Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)



Choose random v

$y_a = a^v \bmod p$



Choose random w, Compute

$K_{ba} = (y_a)^w \bmod p$

$y_b = a^w \bmod p$

$SIG_{bob} = \{y_b, y_a\}^{Bob}$

$CERT_{bob}, y_b, SIG_{bob}$

Compute

$K_{ab} = (y_b)^v \bmod p$

$SIG_{alice} = \{y_a, y_b\}^{alice}$

$CERT_{alice}, SIG_{alice}$

1

1

x.509 Authentication & Key Distribution Protocols

One-msg A→B



$\{1, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$



Two-msg A→B



$\{2, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$

$\{2, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A}\}_{SK_B}$



Three-msg A←→B



$\{3, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$

$\{3, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A}\}_{SK_B}$

$\{3, r_b\}_{SK_A}$



2

2

Lecture 11

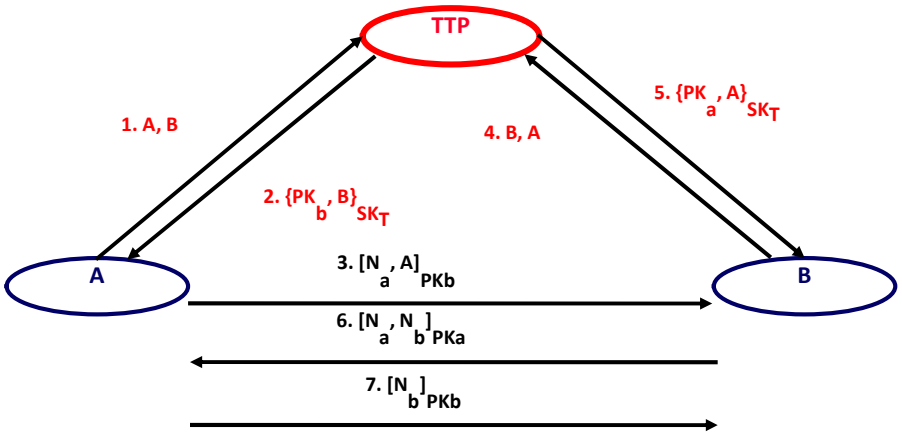
Public Key Distribution (and Certification)

(Chapter 15 in KPS)

3

3

Recall PK-based Needham-Schroeder



Here, TTP acts as an "on-line" certification authority (CA) and takes care of revocation

4

4

What If?

- Alice and Bob have:
  - No common mutually trusted TTP(s)

    and/or

  - No on-line TTP(s)

5

5

Trusted Intermediaries, or Trusted 3rd Party (TTP)

Symmetric Key Problem:

- How do two entities establish shared secret key over a distance (i.e., over a network)?

Solution:

- Mutually trusted **on-line** key distribution center (KDC) acts as TTP

Public Key Problem:

- When Alice gets Bob's public key (from a web site, email, disk, bboard), how does she know it is really Bob's?

Solution:

- Trusted **off-line** certification authority (CA) as a TTP
- Or, a hierarchy thereof, called a PKI

6

6

Public Key Infrastructure (Distribution)

Problem: How to determine the correct public key of a given entity, i.e., a **binding** between IDENTITY and PUBLIC KEY

- Possible Attacks
  - Name spoofing: Eve binds Alice's name with Eve's public key
  - Key spoofing: Eve binds Alice's key with Eve's name
  - DoS: Eve binds Alice's name with a random key
- What happens in each case?

7

7

Public Key Distribution

- Popek - Kline (1979) proposed "trusted third parties" (TTPs) as a means of PK distribution:
  - Each org-n has a TTP that knows public keys of all of its constituent entities and distributes them on-demand
  - On-line protocol like the one we already saw
  - TTP = single point of failure
  - Denial-of-Service (DoS) attacks

8

8

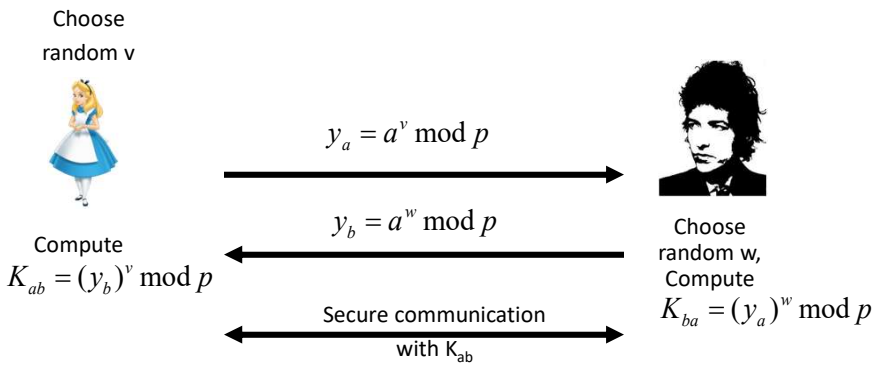
Certificates

- Kohnfelder (BS Thesis, MIT, 1978) proposed "certificates" as yet another public-key distribution method
- Certificate = an explicit binding between a public key and its owner's (unique) name
- Must be issued (and signed) by a recognized trusted Certificate Authority (CA)
- Issuance is done **off-line**

9

9

Recall Diffie-Hellman Interactive (Public) Key Exchange



10

10

The Man-in-the-Middle (MitM) Attack

(assume Eve is active!)

Choose random u

K_{ae} K_{eb}

Choose random v



$y_a = a^v \bmod p$

$y_e = a^u \bmod p$

Secure communication using K_{ae}



Compute $K_{ae} = a^{vu} \bmod p$

Secure communication using K_{eb}

$y_b = a^w \bmod p$

y_e

Root cause: Alice has no way to verify whether y_e is associated with Bob's identity!

Choose random w, Compute $K_{eb}^{11} = a^{uw} \bmod p$



11

Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)

$CERT_{alice} = SK_{CA}\{PK_A, \text{“Alice”}\}$

$CERT_{bob} = SK_{CA}\{PK_B, \text{“Bob”}\}$

Choose random v





$y_a = a^v \bmod p$

Choose random w, Compute $K_{ba} = (y_a)^w \bmod p$

$CERT_{bob}, y_b, SIG_{bob}$

$y_b = a^w \bmod p$

Compute $K_{ab} = (y_b)^v \bmod p$

$SIG_{alice} = SK_A(y_a, y_b)$

$CERT_{alice}, SIG_{alice}$

$SIG_{bob} = SK_B(y_b, y_a)$

12

12

Certificates

- Procedure
  - Bob generated SK_B, PK_B, registers PK_B with CA
  - Bob receives his certificate:

$CERT_B = SK_{CA}\{PK_B$, "Bob", issuance_time, expiration_time, etc.,...$\}$

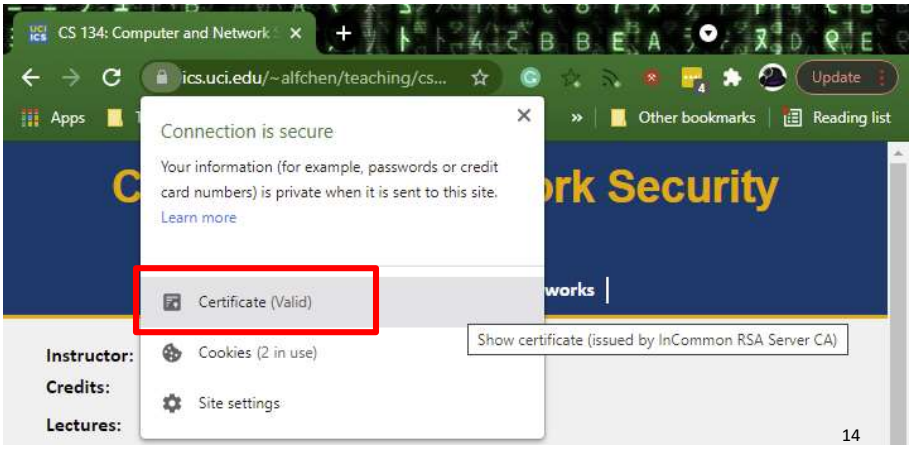
  - Bob sends $CERT_B$ to Alice
  - Alice verifies CA's signature, checks expiration, ...
    - PK_{CA} hard-coded in Alice's (software or hardware)
  - Alice uses PK_B to encrypt for Bob and/or verifying Bob's signatures

13

13

Recall last lecture…

Open your browser to check it yourself!



14

Who Issues Certificates?

CA: **Certification Authority**

- e.g., GlobalSign, VeriSign, Thawte, etc.
- look into your browser or smartphone...

CA must:

- Be trustworthy (at least to its users/clients)
- Operate off-line (usually)
- Have own well-known long-term certificate
- Store (as backup/optionally) issued certificates
- Be very secure: physically and electronically

15

15

How does it work?

- A public/private key-pair is generated by a user
- User requests certificate via a local application (e.g., web browser, email, in person, etc.)
  - Good idea to prove knowledge of private key as part of the certificate request. Why?
- Public key and owner's name are part of a certificate
- Private keys only used for small amount of data (signing, encryption of session keys)
- Symmetric keys (e.g., AES) used for bulk data encryption

16

16

Certification Authority (CA)

- CA must verify/authenticate the entity requesting a new certificate.
- CA's own certificate is signed by a higher-level CA. Root CA's certificate is self-signed and pre-installed in devices
- CA is a critical part of the system and must operate in a secure and predictable way according to some policy.

17

17

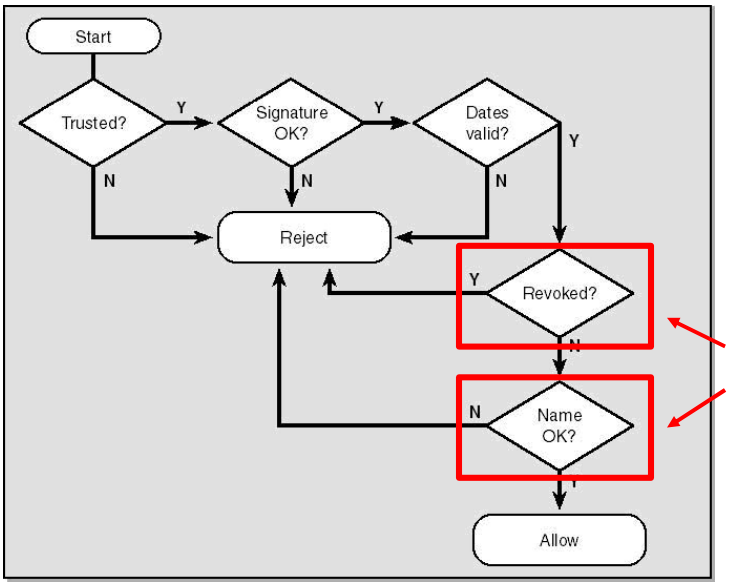
Who needs them?

- Alice's certificate is checked by whomever wants to: 1) verify her signatures, and/or 2) encrypt data for her.
- A signature verifier (or encryptor) must:
  - know the public key of the CA(s)
  - trust all CAs involved
- Certificate checking requires verification of validity and CA signature
- Validity = purpose, expiration, & revocation checking

18

18

Verifying a Certificate (assuming Common CA)



19

19

What are PK Certificates Good For?

- Secure channels in TLS / SSL for web servers
- Signed and/or encrypted email (PGP, S/MIME)
- Authentication (e.g., SFTP, slogin, SSH)
- Code signing (for distribution, updates, etc.)
- Encrypting file-systems (EFS in Windows or IOS)
- IPSec: encryption/authentication at the network layer
- Wireless Security

20

20

Components of a Certification System

- Request and issue certificates (different categories) with verification of identity
- Storage of certificates
- Publishing/distribution of certificates (LDAP, HTTP)
- Pre-installation of root certificates in a trusted environment
- Support by OS platforms, applications and services
- Maintenance of database of issued certificates (no private keys!)
- Helpdesk (information, lost + compromised private keys)
- Advertising revoked certificates (and support for applications to perform revocation checking)
- Storage "guidelines" for private keys

21

21

CA Security

- Must minimize risk of CA private key being compromised
- That's why it's best to have an off-line CA
  - Requests may come in electronically but should not be processed in real time
- In addition, using tamper-resistant hardware for the CA would help (ideally, to make it impossible to extract CA's private key)

22

22

Storage of Private Key

- The problem of having the user to manage the private key (user support, key loss or compromise)
- Modern OS-s offer Protected Storage → saves private keys (encrypted).
- Applications take advantage of this; Browsers sometimes save private keys encrypted in their configuration directory
- Users who mix applications or platforms must manually import / export private keys via PFX files.

23

23

Key Lengths

- A CA should have an (RSA) modulus size of >= 3072 bits given its importance and typical lifetime
- A personal (end-user) RSA public key should have a modulus size of at least 2048 bits

24

24

Key Lengths

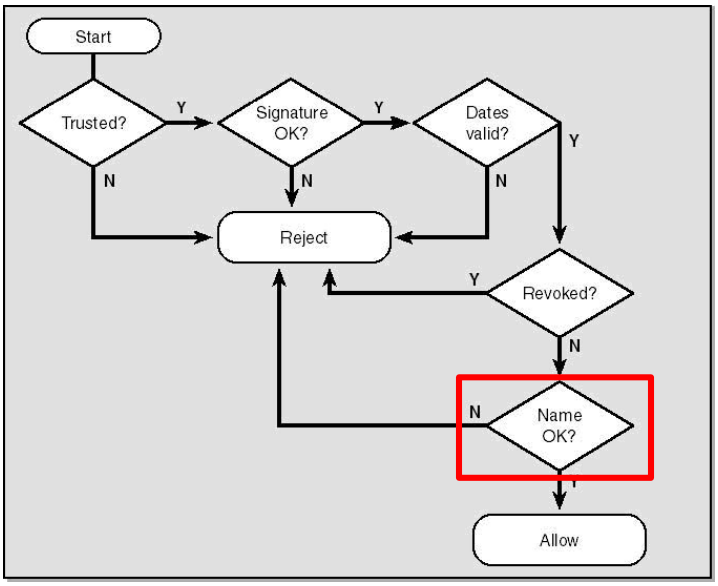
January 2016 Recommendation from the NSA

https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf

Algorithm	Usage
RSA 3072-bit or larger	Key Establishment, Digital Signature
Diffie-Hellman (DH) 3072-bit or larger	Key Establishment
ECDH with NIST P-384	Key Establishment
ECDSA with NIST P-384	Digital Signature
SHA-384	Integrity
AES-256	Confidentiality

Naming Comes First!

Verifying a Certificate (assuming Common CA)



27

27

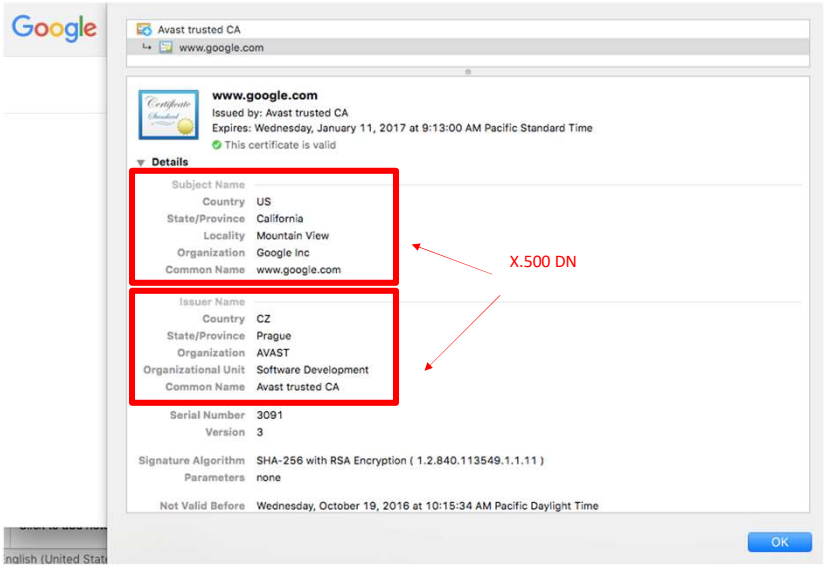
Naming Comes First!

- Can not have certificates without a comprehensive naming scheme
- Can not have PKI without a comprehensive distribution/access method
- X.509 certificate format uses X.500 naming
- X.500 Distinguished Names (DNs) contain a subset of:
  - **C** **Country**
  - **SP** **State/Province**
  - **L** **Locality**
  - **O** **Organization**
  - **OU** **Organizational Unit**
  - **CN** **Common Name**

28

28

A Sample Certificates in Practice (1/3)



29

29

X.500

- ISO standard for directory services
- Global, distributed
- First solid version in 1988. (second in 1993.)
- Documentation - several Internet Standard Request for Comments (RFC)

30

30

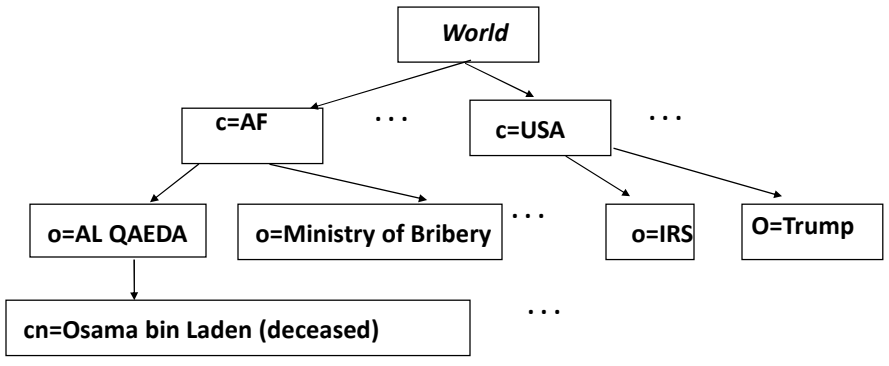
X.500

- Data Model:
  - Based on hierarchical namespace
  - Directory Information Tree (DIT)
  - Geographically organized
  - Entry is defined with its **DN** (Distinguished Name)
- Searching:
  - You must select a location in DIT to base your search
  - A one-level search or a subtree search
  - Subtree search can be slow

31

31

X.500 - DIT



dn: cn=Osama bin Laden, o=Al Qaeda, c=AF

32

32

X.500

- Accessible through:
  - Telnet (client programs known as dua, dish, ...)
  - WWW interface
- Hard to use and very heavy …
  - … thus LDAP was developed

33

33

LDAP

- LDAP - **Lightweight Directory Access Protocol**
- LDAP v2 - RFC 1777, RFC 1778
- LDAP v3 - RFC 1779
- developed to make X.500 easier to use
- provides basic X.500 functions
- referral model instead original chaining
  - server informs client to ask another server (without asking question on the behalf of client)
- LDAP URL format:
  - ldap://server_address/dn
- (ldap://ldap.uci.edu/cn=Alfred Chen,o=UCI,c=US)

34

34

Some Relevant Standards

- The IETF Reference Site
  - http://ietf.org/html.charters/wg-dir.html#Security_Area
- Public-Key Infrastructure (X.509, PKIX)
  - RFC 2459 (X.509 v3 + v2 CRL)
- LDAP v2 for Certificate and CRL Storage
  - RFC 2587
- Guidelines & Practices
  - RFC 2527
- S/MIME v3
  - RFC 2632 & 2633
- TLS 1.0 / SSL v3
  - RFC 2246

Lecture 12

Public Key Certification and Revocation

Certificates

- Kohnfelder (BS Thesis, MIT, 1978) proposed "certificates" as yet another public-key distribution method
- Certificate = **explicit *binding* between a public key and its owner's (unique) name**
- Must be issued (and signed) by a recognized trusted Certificate Authority (CA)
- Issuance is done **off-line**

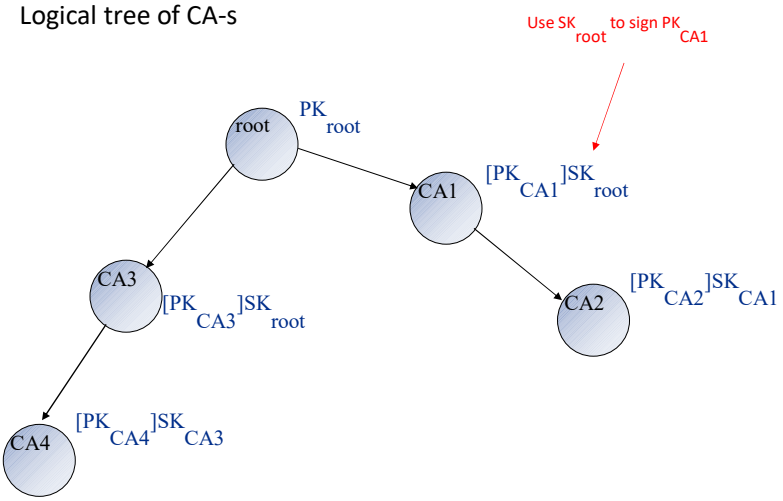
Cannot let root CA to sign all the certifcates in the world --- not scalable

37

37

Certification Tree / Hierarchy

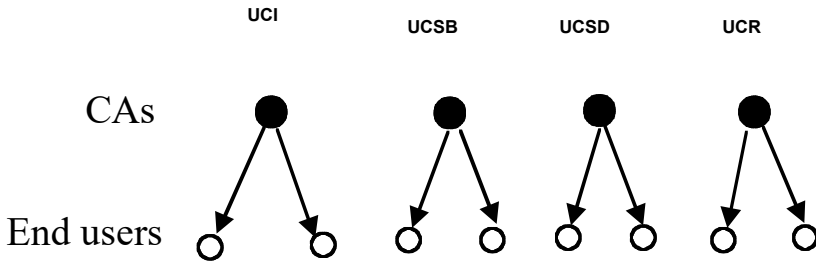
Logical tree of CA-s



38

38

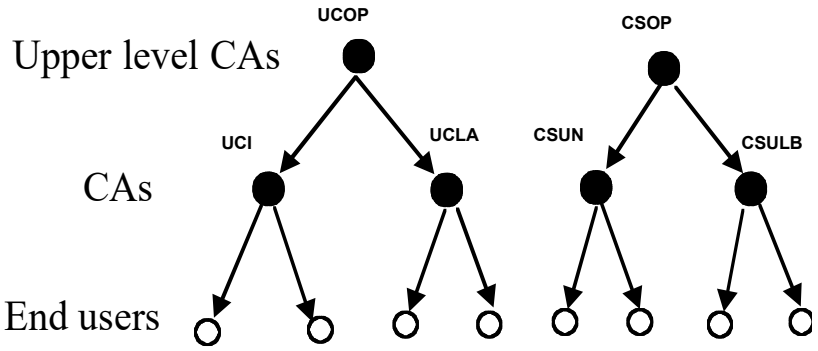
Hierarchical Public Key Infrastructure (PKI) Example



39

39

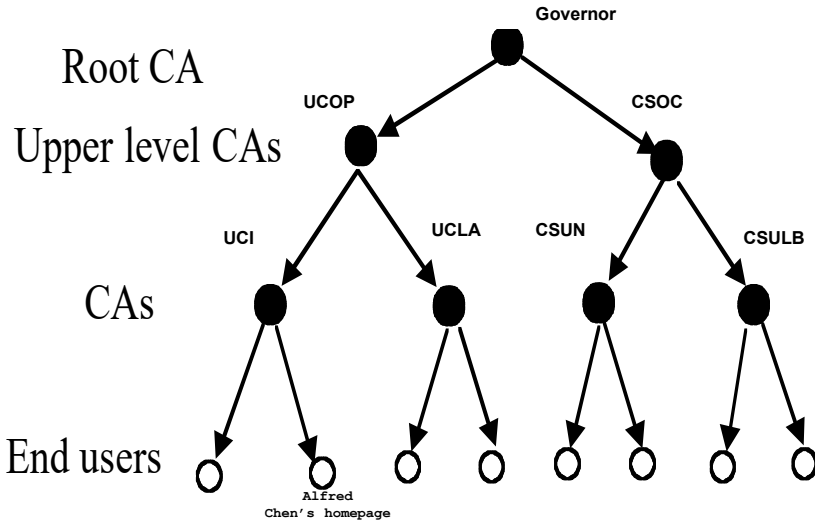
Hierarchical PKI Example



40

40

Hierarchical PKI Example

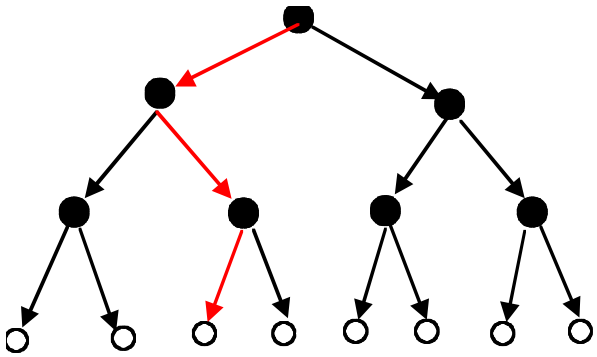


41

41

Certificate Paths

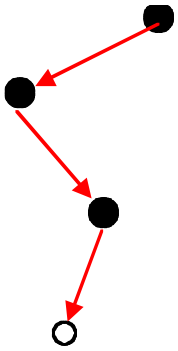
Derived from PKI



42

42

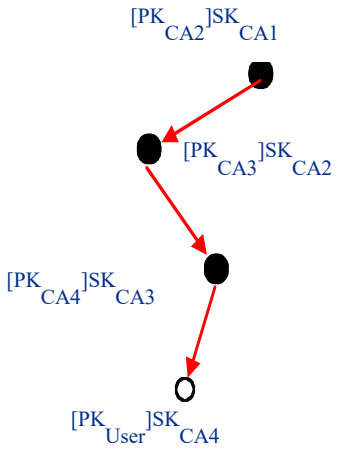
Certificate Paths



43

43

Certificate Paths



- Verifier must know public key of the first CA (usually the root CA)
- Other public keys are 'discovered' one by one
- All CAs on the path must be (implicitly) trusted by the verifier

44

44

X.509 Standard

- X.509v3 is the current version
  - ITU standard
  - ISO 9495-2 is the equivalent ISO standard
- Defines certificate format, not PKI
- **End users cannot be CAs**

45

45

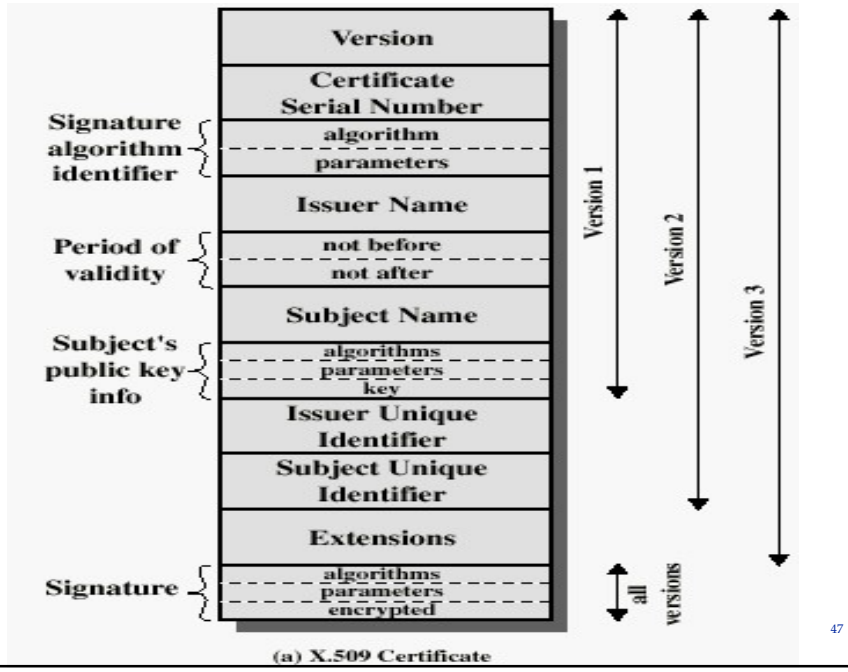
X.509 Certificate Format

- version
- serial number
- signature algorithm ID
- issuer name(X.500 Distinguished Name)
- validity period
- subject(user) name (X.500 Distinguished Name)
- subject public key information
- issuer unique identifier (version 2 and 3 only)
- subject unique identifier (version 2 and 3 only)
- extensions (version 3 only), e.g., revocation info
- signature on the above fields

46

46

X.509 Certificate Format



(a) X.509 Certificate

47

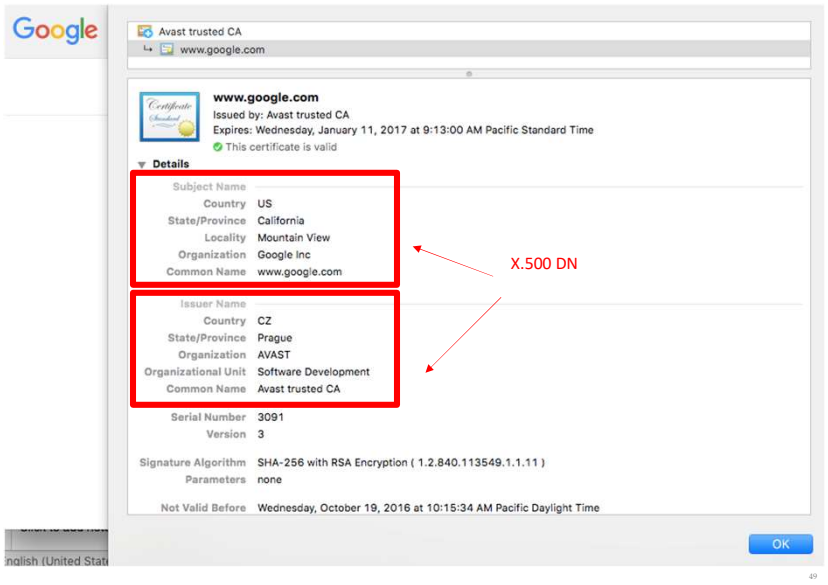
A Sample X.509v3 Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            10:e6:fc:62:b7:41:8a:d5:00:5e:45:b6
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256 - G2
        Validity
            Not Before: Nov 21 08:00:00 2016 GMT
            Not After : Nov 22 07:59:59 2017 GMT
        Subject: C=US, ST=California, L=San Francisco, O=Wikimedia Foundation, Inc., CN=*.wikipedia.org
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
            pub:
                    00:c9:22:69:31:8a:d6:6c:ea:da:c3:7f:2c:ac:a5:
                    af:c0:02:ea:81:cb:65:b9:fd:0c:6d:46:5b:c9:1e:
                    9d:3b:ef
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement
            Authority Information Access:
                CA Issuers - URI:http://secure.globalsign.com/cacert/gsorganizationvalsha2g2r1.crt
                OCSP - URI:http://ocsp2.globalsign.com/gsorganizationvalsha2g2
            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.4146.1.20
                  CPS: https://www.globalsign.com/repository/
                Policy: 2.23.140.1.2.2
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://crl.globalsign.com/gs/gsorganizationvalsha2g2.crl
            X509v3 Subject Alternative Name:
                DNS:*.wikipedia.org, DNS:*.m.mediawiki.org, DNS:*.m.wikibooks.org, DNS:*.m.wikidata.org, DNS:*.m.wikimedia.org, DNS:*.m.wikimediafoundation.org, DNS:*.m.wikinews.org,
DNS:*.m.wikipedia.org, DNS:*.m.wikiquote.org, DNS:*.m.wikisource.org, DNS:*.m.wikiversity.org, DNS:*.m.wikivoyage.org, DNS:*.m.wiktionary.org, DNS:*.mediawiki.org,
DNS:*.planet.wikimedia.org, DNS:*.wikibooks.org, DNS:*.wikidata.org, DNS:*.wikimedia.org, DNS:*.wikimediafoundation.org, DNS:*.wikinews.org, DNS:*.wikiquote.org, DNS:*.wikisource.org,
DNS:*.wikiversity.org, DNS:*.wikivoyage.org, DNS:*.wiktionary.org, DNS:*.wmfusercontent.org, DNS:*.zero.wikipedia.org, DNS:mediawiki.org, DNS:w.wiki, DNS:wikibooks.org,
DNS:wikidata.org, DNS:wikimedia.org, DNS:wikimediafoundation.org, DNS:wikinews.org, DNS:wikiquote.org, DNS:wikisource.org, DNS:wikiversity.org, DNS:wikivoyage.org, DNS:wiktionary.org,
DNS:wmfusercontent.org, DNS:wikipedia.org
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                28:2A:26:2A:57:8B:3B:CE:B4:D6:AB:54:EF:D7:38:21:2C:49:5C:36
            X509v3 Authority Key Identifier:
                keyid:96:DE:61:F1:BD:1C:16:29:53:1C:C0:CC:7D:3B:83:00:40:E6:1A:7C

    Signature Algorithm: sha256WithRSAEncryption
         8b:c3:ed:d1:9d:39:6f:af:40:72:bd:1e:18:5e:30:54:23:35:
         ...
```

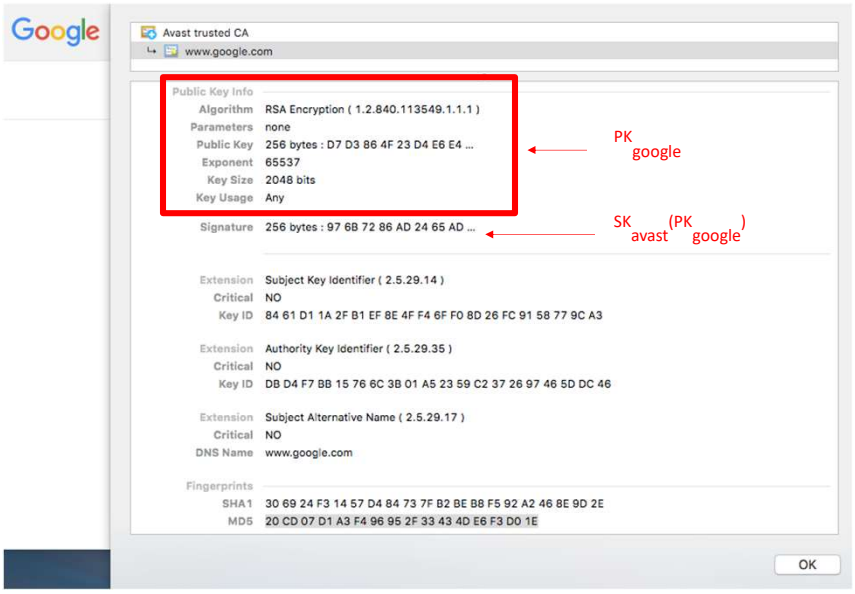
48

A Sample Certificates in Practice (1/3)



49

A Sample Certificates in Practice (2/3)



50

A Sample Certificates in Practice (3/3)


```
-----BEGIN CERTIFICATE-----
MIIDTzCCAvmgAwIBAgIBATANBgkqhkiG9w0BAQQFADBcMSEwHwYDVQQKExhFdXJv
cGVhbiBJQ0UtVEVMIHByb2plY3QxIzAhBgNVBAsTGlYzLUNlcnRpZmljYXRpb24g
QXV0aG9yaXR5MRIwEAYDVQQHEwlEYXJtc3RhZHQwHhcNOTcwNDAyMTczNTU5WhcN
OTgwNDAyMTczNTU5WjBrMSEwHwYDVQQKExhFdXJvcGVhbiBJQ0UtVEVMIHByb2pl
Y3QxIzAhBgNVBAsTGlYzLUNlcnRpZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQH
EwlEYXJtc3RhZHQxDTALBgNVBAMTBFVTRVIwWTAKBgRVCAEBAgICAANLADBIAkEA
qKhTY0kbk8PDC2yIEVXefmri+VKg3GklxMi/VeExqM7kqSmFmYoVmt72L+G0UF9e
BHWm9HbcPA453Dq+PqRhiwIDAQABo4IBmDCCAZQwHwYDVR0jBBgwFoAUfnLy+DqG
nEKINDRmdcPU/NGiETMwHQYDVR0OBBYEFJfc4B8gjSoRmLUx4Sq/ucIYiMrPMA4G
A1UdDwEB/wQEAwIB8DAcBgNVHSABAf8EEjAQMAYGBCoDBAUwBgYECQgHBjBDBgNV
HREEPDA6gRV1c2VyQGRhcm1zdGFkdC5nbWQuZGWGIWh0dHA6Ly93d3cuZGFybXN0
YWR0LmdtZC5kZS9+dXNlcjCBsQYDVR0SBIGpMIGmgQxnbWRjYUBnbWQuZGWGEWh0
dHA6Ly93d3cuZ21kLmRlghdzYXR1cm4uZGFybXN0YWR0LmdtZC5kZaRcMSEwHwYD
VQQKExhFdXJvcGVhbiBJQ0UtVEVMIHByb2plY3QxIzAhBgNVBAsTGlYzLUNlcnRp
ZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQHEwlEYXJtc3RhZHSHDDE0MS4xMi42
Mi4yNjAMBgNVHRMBAf8EAjAAMB0GA1UdHwQWMBQwEqAQoA6BDGdtZGNhQGdtZC5k
ZTANBgkqhkiG9w0BAQQFAANBAGkM4ben8tj76GnAE803rSEGIk3oxtvxBAu34LPW
DIEDzsNqPsfnJCSkkmTCg4MGQlMObwkehJr3b2OblJmD1qQ=
-----END CERTIFICATE-----
```

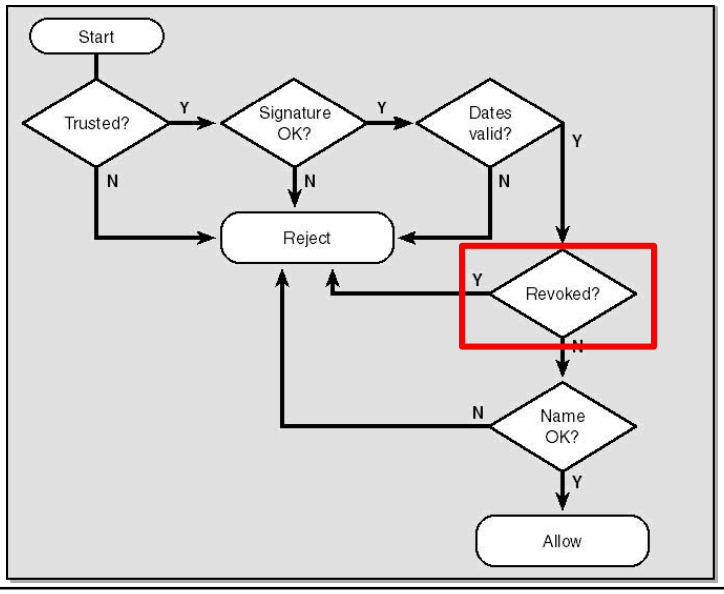

51

Certificates in Practice

- X.509 certificate format is defined in Abstract Syntax Notation 1 (ASN.1)
- ASN.1 structure is encoded using the Distinguished Encoding Rules (DER)
- A DER-encoded binary string is typically base-64 encoded to get an ASCII representation (previous slide)

52

Verifying a Certificate (assuming Common CA)



53

53

Certificate Revocation Scenarios

After a certificate is issued, lots of things can happen before it expires
What if:

- Bob's CA goes out of control?
- Bob left the company?
- Bob forgets his private key?
- Someone steals Bob's private key?
- Bob willingly discloses his private key?

Eve can decrypt/sign while Bob's certificate is still valid!

- Bob reports key loss to CA (or CA finds out somehow)
- CA issues a **Certificate Revocation List (CRL)**
  - Distributed/published in public announcements/databases
- When verifying Bob's signature or encrypting a message for Bob, Alice first checks if Bob's certificate is still valid!
- IMPORTANT: what about signatures Eve generated before Bob realized his key is lost?

54

54

Requirements for Revocation

- Timeliness
  - Before using a certificate, must check most recent revocation status
- Efficiency
  - Computation
  - Bandwidth and Storage
  - Availability
- Security

55