Certificates
  - site certificates
    - used to authenticate web servers
  - personal certificates
    - used to authenticate individual users
  - software publishers certificates
    - used to authenticate executable
  - CA certificates
    - used to authenticate CA's public keys
  - All certificates have the common format standard of X.509v3