Professional Masters in Information and Cyber Security (PMICS)

Computer Science and Engineering,
University of Dhaka

CSE 802

Information Security Fundamentals

Lecture Content

- Hash Functions
- Digital Signature
- Key Distribution
  - Key distribution using symmetric key
  - Key distribution using asymmetric key
  - Public key distribution
  - Public key certificate
  - X.509 Certificate
  - PKIX
  - CA Hierarchy
  - Certificate validation

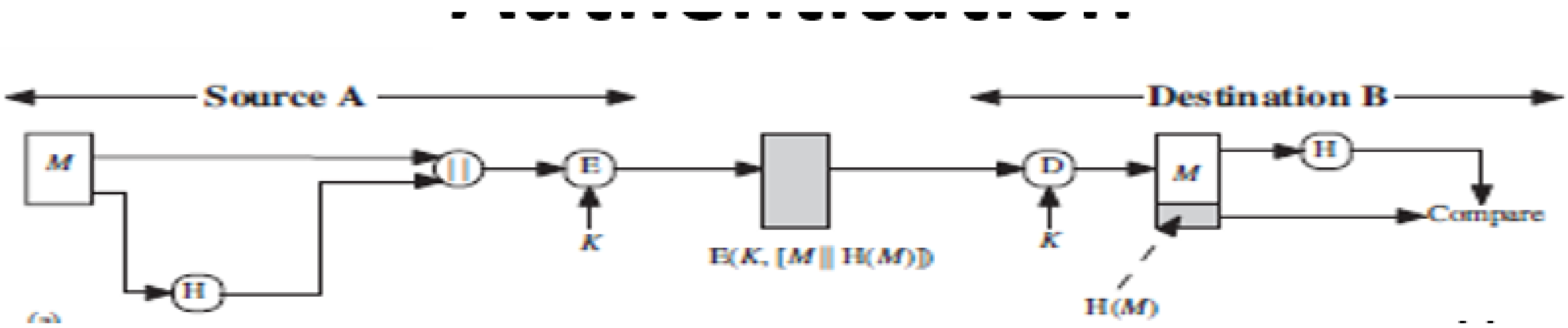
Hash Functions

- ✓ A hash function **H** takes a variable length block of data **M** as input and produces a fixed-size hash value **h = H(M)**.
  - ✓ Good hash function applying on large set of data produces output that are evenly distributed and apparently random.
  - ✓ A change to any bit or bits in **M** results with high probability, in a change to the result of hash function.

- ✓ **Cryptographic Hash Function:**
  - ✓ Deterministic: Same message always produces same result.
  - ✓ Quick to compute a hash value.
  - ✓ Computationally infeasible to find **M** from a **H(M)**. (One-way property).
  - ✓ Computationally infeasible to find $\mathbf{M_1}$ and $\mathbf{M_2}$ such that $\mathbf{H(M_1) = H(M_2)}$**.** Known as collision – resistant property.
  - ✓ A small change in **M** will significantly change **H(M)**.

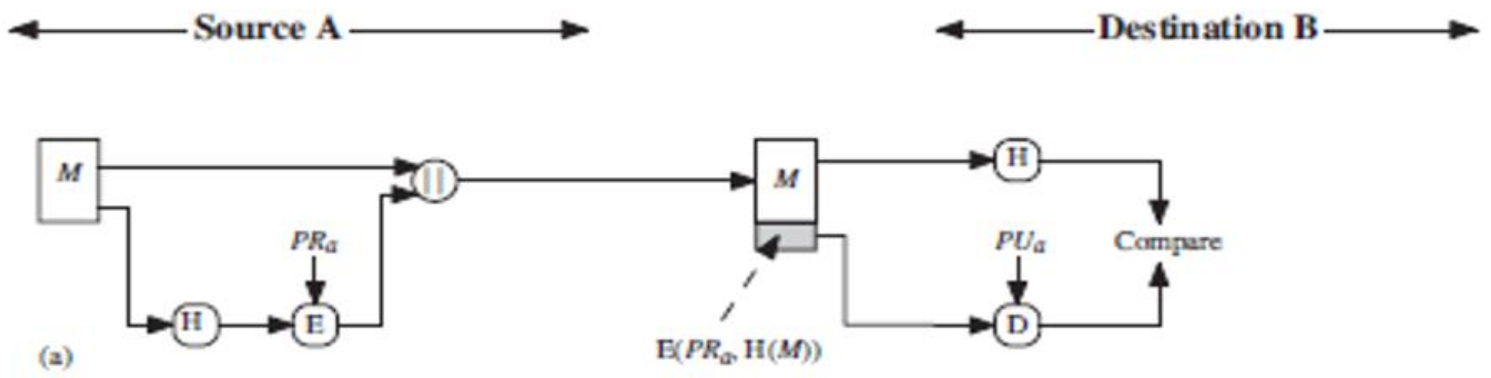
Applications of Cryptographic Hash Function-Message Authentication

- A mechanism used to verify the integrity of a message: message is not modified, no insertion and deletion or replay.
- When hash function is used to provide message authentication, the hash value is called is ***message digest***.



Application of Cryptographic Hash Function – Digital Signature

- Instead of digitally sign the whole document, hash of the document is signed to reduce cost.



Other Applications of Hash Function

- ✓ One-way password file: used in most operating system.
- ✓ Can be used for intrusion detection and virus detection.
- ✓ Used to construct **Pseudorandom function (PRF)** and **Pseudorandom number generator (PRNG).**

Secure Hashing Algorithm – SHA algorithm

- Generates cryptographically secure one-way hash function.
- Published by National Institute of Standards and Technology (NIST) and US Federal Information Processing Standard (FIPS).

Table 11.3 Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

Message Digest Function: MD5 and MD6

- MD5 algorithm takes an arbitrary length message and produces a fixed 128 bit digest.
- MD5 is not collision-resistant.
- MD6 uses Merkle tree like structure to provide the scope of huge parallelism.
-

Hash a File using Openssl

```
anu@anu-Vostro-5471:~$ vi data.txt
anu@anu-Vostro-5471:~$ cat data.txt
hello world.
anu@anu-Vostro-5471:~$ openssl list --digest-commands
blake2b512        blake2s256        md5               rmd160
sha1              sha224            sha256            sha3-224
sha3-256          sha3-384          sha3-512          sha384
sha512            sha512-224        sha512-256        shake128
shake256          sm3

anu@anu-Vostro-5471:~$
```

```
anu@anu-Vostro-5471:~$ openssl dgst -sha256 data.txt
SHA2-256(data.txt)= d0083d8df0fbb4a4bf8ba6c85a8155abd392314fe99b2b61cefe476a46a9af32
anu@anu-Vostro-5471:~$
```

Digital Signature Overview

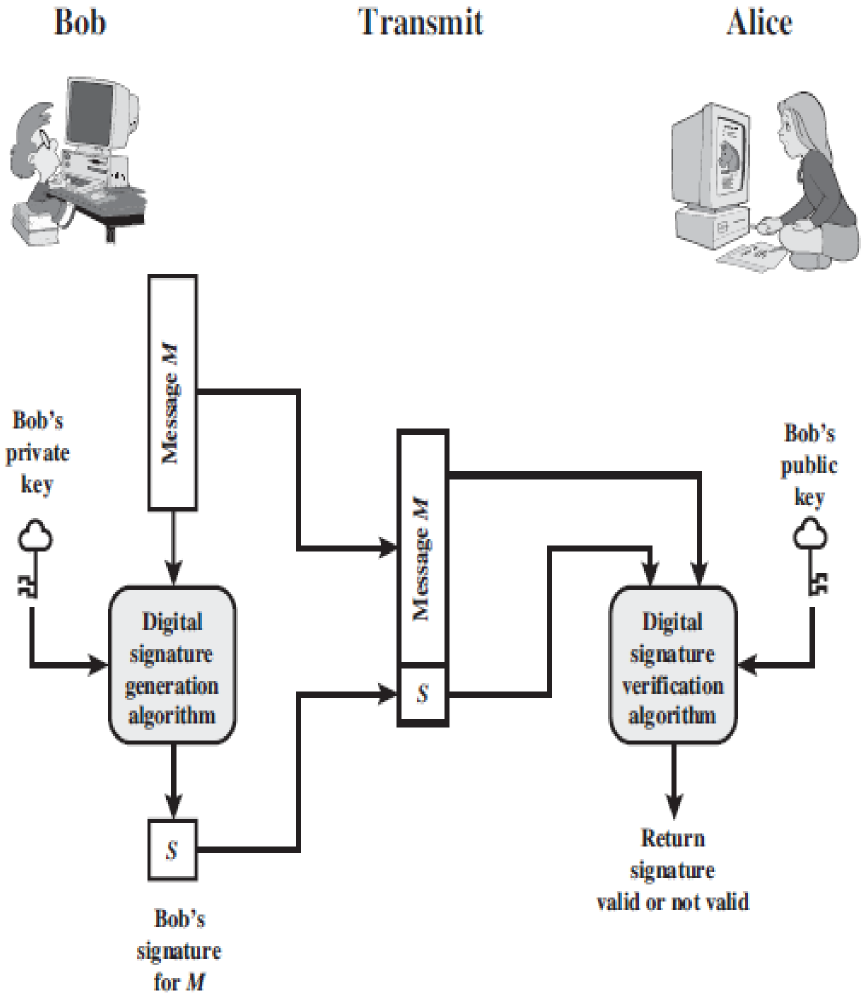


Figure 13.1 Generic Model of Digital Signature Process

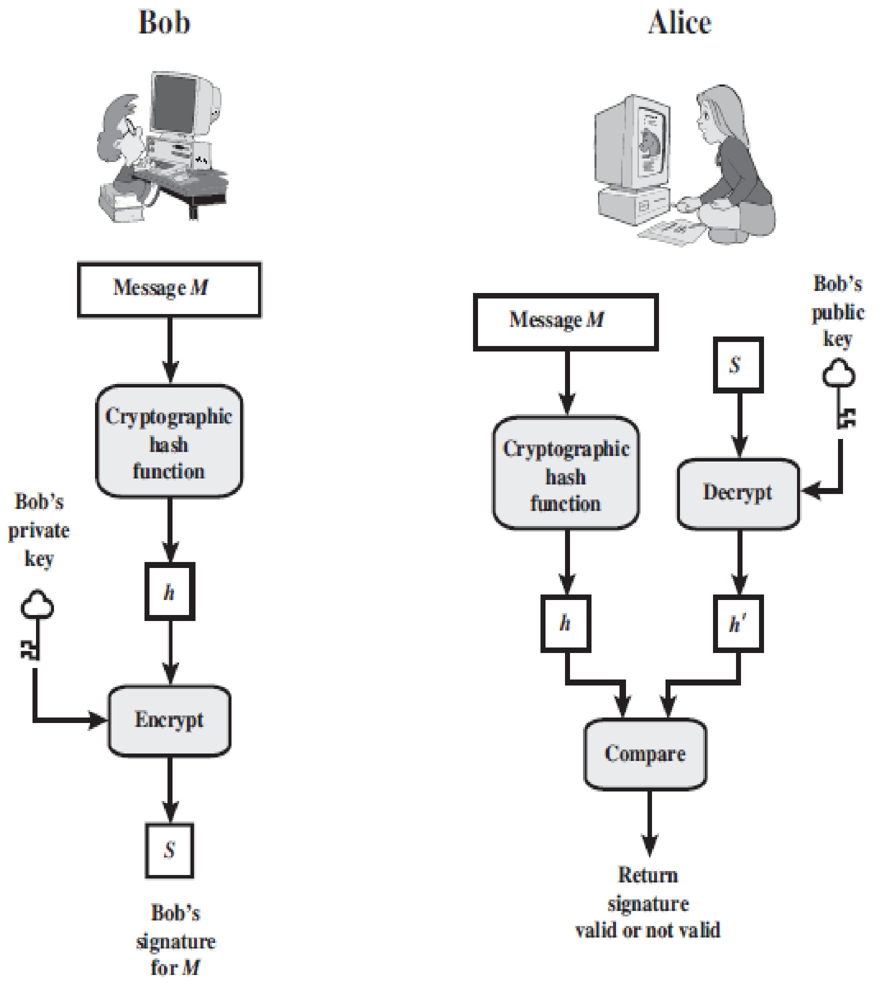


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

Scenarios for Digital Signature

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

- **<u>Properties of Digital signature:</u>**

1. It must verify the author and the date and time of the signature.
2. It must authenticate the contents at the time of the signature.
3. It must be verifiable by a third party to resolve the disputes.

Digital Signature Requirements

- Must depend on the message being signed
- Must use information unique to sender to prevent both forgery and denial
- Must be relatively easy to produce
- Must be relatively easy to recognize and verify
- Be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message

Digital Signature Scheme

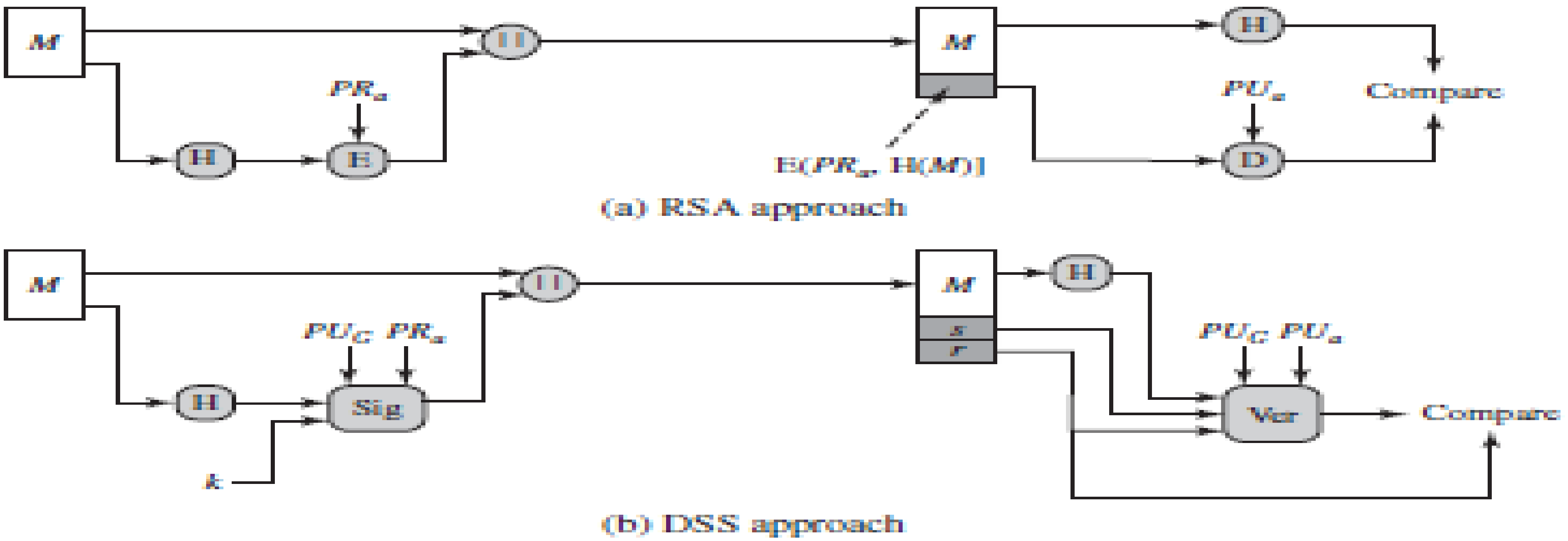


Figure 13.3 Two Approaches to Digital Signatures

- ✓ RSA can be used for encryption, digital signature and key exchange.
- ✓ DSA is only used for digital signature generation. It is a public-key technique.

Digital Signature Standard (DSS)

- US Govt approved signature scheme
- Designed by NIST in early 90's
- Published as FIPS-186 in 1991
- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm

Digital Signature Algorithm (DSA):

- creates a 320 bit signature
- smaller and faster than RSA
- security depends on difficulty of computing discrete logarithms

Signing a Document using Openssl

```
anu@anu-Vostro-5471:~$ openssl genrsa -out key3.pem 4096
anu@anu-Vostro-5471:~$ ls
data.txt    file.dec  key3.pem  Public       Templates   test.pem
Desktop     file.enc  key.pem   public1.pem  test1.cert  Videos
Documents   file.txt  Music     public.pem   test1.pem
Downloads   key1.pem  Pictures  snap         test.cert
anu@anu-Vostro-5471:~$ openssl rsa -in key3.pem -pubout > key3.pub
writing RSA key
anu@anu-Vostro-5471:~$ ls
data.txt    file.dec  key3.pem  Pictures     snap        test.cert
Desktop     file.enc  key3.pub  Public       Templates   test.pem
Documents   file.txt  key.pem   public1.pem  test1.cert  Videos
Downloads   key1.pem  Music     public.pem   test1.pem
anu@anu-Vostro-5471:~$
```

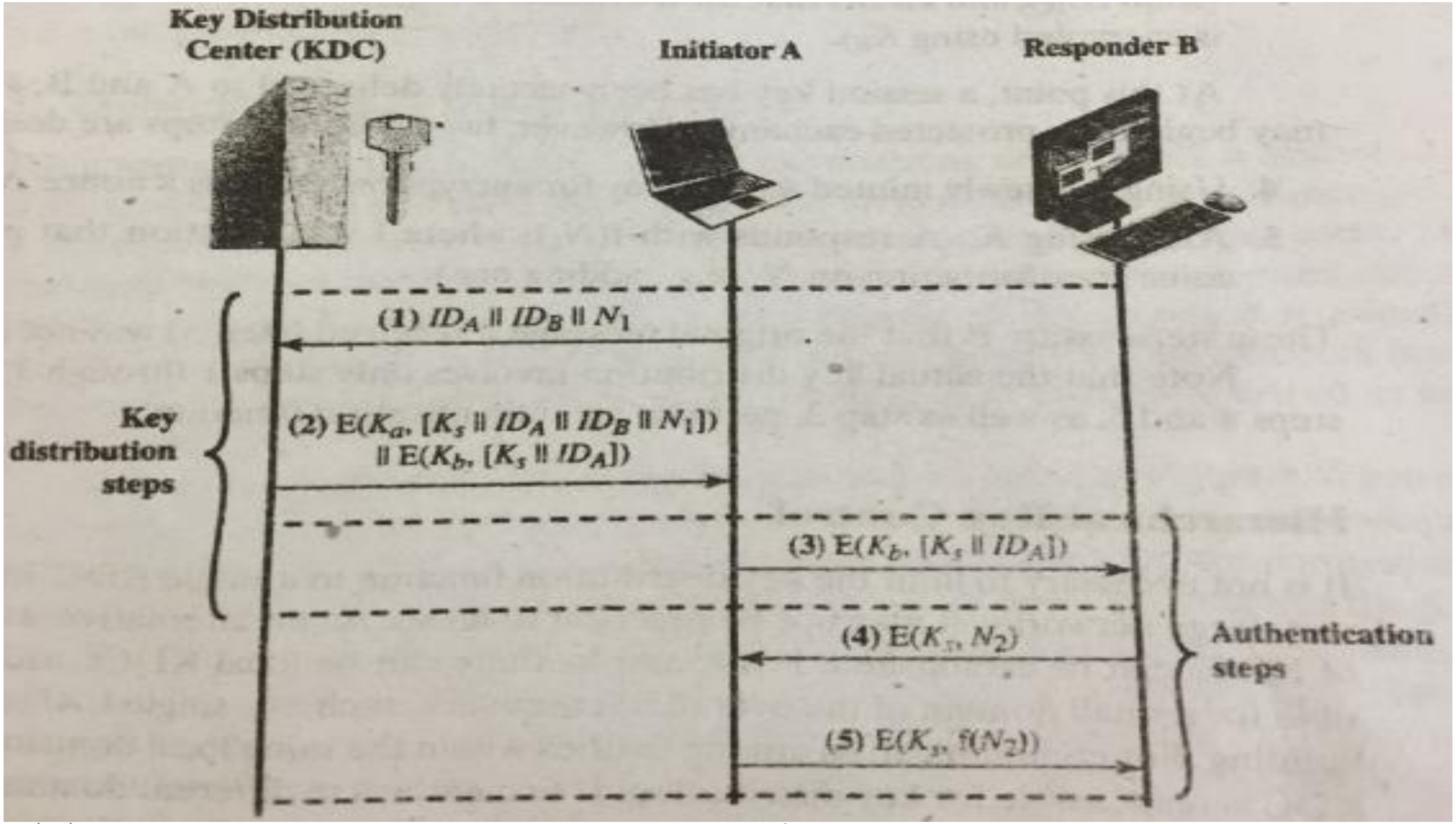
```
anu@anu-Vostro-5471:~$ openssl pkeyutl -verify -rawin -pubin -inkey key3.pub -in data.txt -sigfile data.sig
Signature Verified Successfully
anu@anu-Vostro-5471:~$
```

```
anu@anu-Vostro-5471:~$ openssl pkeyutl -sign -rawin -inkey key3.pem -in data.txt -out data.sig
anu@anu-Vostro-5471:~$ ls
data.sig      Documents  file.txt  key.pem   public1.pem  test1.cert  Videos
data.txt      Downloads  key1.pem  Music     public.pem   test1.pem
data.zip.sign file.dec   key3.pem  Pictures  snap         test.cert
Desktop       file.enc   key3.pub  Public    Templates    test.pem
anu@anu-Vostro-5471:~$ cat data.sig
D◆A◆◆j;{◆◆h◆◆◆◆(\◆*e◆C ◆◆◆◆au#]◆
                    B◆◆bk◆◆U◆◆    ◆◆JQ"'S7&>◆◆◆◆6◆[Xby◆p,7foq◆aN-◆◆ZG◆◆3I◆d◆m◆8◆◆R◆*◆
            =<◆#◆◆◆◆◆◆  N◆◆C◆◆KJ◆◆`◆◆◆◆r◆& ◆◆`◆-U◆◆7◆k◆&◆hs◆?,◆◆◆T◆[◆◆◆◆)Cv◆š5◆◆◆|/◆y@'1h
          ◆?◆◆◆vsF◆◆◆◆◆>◆q◆&◆E"h◆h◆}#yV◆◆G.◆◆◆◆◆◆◆0◆◆j7rU◆    ◆f3\◆<r◆4;◆>◆7◆◆◆◆◆◆a◆=Vn◆◆◆J◆◆◆◆◆&◆◆{◆Cp◆◆E◆◆.%◆◆x\◆◆K ◆◆9◆◆◆◆E◆f◆◆=◆! ◆◆◆◆i/◆◆
◆◆x1;◆◆◆<◆◆ù◆?◆"◆3◆s.◆!Y◆◆d◆6◆◆◆◆◆◆ ◆s^◆◆◆◆agVvz◆7◆◆◆U
anu@anu-Vostro-5471:~$
```

Key Distribution (Symmetric Key Distribution using Symmetric Key)

- ✓ symmetric schemes require both parties to share a common secret key
- ✓ issue is how to securely distribute this key whilst protecting it from others
- ✓ frequent key changes can be desirable
- ✓ often secure system failure due to a break in the key distribution scheme
- ✓ given parties A and B have various **key distribution** alternatives:
  - ✓ A can select key and physically deliver to B
  - ✓ third party can select & physically deliver key to A & B
  - ✓ if A & B have communicated previously can use previous key to encrypt a new key
  - ✓ if A & B have secure communications with a third party C, C can relay key between A & B

A Key Distribution Scenario



Key Distribution Issues

- **<u>Hierarchical Key Control:</u>**
  - ✓ For large network, a hierarchy of KDC is established.
  - ✓ For communication among the same local domain, the local KDC is responsible for key distribution.
  - ✓ For two entities in different domains, the corresponding KDCs can communicate through a global KDC.
  - ✓ Number of layers in hierarchical key control depends on the size of the population and the geographic scope of the internetwork.
  - ✓ Advantages:
    - ✓ Reduces the overhead of master key sharing.
    - ✓ Limits the damage of a faulty or subverted KDC to its local area only.

Decentralized Key Distribution

- Eliminates the need that KDC should be trusted and not be compromised.
- May be suitable for local context.
- Need ***n(n-1)/2*** master keys to support ***n*** systems.

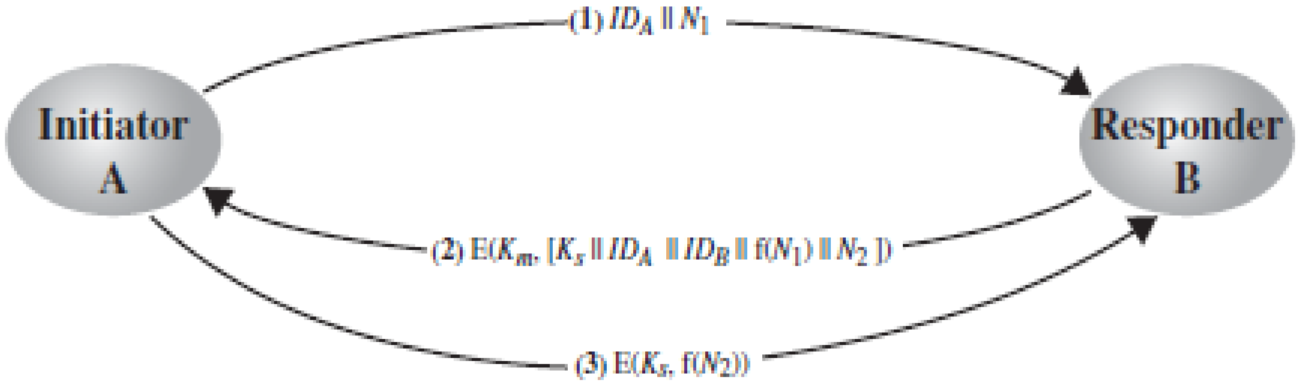
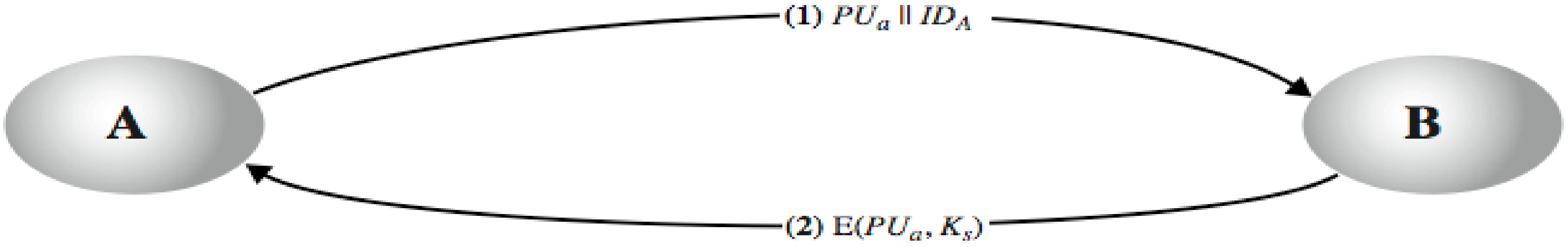


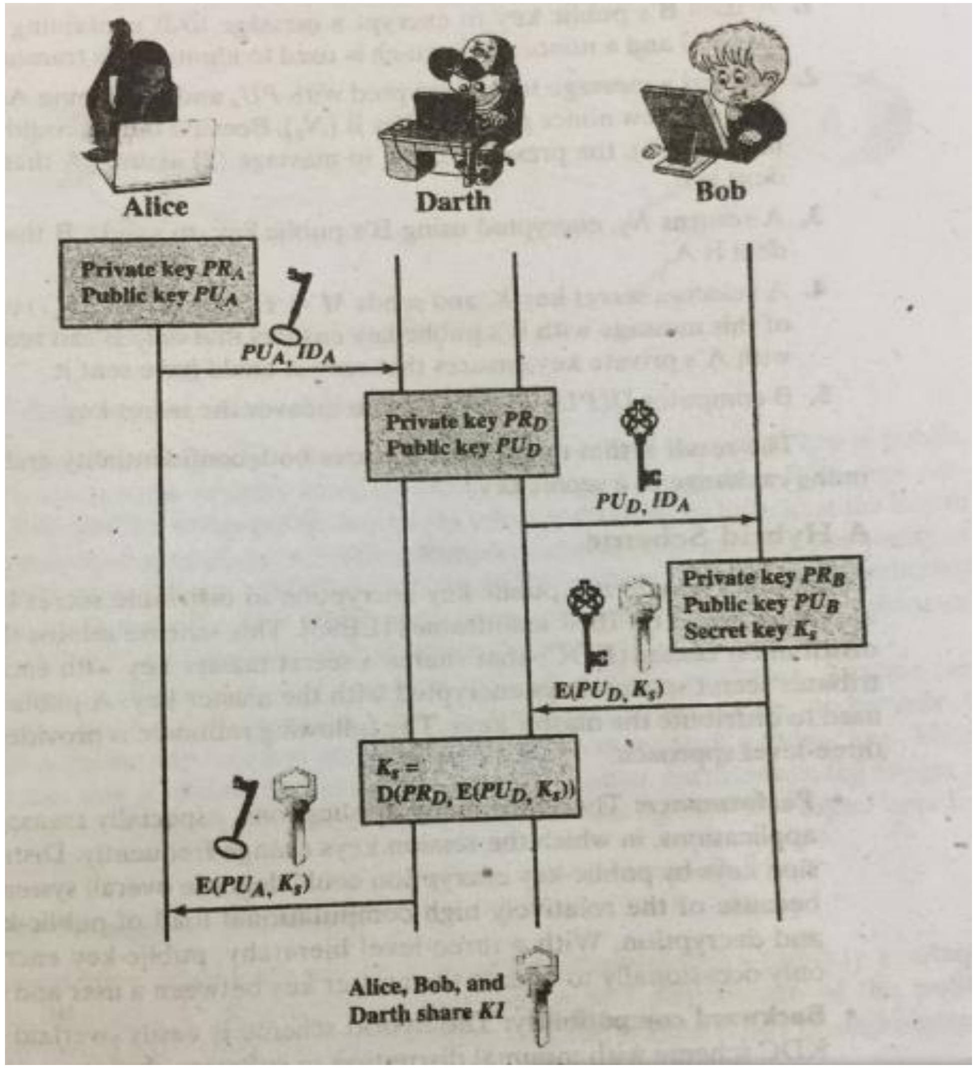
Figure 14.5 Decentralized Key Distribution

Symmetric Key Distribution using Asymmetric Encryption

- Merkle proposed this very simple scheme
  - allows secure communications
  - no keys before/after exist



Man-in-the Middle Attack



Secret Key Distribution with Confidentiality and Authentication

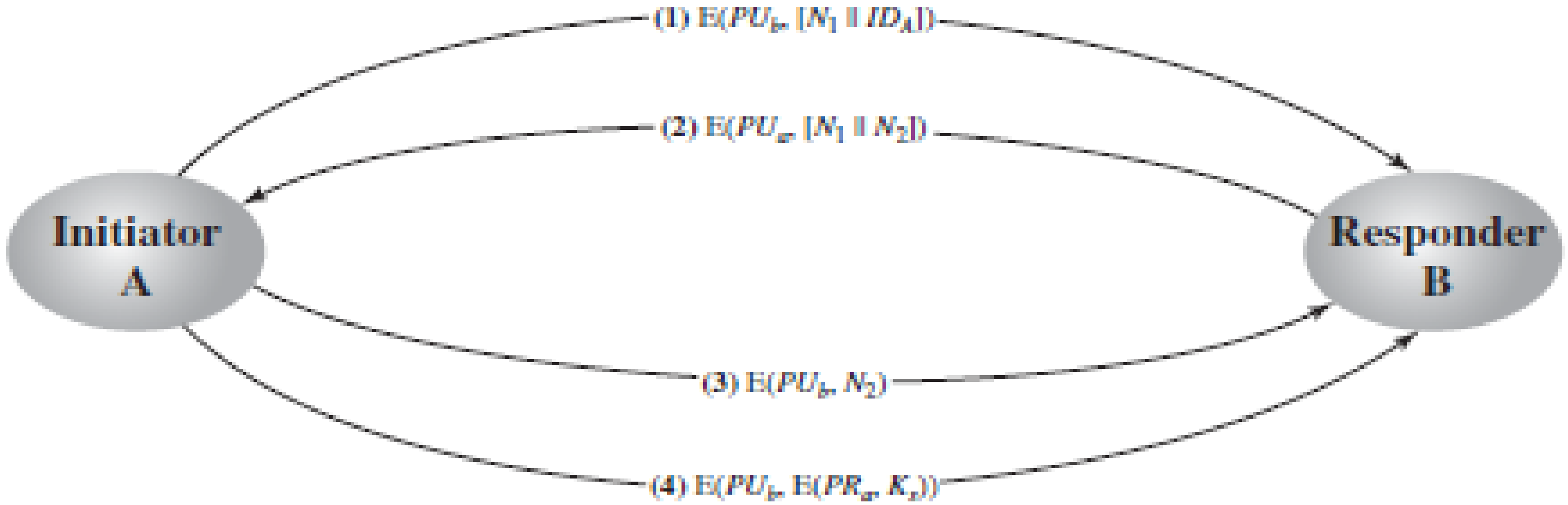


Figure 14.8 Public-Key Distribution of Secret Keys

Distribution of Public Key

- can be considered one of the following:
  - public announcement
  - publicly available directory
  - public-key authority
  - public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
  - eg. append keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user

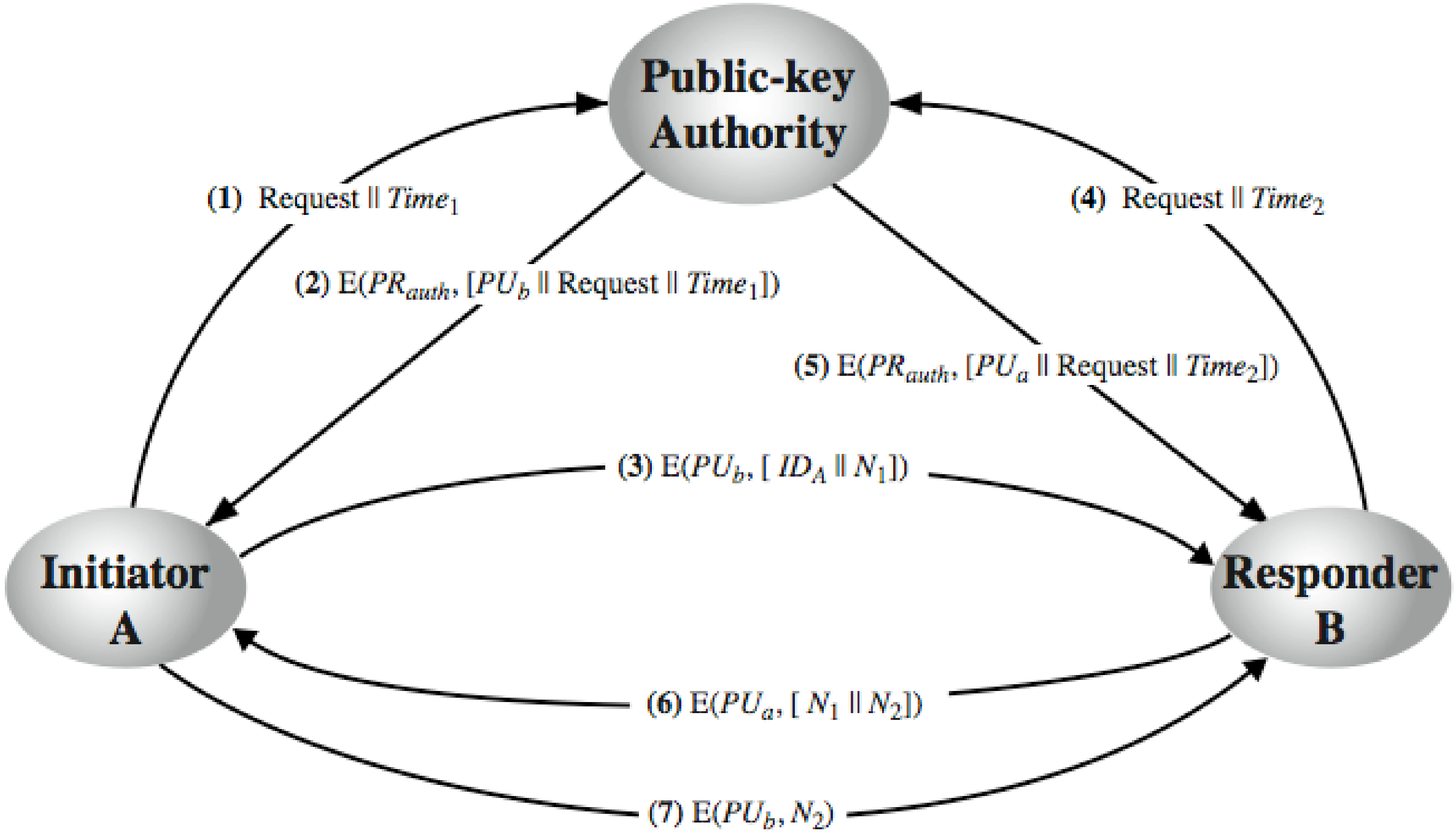
Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name,public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery
  - Can approve counterfeit public keys
  - Can modify existing records and pretend as another entity.

Public Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed
  - may be vulnerable to tampering

Public Key Authority



Public Key Certificate

- **Problems of Public Key Directory:**
  - Every users have to contact the public key directory to obtain the public key of the users they want to contact.
  - Directory maintained by the public key authority is vulnerable to tampering.
- **Public Key Certificate:**
  - Overcomes the problem of public key directory.
  - A certificate contains a **public key** and an **identifier of the key owner**. The certificate is signed by a trusted third party.
  - The trusted third party can be a government organization or a financial institute trusted by the user community.

Public Key Certificate

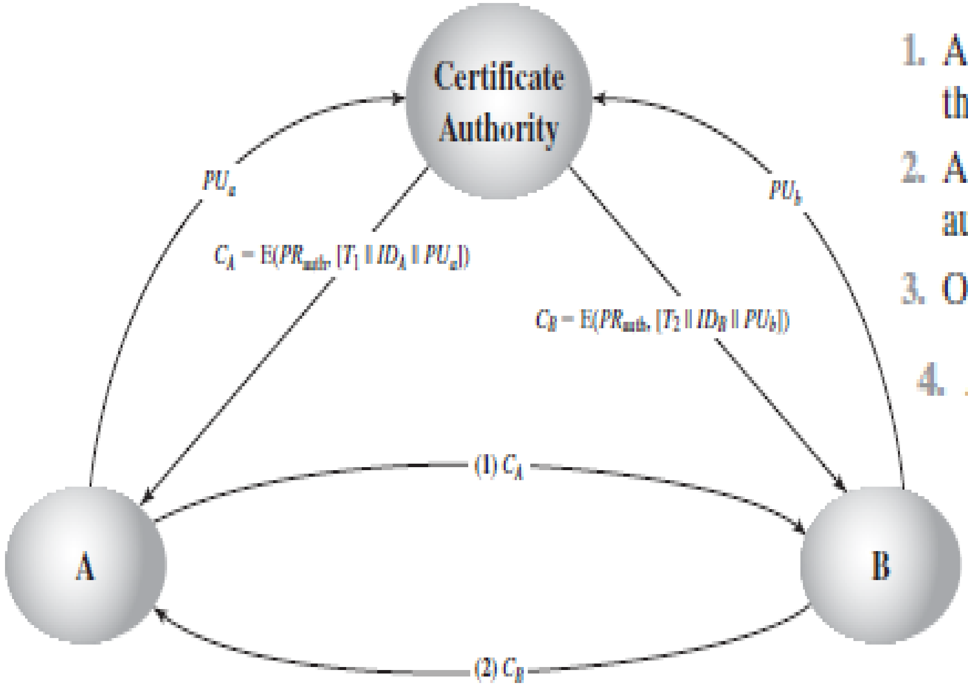


Figure 14.12 Exchange of Public-Key Certificates

Requirements for Public Key Certificate:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

X.509 Certificates

- part of ITU-T X.500 directory service standards
  - distributed servers maintaining user info database
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended
- X.509 certificates are widely used in S/MIME, IP security and SSL/TLS.
  - have 7 versions
- Specified in RFC 5280.

X.509 Certificates

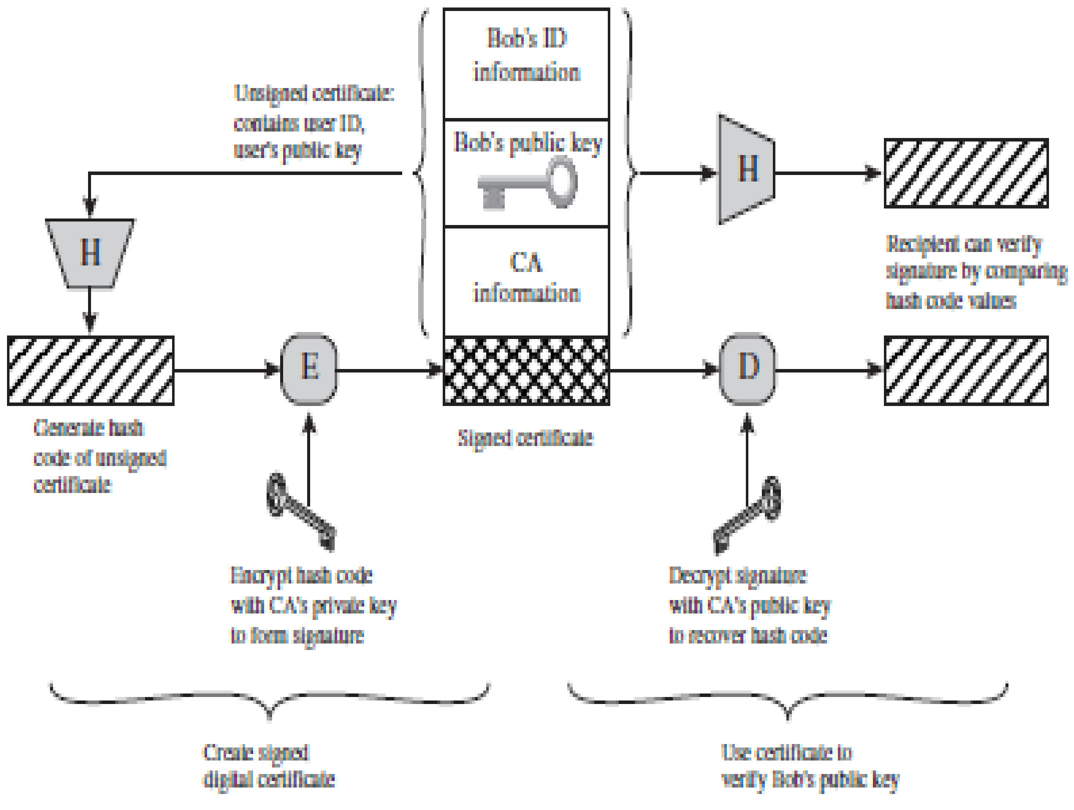
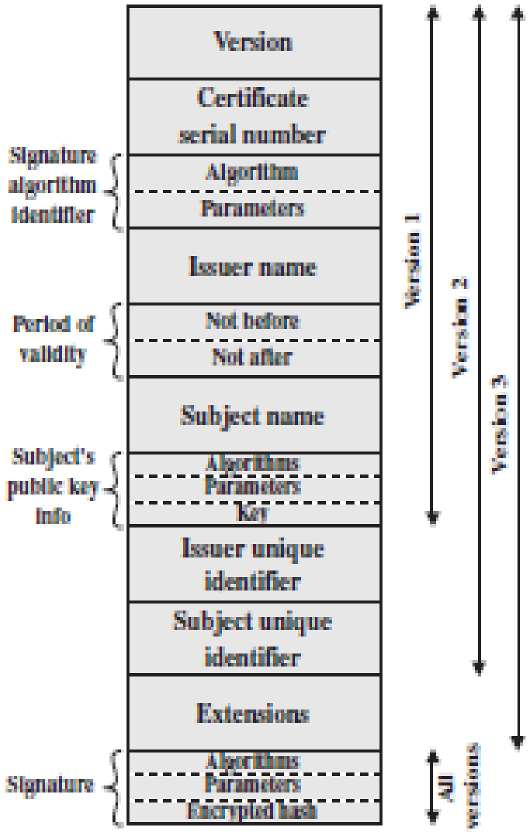


Figure 14.13 Public-Key Certificate Use



(a) X.509 certificate

Obtaining a User’s Certificate

- Any user with access to the public key of CA can verify that the user public key is certified issued by the same CA.
- Only the CA can modify a certificate
- Because cannot be forged, certificates can be placed in a public directory
- **Case Study:** A obtains a certificate from CA $\mathbf{X_1}$ and B has obtained a certificate from CA $\mathbf{X_2}$.
  - If $\mathbf{X_1}$ and $\mathbf{X_2}$ exchanged their public key secretly then

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 X_1’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

$$X_1 \ll X_2 \gg X_2 \ll B \gg$$

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

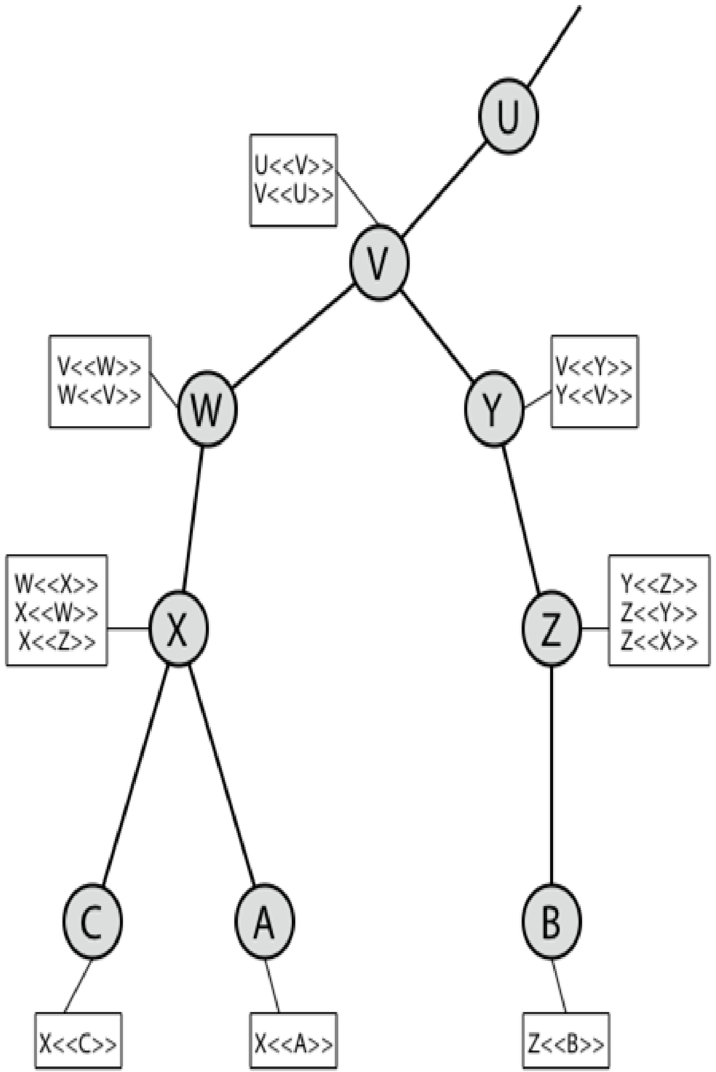
$$X_2 \ll X_1 \gg X_1 \ll A \gg$$

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

$$X_1 \ll X_2 \gg X_2 \ll X_3 \gg \ldots X_N \ll B \gg$$

In this case, each pair of CAs in the chain (X_i, X_{i+1}) must have created certificates for each other.

X.509 Hierarchy



A to B:

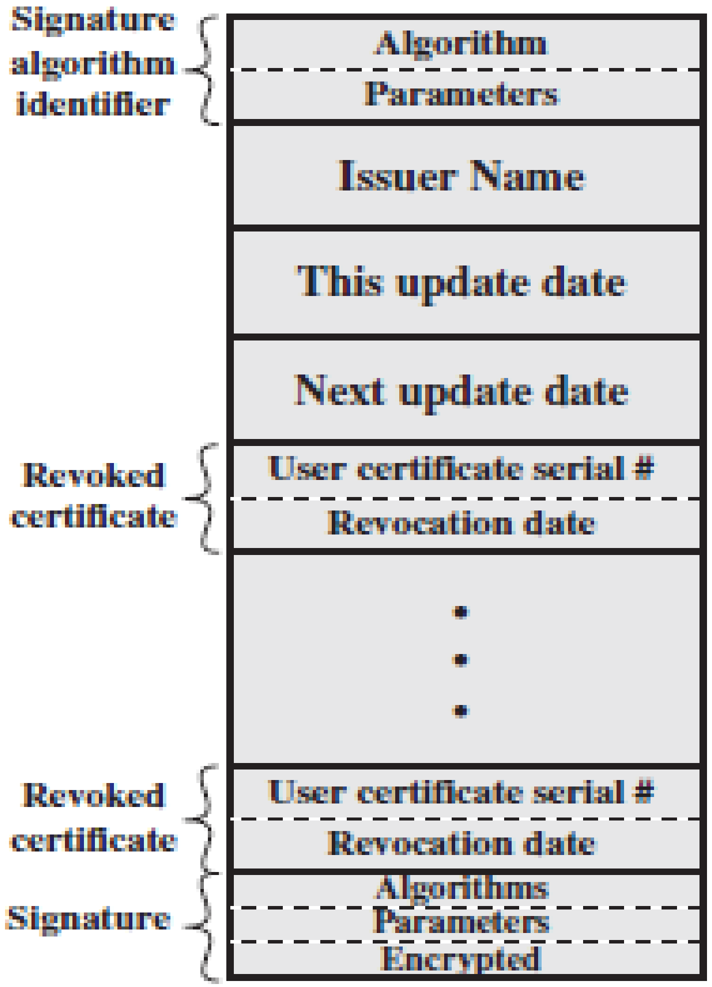
X ≪W≫ W ≪V≫ V ≪Y≫ Y ≪Z≫ Z ≪B≫

B to A:

Z ≪Y≫ Y ≪V≫ V ≪W≫ W ≪X≫ X ≪A≫

Certificate Revocation

- certificates have a period of validity
- may need to revoke before expiry, eg:
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA’s maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- users should check certificates with CA’s CRL



(b) Certificate revocation list

X.509 Version 3

- ✓ has been recognised that additional information is needed in a certificate
  - ✓ email/URL, policy details, usage constraints
- ✓ rather than explicitly naming new fields defined a general extension method
- ✓ extensions consist of:
  - ✓ extension identifier
  - ✓ criticality indicator
  - ✓ extension value

X.509 Version 3

- **<u>Certificate Extensions:</u>**

✓ key and policy information
  - ✓ convey info about subject & issuer keys, plus indicators of certificate policy

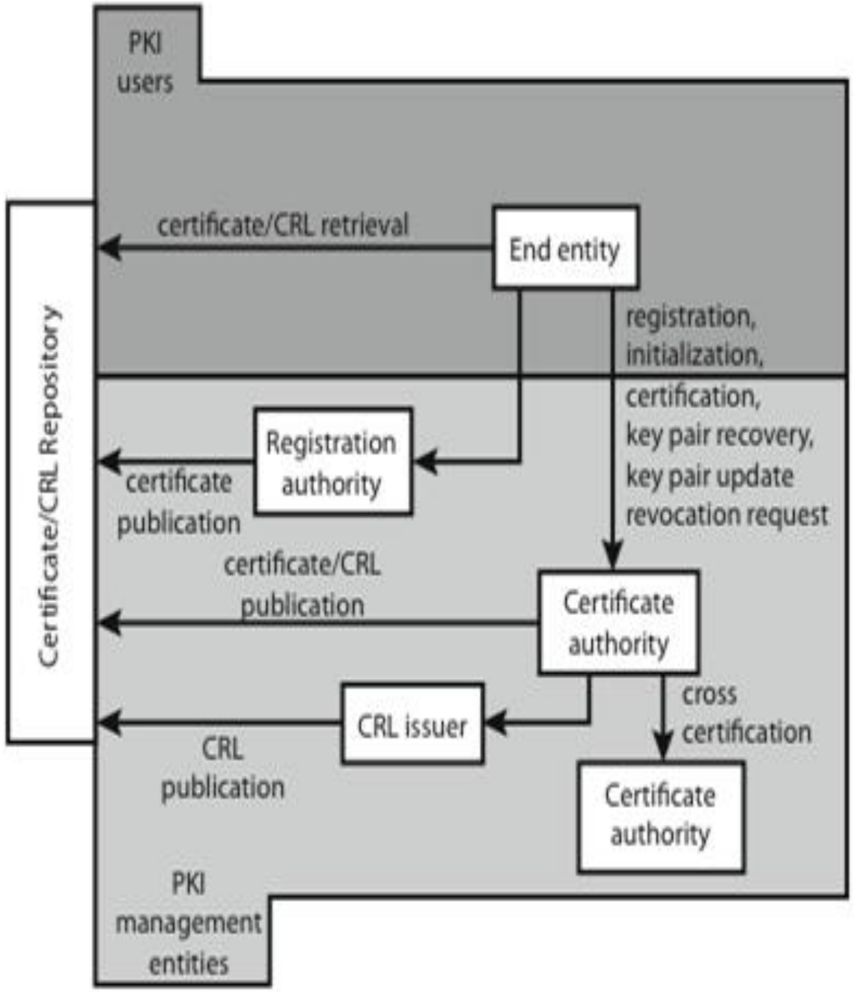
✓ certificate subject and issuer attributes
  - ✓ support alternative names, in alternative formats for certificate subject and/or issuer

✓ certificate path constraints
  - ✓ allow constraints on use of certificates by other CA’s

Public-Key Infrastructure (PKI)

The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group established a formal model based on X.509 for deploying a certificate-based architecture on the internet.

PKIX Architecture Model



PKIX Management

- Functions:
  - registration
  - initialization
  - certification
  - key pair recovery
  - key pair update
  - revocation request
  - cross certification
- Protocols: CMP, CMC

Different Types of Certificates

- Extended Validation (EV) certificates
  - A rigorous identity verification process is required to establish the legitimacy of an organization.
  - It involves verifying that the applicant organization has a legal and physical existence and the information provided by the applicant matches with information gleaned from other government and other records.
  - whether the applicant has exclusive rights to the domain specified in the application.
  - For a website that offers such a certificate to your browser, some part of the URL window will turn green.
  - It may take several days for a CA to issue such a certificate and most expensive certificates.

- Organization Validation (OV) certificates
  - Identity checks are less intense.
  - Usually, the existence of the organization is verified, the name of the domain is verified, which may be followed by a phone call from the CA.

Different Types of Certificates

- Domain Validation (DV) certificates.
  - The only check that is made is that the applicant has the right to use a specific domain name by checking information provided during certificate application.
  - by comparing the domain name against the database of the currently existing domain names, and by checking various internet directories.
  - Such certificates are the least expensive and are normally issued in just a few minutes.

CA Hierarchy

- The CAs operate through a strict hierarchical organization 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.
- Obviously, each root CA sits at the top of a tree-like structure of intermediate-level CAs.
- Your computer comes pre-loaded with the public keys for the root CAs.
- In a Linux machine, these certificates typically reside in the directory "/etc/ssl/certs/".
- You can view any of these certificates by executing the cat command.
- Some well known CA: Comodo, DigiCert, goDaddy, IdenTrust.

```
anu@anu-Vostro-5471:/etc/ssl/certs$ cat vTrus_Root_CA.pem
```


```
-----BEGIN CERTIFICATE-----
MIIFVjCCAz6gAwIBAgIUQ+NxE9izWRRdt86M/TX9b7wFjUUwDQYJKoZIhvcNAQEL
BQAwQzELMAkGA1UEBhMCQ04xHDAaBgNVBAoTE2lUcnVzQ2hpbmEgQ28uLEx0ZC4x
FjAUBgNVBAMTDXZUcnVzIFJvb3QgQ0EwHhcNMTgwNzMxMDcyNDA1WhcNNDMwNzMx
MDcyNDA1WjBDMQswCQYDVQQGEwJDTjEcMBoGA1UEChMTaVRydXNDaGluYSBDby4s
THRkLjEWMBQGA1UEAxMNdlRydXMgUm9vdCBDQTCCAiIwDQYJKoZIhvcNAQEBBQAD
ggIPADCCAgoCggIBAL1VfGHTuB0EYgWgrmy3cLRB6ksDXhA/kFocizuwZotsSKYc
IrrVQJLuM7IjWcmOvFjai57QGfIvWcaMY1q6n6MLsLOaXLoRuBLpDLvPbmyAhykU
AyyNJJrIZIO1aqwTLDPxn9wsYTwaP3BVm60AUn/PBLn+NvqcwBauYv6WTEN+VRS+
GrPSbcKvdmaVayqwlHeFXgQPYh1jdfdr58tbmnDsPmcF8P4HCIDPKNsFxhQnL4Z9
8Cfe/+Z+M0jnCx5Y0ScrUw5XSmXX+6KAYPxMvDVTAWqXcoKv8R1w6Jz1717CbMdH
flqUhSZNO7rrTOiwCcJlwp2dCZtOtZcFrPUGoPc2BX70kLJrxLT5ZOrpGgrIDajt
J8nU57O5q4IikCc9Kuh8kO+8T/3iCiSn3mUkpF3qwHYw03dQ+A0Em5Q2AXPKBlim
0zvc+gRGE1WKyURHuFE5Gi7oNOJ5y1lKCn+8pu8fA2dqWSslYpPZUxlmPCdiKYZN
pGvu/9ROutW04o5IWgAZCfEF2c6Rsffr6TlP9m8EQ5pV9T4FFL2/s1m02I4zhKOQ
UqqzApVg+QxMaPnu1RcN+HFXtSXkKe5lXa/R7jwXC1pDxaWG6iSe4gUH3DRCEpHW
OXSuTEGC2/KmSNGzm/MzqvOmwMVO9fSddmPmAsYiS8GVP1BkLFTltvA8Kc9XAgMB
AAGjQjBAMB0GA1UdDgQWBBRUYnBj8XWEQ1iO0RYgscasGrz2iTAPBgNVHRMBAf8E
BTADAQH/MA4GA1UdDwEB/wQEAwIBBjANBgkqhkiG9w0BAQsFAAOCAgEAKbqSSaet
8PFww+SX8J+pJdVrnjT+5hpk9jprUrIQeBqfTNqK2uwcN1LgQkv7bHbKJAs5EhWd
nxEt/Hlk3ODg9d3gV8mlsnZwUKT+twpw1aA08XXXTUm6EdGz2OyC/+sOxL9kLX1j
bhd47F18iMjrjld22VkE+rxSH0Ws8HqA7Oxvdq6R2xCOBNyS36D25q5J08FsEhvM
Kar5CKXiNxTKsbhm7xqC5PD48acWabfbqWE8n/Uxy+QARsIvdLGx14HuqCaVvIiv
TDUHKgLKeBRtRytAVunLKmChZwOgzoy8sHJnxDHO2zTlJQNgJXtxmOTAGytfdELS
S8VZCAeHvsXDf+eW2eHcKJfWjwXj9ZtOyh1QRwVTsMo554WgicEFOwE30z9J4nfr
I8iIZjs9OXYhRvHsXyO466JmdXTBQPfYaJqT4i2pLr0cox7IdMakLXogqzu4sEb9
b91fUlV1YvCXoHzXOP0l382gmxDPi7g4Xl7FtKYCNqEeXxzP4padKar9mK5S4fNB
UvupLnKWnyfjqnN9+BojZns7q2WwMgFLFT49ok8MKzWixtlnEjUwzXYuFrOZnk1P
Ti07NEPhmg4NpGaXutIcSkwsKouLgU9xGqndXHt7CMUADTdA43x7VF8vhV929ven
sBxXVsFy6K2ir40zSbofitzmdHxghm+Hl3s=
-----END CERTIFICATE-----
```

CA Hierarchy

- Need for Intermediate-Level CAs:
  - the public keys for the Root CAs come pre-loaded with your computer (and also with the browsers).
  - If the private key of a Root CA become compromised for some reason, we need to update the software.
  - Not possible to update all computers and devices to update.
  - You don't run into this problem when the private key of an Intermediate-Level CA is compromised.
  - The affected certificates can now simply be added to a "Certificate Revocation List" maintained by a higher-level CA.
  - The affected CA can then proceed to issue fresh certificates to the affected parties.

Certification Validation

- Consider a certificate issued by a CA that is not just below the root in the tree of CAs, but somewhere further down in the tree.
- Before your browser trusts such a certificate, it will verify the public key of the next higher level CA that validated the certificate your browser has received. This process is recursive until the root certificate that is pre-loaded in your computer is invoked.
- In order to save your browser from having to make repeated requests for the certificates as it goes up the tree of CAs, the webserver that sent you the certificate you are specifically interested in may send the whole bundle of higher level certificates also.
- A CA 1) authenticates a certificate, 2) the owner of the certificate and 3)its public key.
- If an entity C steals certificate of A, does it impose any threat?

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

(Party A initiates a request for the link)

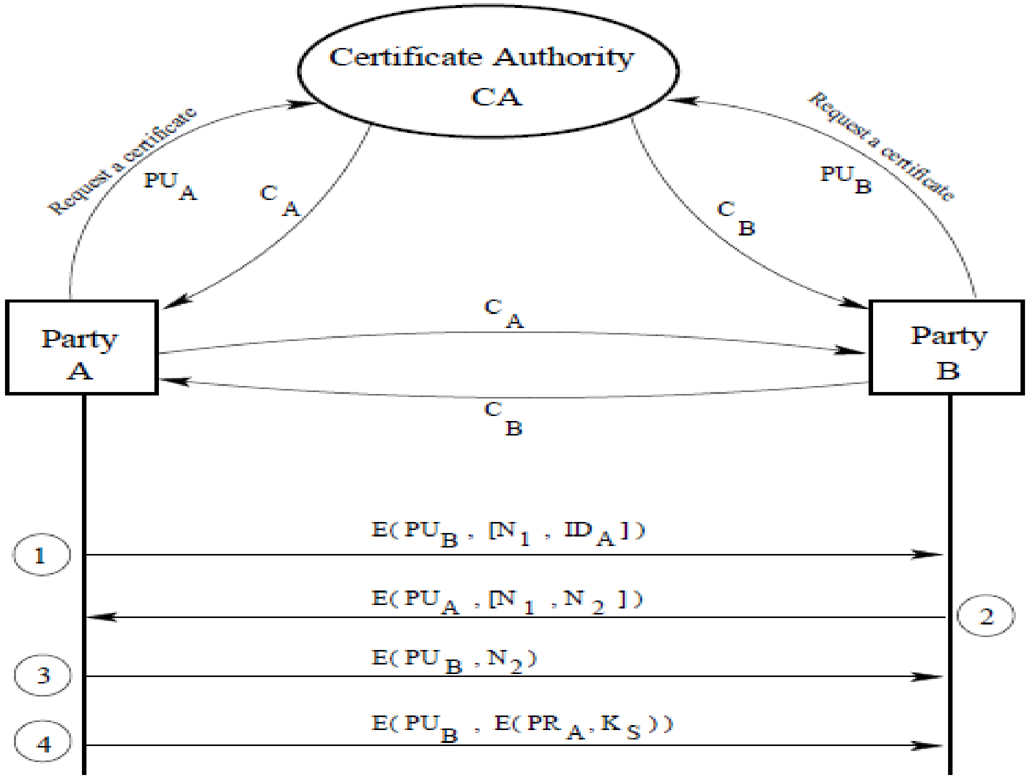


Figure 1: *Messages exchanged between two parties for acquiring each other's CA authenticated public keys.* (*This figure is from Lecture 13 of "Computer and Network Security" by Avi Kak.*)

Case Study

- Suppose you are browsing *engineering.purdue.edu* to access the course material for this lecture.
- *engineering.purdue.edu* uses certificate signed by *InCommonCA*, an intermediate CA.
- The certificate of the *InCommonCA* is signed by a root CA known as *AddTrust* CA.
- *AddTrust*'s public is available in */etc/ssl/cert/*

Case Study

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            7f:71:c1:d3:a2:26:b0:d2:b1:13:f3:e6:81:67:64:3e
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=SE, O=AddTrust AB, OU=AddTrust External TTP Network, CN=AddTrust External CA Roc
        Validity
            Not Before: Dec  7 00:00:00 2010 GMT
            Not After : May 30 10:48:38 2020 GMT
        Subject: C=US, O=Internet2, OU=InCommon, CN=InCommon Server CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:97:7c:c7:c8:fe:b3:e9:20:6a:a3:a4:4f:8e:8e:
                    34:56:06:b3:7a:6c:aa:10:9b:48:61:2b:36:90:69:
                    e3:34:0a:47:a7:bb:7b:de:aa:6a:fb:eb:82:95:8f:
                    ca:1d:7f:af:75:a6:a8:4c:da:20:67:61:1a:0d:86:
                    c1:ca:c1:87:af:ac:4e:e4:de:62:1b:2f:9d:b1:98:
                    af:c6:01:fb:17:70:db:ac:14:59:ec:6f:3f:33:7f:
                    a6:98:0b:e4:e2:38:af:f5:7f:85:6d:0e:74:04:9d:
                    f6:27:86:c7:9b:8f:e7:71:2a:08:f4:03:02:40:63:
                    24:7d:40:57:8f:54:e0:54:7e:b6:13:48:61:f1:de:
                    ce:0e:bd:b6:fa:4d:98:b2:d9:0d:8d:79:a6:e0:aa:
                    cd:0c:91:9a:a5:df:ab:73:bb:ca:14:78:5c:47:29:
                    a1:ca:c5:ba:9f:c7:da:60:f7:ff:e7:7f:f2:d9:da:
                    a1:2d:0f:49:16:a7:d3:00:92:cf:8a:47:d9:4d:f8:
                    d5:95:66:d3:74:f9:80:63:00:4f:4c:84:16:1f:b3:
                    f5:24:1f:a1:4e:de:e8:95:d6:b2:0b:09:8b:2c:6b:
                    c7:5c:2f:8c:63:c9:99:cb:52:b1:62:7b:73:01:62:
                    7f:63:6c:d8:68:a0:ee:6a:a8:8d:1f:29:f3:d0:18:
                    ac:ad
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:AD:BD:98:7A:34:B4:26:F7:FA:C4:26:54:EF:03:BD:E0:24:CB:54:1A

            X509v3 Subject Key Identifier:
                48:4F:5A:FA:2F:4A:9A:5E:E0:50:F3:6B:7B:55:A5:DE:F5:BE:34:5D
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Certificate Policies:
                Policy: X509v3 Any Policy

            X509v3 CRL Distribution Points:
                URI:http://crl.usertrust.com/AddTrustExternalCARoot.crl

            Authority Information Access:
                CA Issuers - URI:http://crt.usertrust.com/AddTrustExternalCARoot.p7c
                CA Issuers - URI:http://crt.usertrust.com/AddTrustUTNSGCCA.crt
                OCSP - URI:http://ocsp.usertrust.com

    Signature Algorithm: sha1WithRSAEncryption
        93:66:21:80:74:45:85:4b:c2:ab:ce:32:b0:29:fe:dd:df:d6:
        24:5b:bf:03:6a:6f:50:3e:0e:1b:b3:0d:88:a3:5b:ee:c4:a4:
        12:3b:56:ef:06:7f:cf:7f:21:95:56:3b:41:31:fe:e1:aa:93:
        d2:95:f3:95:0d:3c:47:ab:ca:5c:26:ad:3e:f1:f9:8c:34:6e:
        11:be:f4:67:e3:02:49:f9:a6:7c:7b:64:25:dd:17:46:f2:50:
        e3:e3:0a:21:3a:49:24:cd:c6:84:65:68:67:68:b0:45:2d:47:
        99:cd:9c:ab:86:29:11:72:dc:d6:9c:36:43:74:f3:d4:97:9e:
        56:a0:fe:5f:40:58:d2:d5:d7:7e:7c:c5:8e:1a:b2:04:5c:92:
        66:0e:85:ad:2e:06:ce:c8:a3:d8:eb:14:27:91:de:cf:17:30:
        81:53:b6:66:12:ad:37:e4:f5:ef:96:5c:20:0e:36:e9:ac:62:
        7d:19:81:8a:f5:90:61:a6:49:ab:ce:3c:df:e6:ca:64:ee:82:
        65:39:45:95:16:ba:41:06:00:98:ba:0c:56:61:e4:c6:c6:86:
        01:cf:66:a9:22:29:02:d6:3d:cf:c4:2a:8d:99:de:fb:09:14:
        9e:0e:d1:d5:c6:d7:81:dd:ad:24:ab:ac:07:05:e2:1d:68:c3:
        70:66:5f:d3
```


```
-----BEGIN CERTIFICATE-----
MIIEwzCCA6ugAwIBAgIQf3HB06ImsNKxE/PmgWdkPjANBgkqhkiG9w0BAQUFADBv
MQswCQYDVQQGEwJTRTEUMBIGA1UEChMLQWRkVHJ1c3QgQUIxJjAkBgNVBAsTHUFk
ZFRydXN0IEV4dGVybmFsIFRUUCBOZXR3b3JrMSIwIAYDVQQDExlBZGRUcnVzdCBF
eHRlcm5hbCBDQSBSb290MB4XDTEwMTIwNzAwMDAwMFoXDTIwMDUzMDEwNDgzOFow
UTELMAkGA1UEBhMCVVMxEjAQBgNVBAoTCUludGVybmV0MjERMA8GA1UECxMISW5D
b21tb24xGzAZBgNVBAMTEkluQ29tbW9uIFNlcnZlciBDQTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAJd8x8j+s+kgaqOkT46ONFYGs3psqhCbSGErNpBp
4zQKR6e7e96qavvrgpWPyh1/r3WmqEzaIGdhGg2GwcrBh6+sTuTeYhsvnbGYr8YB
+xdw26wUWexvPzN/ppgL5OI4r/V/hW0OdASd9ieGx5uP53EqCPQDAkBjJH1AV49U
4FR+thNIYfHezg69tvpNmLLZDY15puCqzQyRmqXfq3O7yhR4XEcpocrFup/H2mD3
/+d/8tnaoS0PSRan0wCSz4pH2U341ZVm03T5gGMAT0yEFh+z9SQfoU7e6JXWsgsJ
iyxrx1wvjGPJmctSsWJ7cwFif2Ns2Gig7mqojR8p89AYrKOCAwEAAaOCAXcwggFz
MB8GA1UdIwQYMBaAFK29mHo0tCb3+sQmVO8DveAky1QaMB0GA1UdDgQWBBRIT1r6
L0qaXuBQ82t7VaXe9b40XTAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0TAQH/BAgwBgEB
/wIBADARBgNVHSAECjAIMAYGBFUdIAAwRAYDVR0fBD0wOzA5oDegNYYzaHR0cDov
L2NybC51c2VydHJ1c3QuY29tL0FkZFRydXN0RXh0ZXJuYWxDQVJvb3QuY3JsMIGz
BggrBgEFBQcBAQSBpjCBozA/BggrBgEFBQcwAoYzaHR0cDovL2NydC51c2VydHJ1
c3QuY29tL0FkZFRydXN0RXh0ZXJuYWxDQVJvb3QucDdjMDkGCCsGAQUFBzAChi1o
dHRwOi8vY3J0LnVzZXJ0cnVzdC5jb20vQWRkVHJ1c3RVVE5TR0NDQS5jcnQwJQYI
KwYBBQUHMAGGGWh0dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20wDQYJKoZIhvcNAQEF
BQADggEBAJNmIYB0RYVLwqvOMrAp/t3f1iRbvwNqb1A+DhuzDYijW+7EpBI7Vu8G
f89/IZVWO0Ex/uGqk9KV85UNPEerylwmrT7x+Yw0bhG+9GfjAkn5pnx7ZCXdF0by
UOPjCiE6SSTNxoRlaGdosEUtR5nNnKuGKRFy3NacNkN089SXnlag/l9AWNLV1358
xY4asgRckmYOha0uBs7Io9jrFCeR3s8XMIFTtmYSrTfk9e+WXCADNumsYn0ZgYr1
kGGmSavOPN/mymTugmU5RZUWukEGAJi6DFZh5MbGhgHPZqkiKQLWPc/EKo2Z3vsJ
FJ4D0dXG14HdrSSrrAcF4h1ow3BmX9M=
-----END CERTIFICATE-----
```

Do Yourself

- Visit a website from google chrome and firefox browser. Now view the certificate.

Creating Your Own Certificate

```
anu@anu-Vostro-5471:~$ openssl req -new -newkey rsa:1024 -days 365 -nodes -x509 -keyout test1.pem -out test1.cert
..........................................................................+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.......+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:61
State or Province Name (full name) [Some-State]:nsw
Locality Name (eg, city) []:sydney
Organization Name (eg, company) [Internet Widgits Pty Ltd]:temp com
Organizational Unit Name (eg, section) []:ict
Common Name (e.g. server FQDN or YOUR name) []:.
Email Address []:.
```

```
anu@anu-Vostro-5471:~$ cat test1.cert
-----BEGIN CERTIFICATE-----
MIICdjCCAd+gAwIBAgIUJivkm1SPLNr4zsevrB5nrA+20aQwDQYJKoZIhvcNAQEL
BQAwTTELMAkGA1UEBhMCNjExDDAKBgNVBAgMA25zdzEPMA0GA1UEBwwGc3lkbmV5
MREwDwYDVQQKDAh0ZW1wIGNvbTEMMAoGA1UECwwDaWN0MB4XDTIzMDkxMzEzNDgw
MloXDTI0MDkxMjEzNDgwMlowTTELMAkGA1UEBhMCNjExDDAKBgNVBAgMA25zdzEP
MA0GA1UEBwwGc3lkbmV5MREwDwYDVQQKDAh0ZW1wIGNvbTEMMAoGA1UECwwDaWN0
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQChUH/ywOEc9jsMLeNTcJKUGBM5
3yHAMftI4aMqpWNt97Xa2A0KUcqgKmOh/1oCVAP0fDzzYsV9oZhPsdiFZUd4PYHv
36OClFgAgOUErt5Act10m838Q5S7+vUm8pZP1FUcSv/mAif0r8w2M4mVFiLFyxUR
jpvxvy1bK2Uf0oP9+QIDAQABo1MwUTAdBgNVHQ4EFgQUUWoPQO/Z08JraMAehRYu
bmdzEbMwHwYDVR0jBBgwFoAUUWoPQO/Z08JraMAehRYubmdzEbMwDwYDVR0TAQH/
BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOBgQCeRovl8mDd3pVMXhF9sgMlEBA+H3gA
17/L/9tnkG6TqZ6giolkwH7WKojmLjJXHGQrgYJVGPo+nojCyAZ05OivIqEa2M2S
4e6wKN1rROi53yiOTrmSxM8viBBUfxnPLYx4HZ+6KmJHbBLomUdWPTmPuszeBxss
9Eda1hRR7zi51g==
-----END CERTIFICATE-----
anu@anu-Vostro-5471:~$
```

```
anu@anu-Vostro-5471:~$ ls
Desktop    Downloads  file.enc  key1.pem  Music     Public       public.pem  Templates  test1.pem  test.pem
Documents  file.dec   file.txt  key.pem   Pictures  public1.pem  snap        test1.cert test.cert  Videos
anu@anu-Vostro-5471:~$ cat test1.pem
-----BEGIN PRIVATE KEY-----
MIICeQIBADANBgkqhkiG9w0BAQEFAASCAmMwggJfAgEAAoGBAKFQf/LA4Rz2Owwt
41NwkpQYEznfIcAx+0jhoyqlY233tdrYDQpRyqAqY6H/WgJUA/R8PPNixX2hmE+x
2IVlR3g9ge/fo4KUWACA5QSu3kBy3XSbzfxDlLv69Sbylk/UVRxK/+YCJ/SvzDYz
iZUWIsXLFRGOm/G/LVsrZR/Sg/35AgMBAAECgYEAnYqEqot8TlCbMjXOgTq7rC4m
+KnVyGIHyxGxzIBhLpBw5h2B/sYKYYmEbD15pjRu+GItFHUt8pfSrGI/12cl99cK
qPWtDMLXt4MEsGzIUo9ky6p8LSsjEgO7SHDfAP7Zv2auCBSfIvwRgLIJMVjmkzCr
0E8A1co6fnA/FyipBkECQQDMPw60nC8Ff/acH588pnsfFVc34qwVwC4T/yFjPBTX
pdWLsNEzzjxCwiiUYc2Eh4gQ41GcAwHDEstkErgCwz4lAkEAyjCPijLZlQBpJAfD
KZRT4SbExlRlYWTkhou9WxoZVW+nw3aB/RYwsY+gNNWgjgn4R1dsa+3jksPkc0n+
DyPmRQJBAKfO4zEm5Uc6hI5vOLBnA99c9ETZOpPASttpEbBri6BGAvZ7dtZ+imo7
BYYr+OP7SqK9ca6hldAFYBA/hOnYKfUCQQChZSYPcQvSLuO+yRt2o1pZjbLhhhjv
J7Rr2jwq7qhuVJScKIsW1ZHFCxsdUbG58Cdp+1UmylwwmYJQrS3KNCPpAkEAo9LC
gAW6wgfFyy9lS2eBel+uwUBQoKNDCL9bz+1HJJTlMjwsyTBUD5f4xRnijKJj7L7/
XQN9hgU/m61T4+fsdQ==
-----END PRIVATE KEY-----
```

Resources

- Chapter 13 of "Cryptography and Network Security (Principles and Practice)" by William Stallings
- Slides 1 – 33 of Lecture 13 from https://engineering.purdue.edu/kak/compsec/.

Acknowledgment

Lecture slides 39-49 are adapted from lecture 13 of Avinash Kak from https://engineering.purdue.edu/kak/compsec/.