

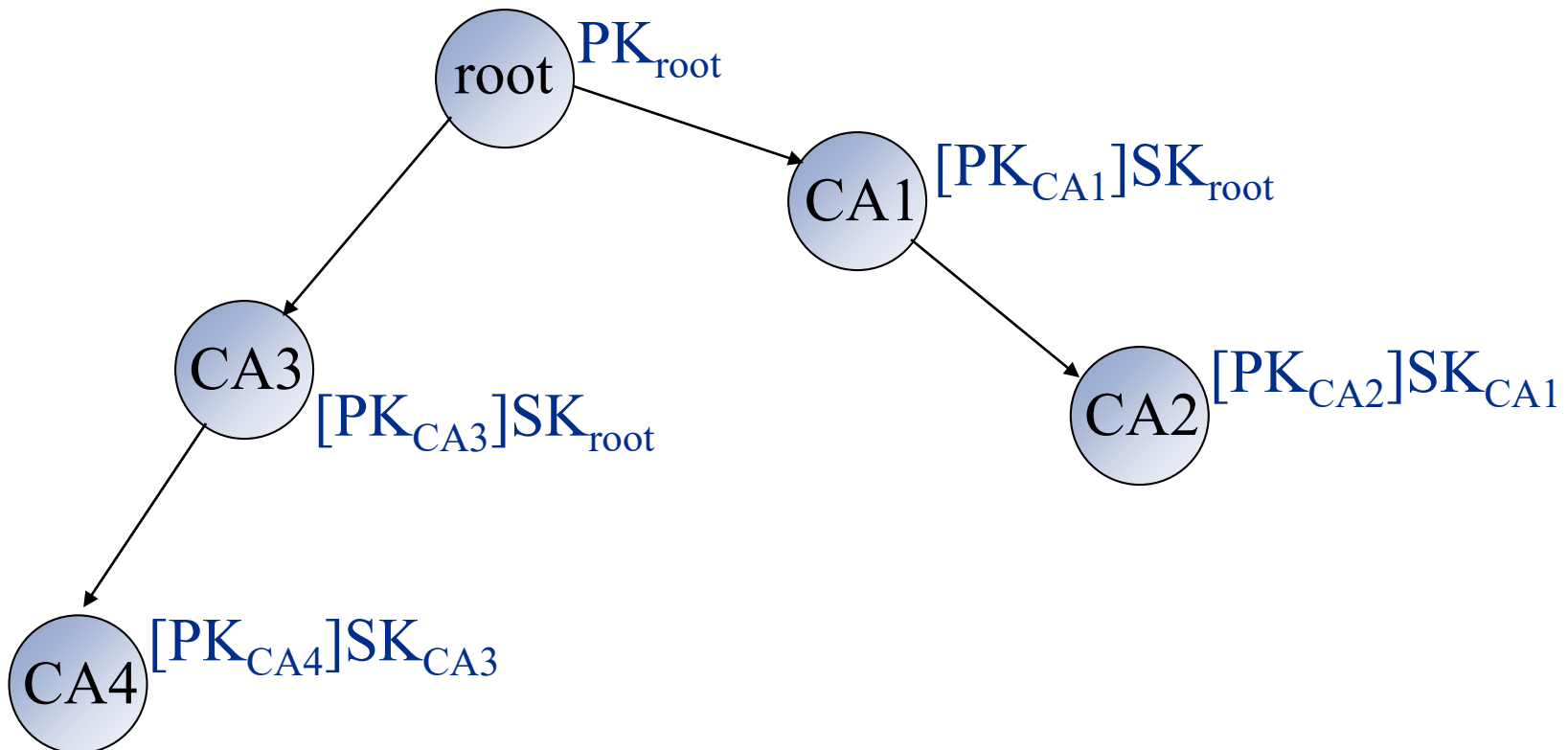
Lecture 12

Public Key Certification and Revocation

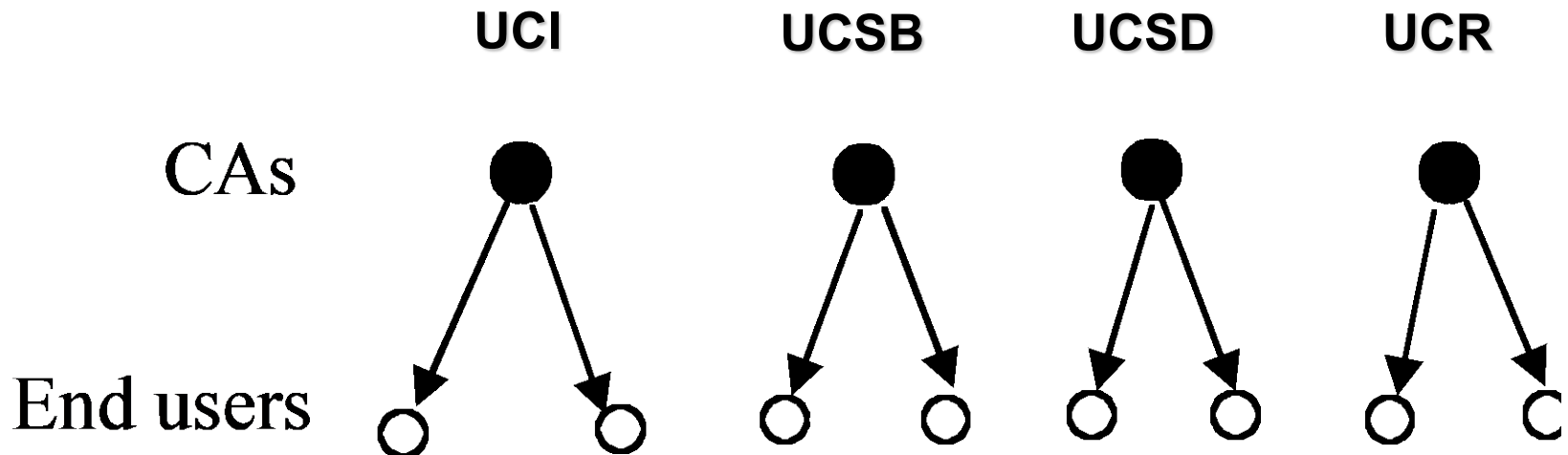
[lecture slides are adapted from previous slides by Prof. Gene
Tsudik]

CertificationTree / Hierarchy

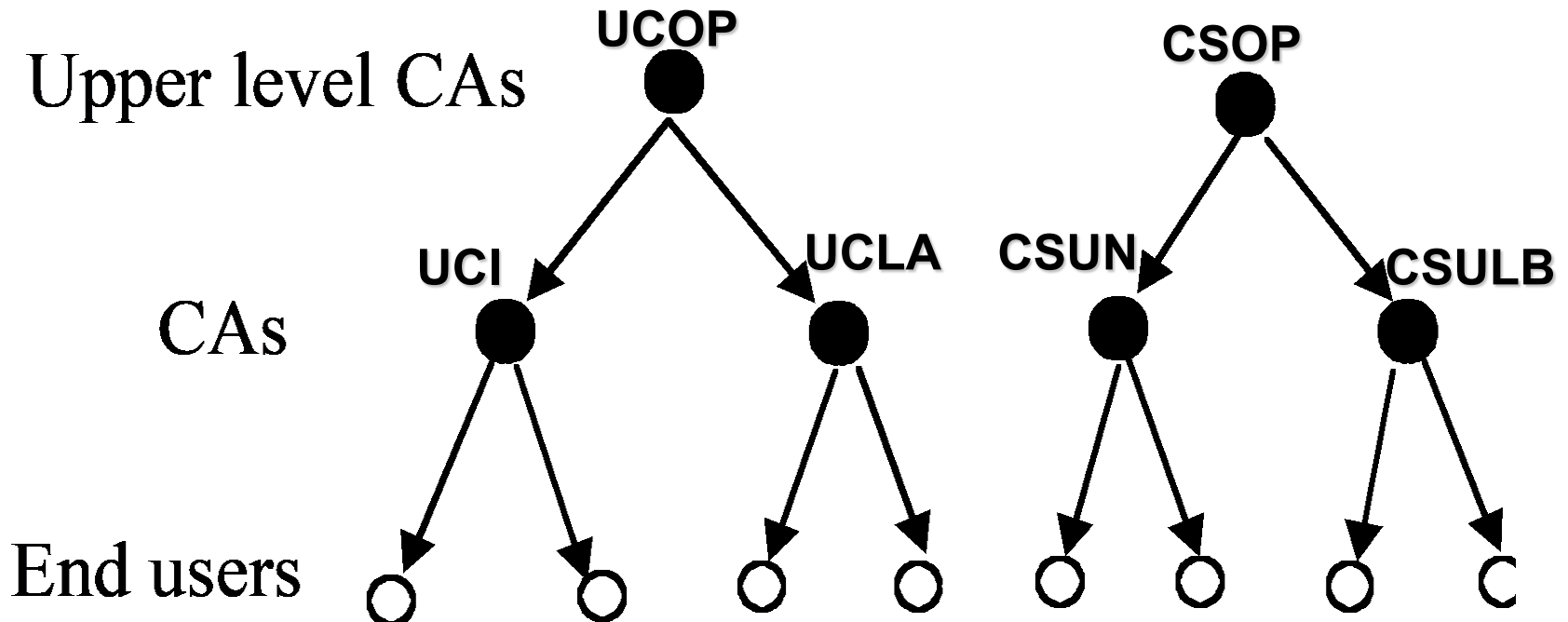
Logical tree of CA-s



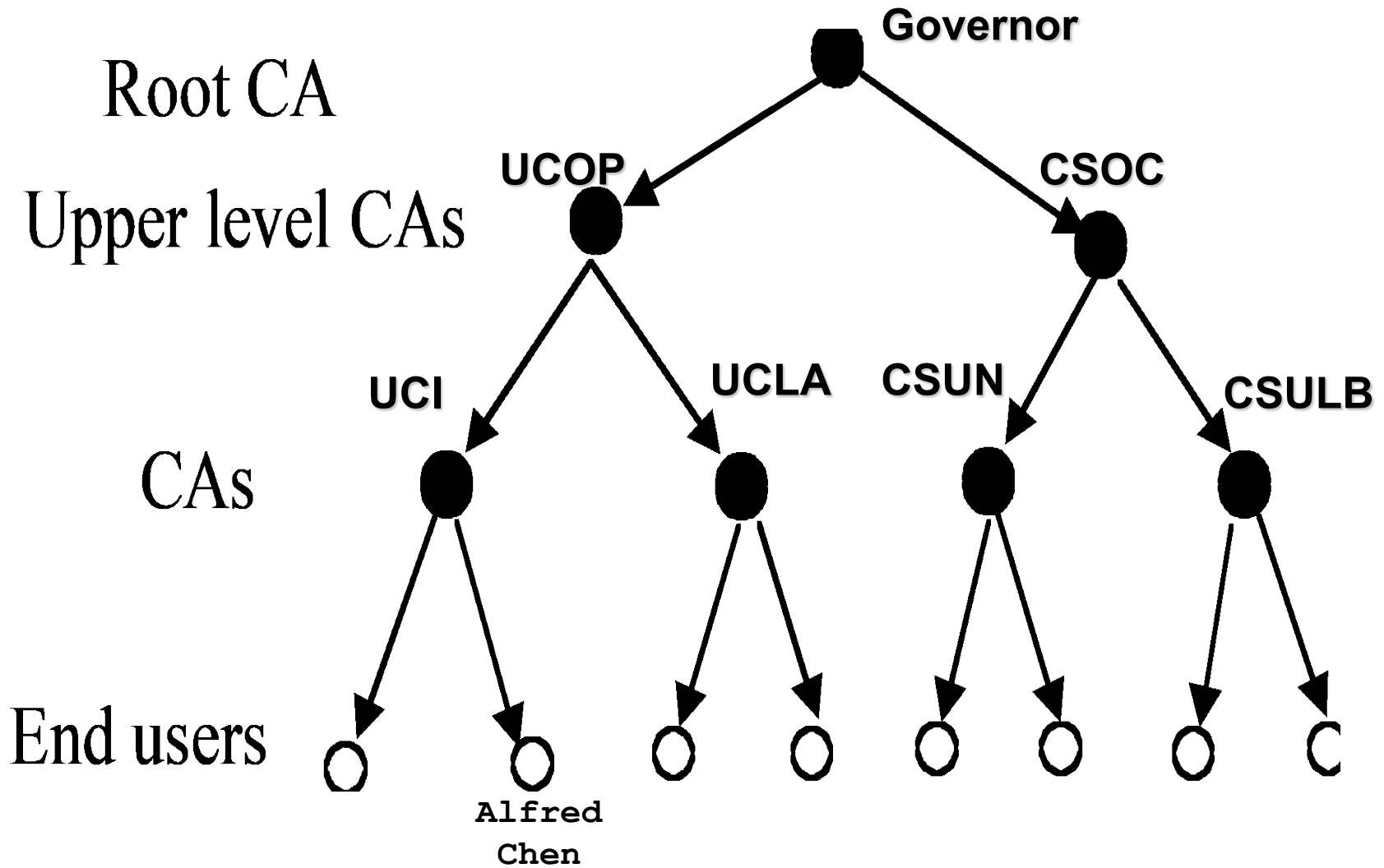
Hierarchical Public Key Infrastructure (PKI) Example



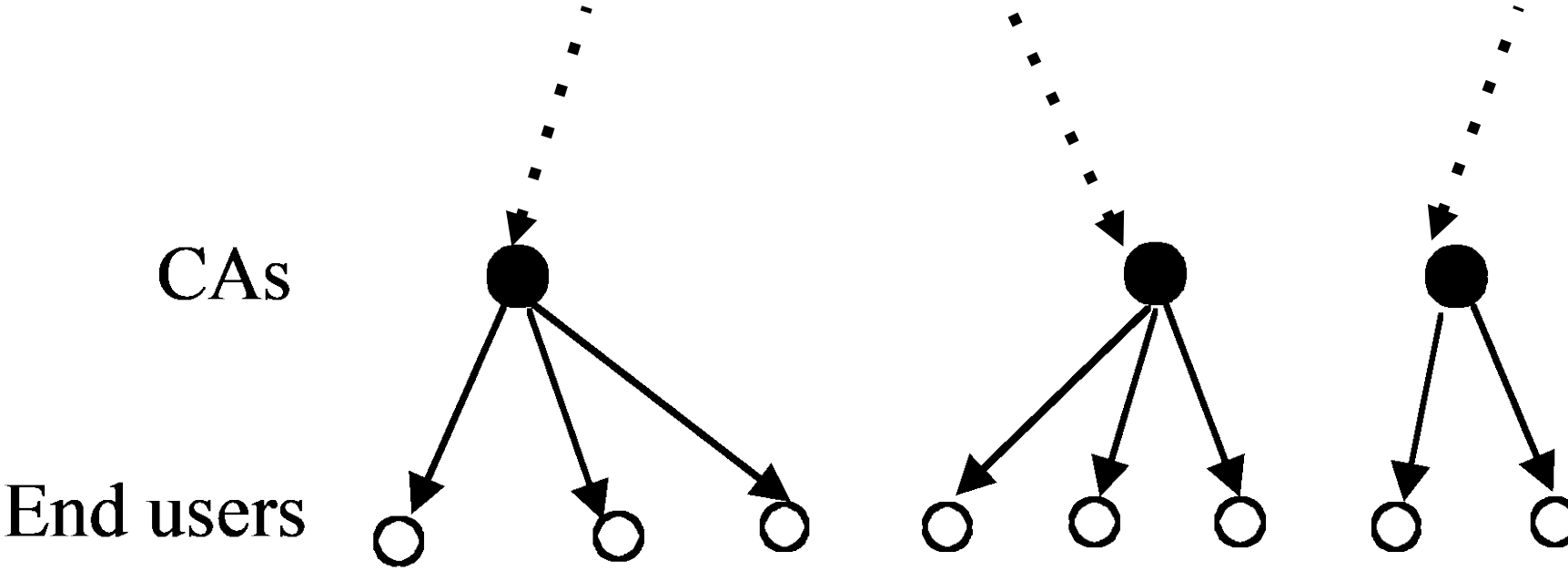
Hierarchical PKI Example



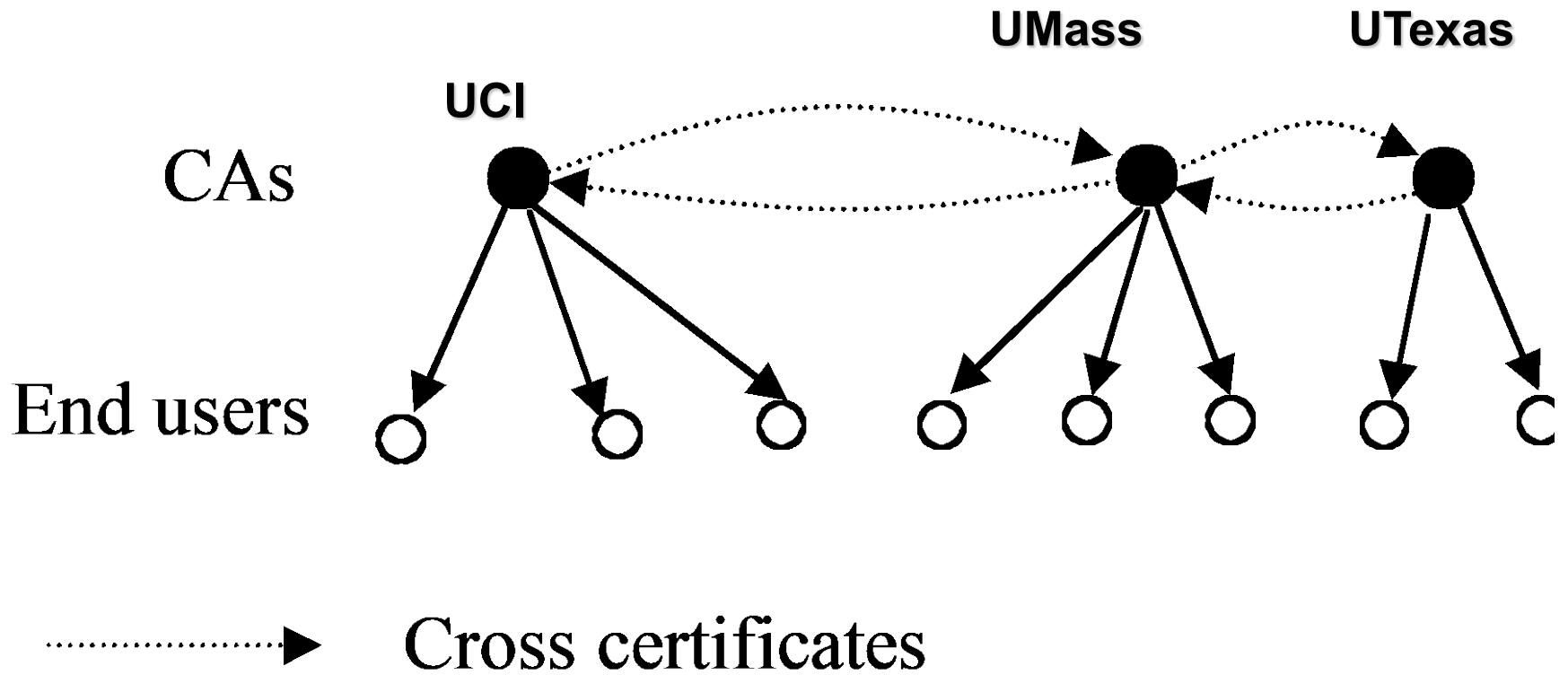
Hierarchical PKI Example



Cross Certificate Based PKI Example

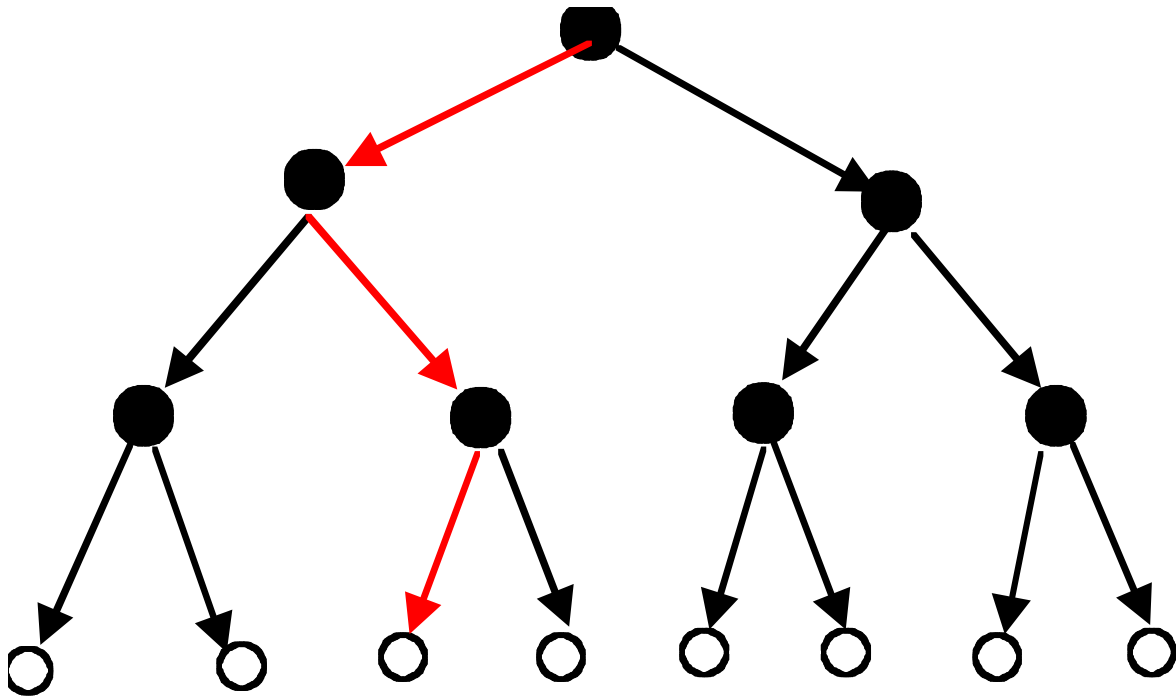


Cross Certificate Based PKI Example

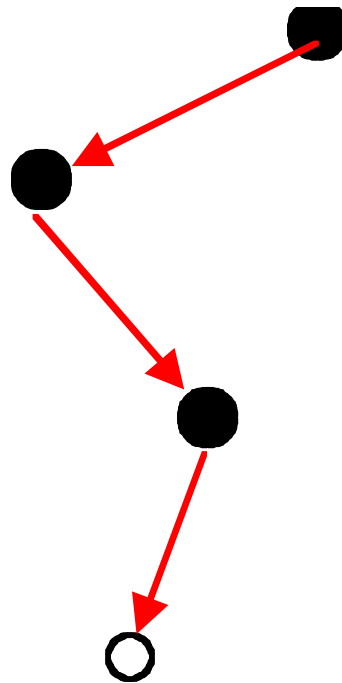


Certificate Paths

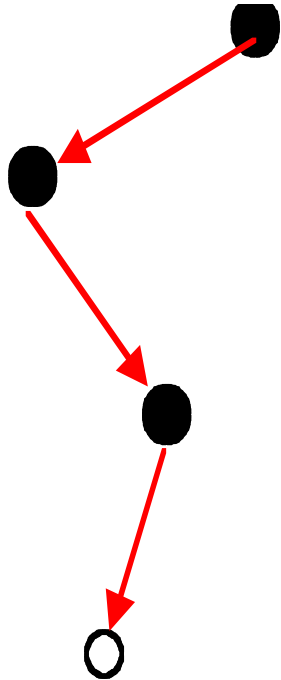
Derived from PKI



Certificate Paths



Certificate Paths



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

X.509 Standard

- X.509v3 is the current version
 - ITU standard
 - ISO 9495-2 is the equivalent ISO standard
- Defines certificate format, not PKI
- Supports both hierarchical model and cross certificates
- **End users cannot be CAs**

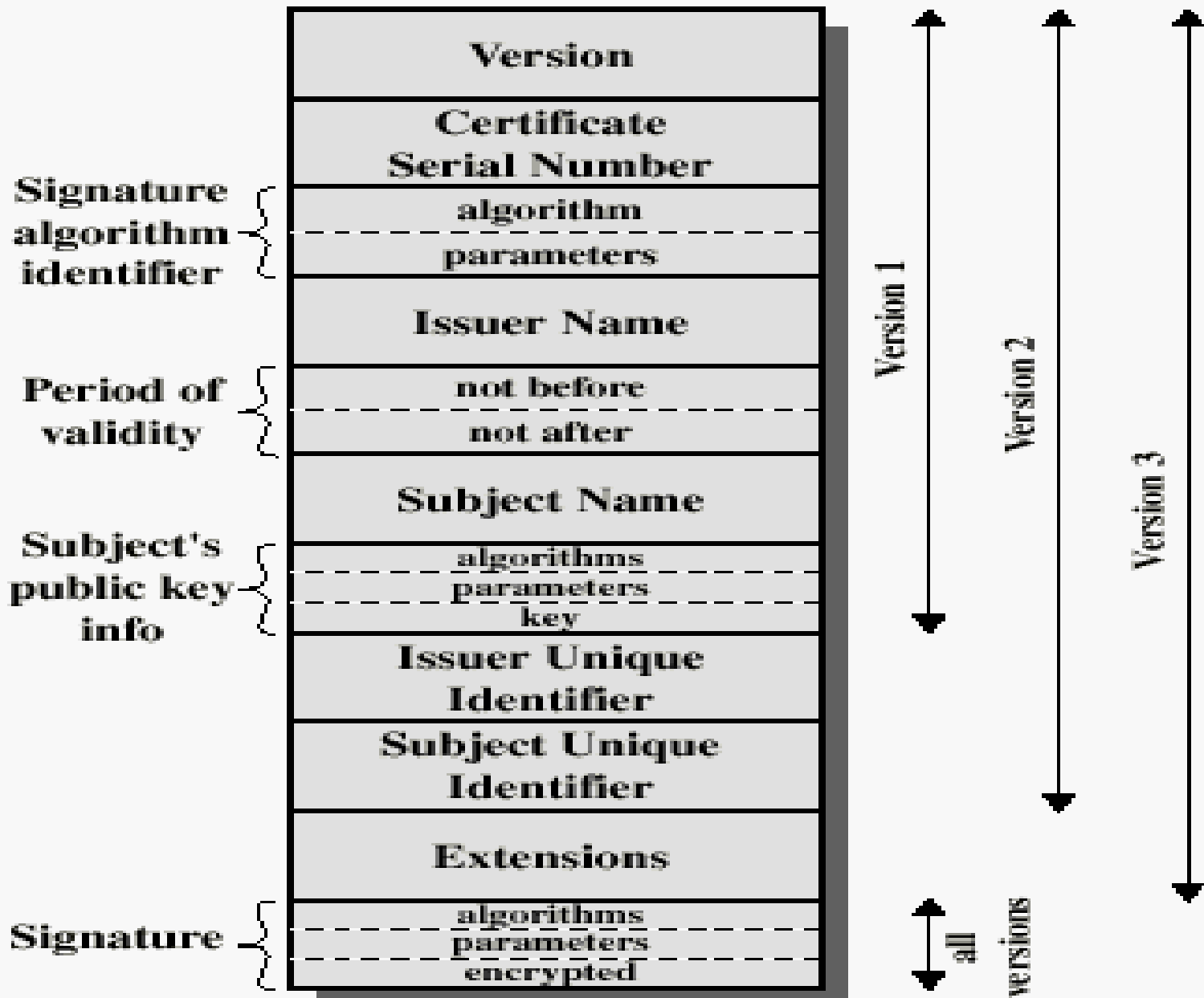
X.509 Service

- Assumes a distributed set of servers maintaining a database about certificates
- Used in S/MIME, PEM, IPSec, SSL/TLS, SSH
- RSA, DSA, SHA, MD5 are most commonly used algorithms

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

X.509 Certificate Format



(a) X.509 Certificate

A Sample X.509v3 Certificate

Certificate:

Data:

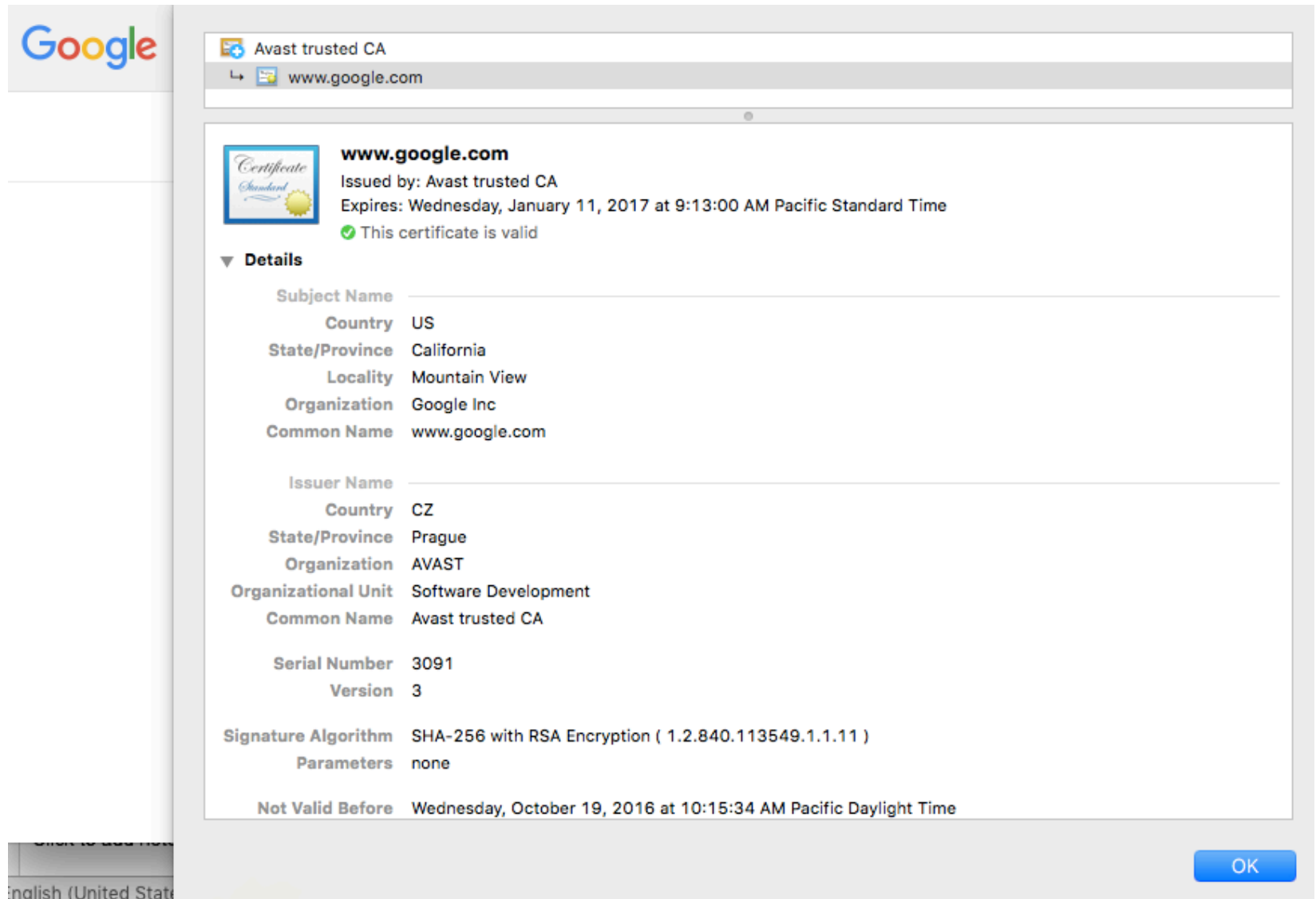
Version: 3 (0x2)
Serial Number:
10:e6:fc:62:b7:41:8a:d5:00:5e:45:b6
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256 - G2
Validity
Not Before: Nov 21 08:00:00 2016 GMT
Not After : Nov 22 07:59:59 2017 GMT
Subject: C=US, ST=California, 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/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:

...

A Sample Certificates in Practice (1/3)



The screenshot shows a web browser window with the Google logo on the left. The address bar displays "Avast trusted CA" and "www.google.com". The main content area shows a certificate for "www.google.com" issued by "Avast trusted CA". The certificate is valid and expires on Wednesday, January 11, 2017 at 9:13:00 AM Pacific Standard Time. A "Details" section is expanded, showing the following information:

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	www.google.com
Issuer Name	
Country	CZ
State/Province	Prague
Organization	AVAST
Organizational Unit	Software Development
Common Name	Avast trusted CA
Serial Number	3091
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters	none
Not Valid Before	Wednesday, October 19, 2016 at 10:15:34 AM Pacific Daylight Time

An "OK" button is visible in the bottom right corner of the certificate display area.

A Sample Certificates in Practice (2/3)

Google

Avast trusted CA
www.google.com

Public Key Info

Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : D7 D3 86 4F 23 D4 E6 E4 ...
Exponent	65537
Key Size	2048 bits
Key Usage	Any
Signature	256 bytes : 97 6B 72 86 AD 24 65 AD ...

Extension Subject Key Identifier (2.5.29.14)

Critical	NO
Key ID	84 61 D1 1A 2F B1 EF 8E 4F F4 6F F0 8D 26 FC 91 58 77 9C A3

Extension Authority Key Identifier (2.5.29.35)

Critical	NO
Key ID	DB D4 F7 BB 15 76 6C 3B 01 A5 23 59 C2 37 26 97 46 5D DC 46

Extension Subject Alternative Name (2.5.29.17)

Critical	NO
DNS Name	www.google.com

Fingerprints

SHA1	30 69 24 F3 14 57 D4 84 73 7F B2 BE B8 F5 92 A2 46 8E 9D 2E
MD5	20 CD 07 D1 A3 F4 96 95 2F 33 43 4D E6 F3 D0 1E

1000000000000000000001

OK

A Sample Certificates in Practice (3/3)

-----BEGIN CERTIFICATE-----

```
MIIDTzCCAvmgAwIBAgIBATANBgkqhkiG9w0BAQQFADBcMSEwHwYDVQQKEzhFdXJvcGVhbiBJQ0UtVEVMIHByb2p1Y3QxIzAhBgNVBAsTG1YzLUNlcnRpZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQHEw1EYXJtc3RhZHQwHhcNOTcwNDAYMTczNTU5WhcN
OTgwNDAYMTczNTU5WjBrMSEwHwYDVQQKEzhFdXJvcGVhbiBJQ0UtVEVMIHByb2p1Y3QxIzAhBgNVBAsTG1YzLUNlcnRpZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQHEw1EYXJtc3RhZHQwDTALBgNVBAMTBFBVTRVIwWTAKBgRVCAEBAgICAANLADBIaKEA
qKhTY0kbk8PDC2yIEVXeFmri+VKg3GklxMi/VeExqM7kqSmFmYoVmt72L+G0UF9e
BHWM9HbcPA453Dq+PqRhiwIDAQABo4IBmDCCAQZQwHwYDVR0jBBgwFoAUFnLy+DqG
nEKINDRmdcPU/NGiETMwHQYDVR0OBBYEFJfc4B8gjSoRmLUx4Sq/ucIYiMrPMA4G
A1UdDwEB/wQEAWIB8DAcBgNVHSABAf8EEjAQMAYGBCoDBAUwBgYECQgHBjBDBgNV
HREEPDA6gRV1c2VyQGRhcm1zdGFkdC5nbWQuZGwGIWh0dHA6Ly93d3cuZGFybnXN0
YWR0LmdtZC5kZS9+dXNlcjCBsQYDVR0SBIGpMIGmgQxnbWRjYUBnbWQuZGwGEWh0
dHA6Ly93d3cuZ21kLmRlghdzYXR1cm4uZGFybnXN0YWR0LmdtZC5kZaRcMSEwHwYD
VQQKEzhFdXJvcGVhbiBJQ0UtVEVMIHByb2p1Y3QxIzAhBgNVBAsTG1YzLUNlcnRp
ZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQHEw1EYXJtc3RhZSHDDE0MS4xMi42
Mi4yNjAMBgNVHRMBAf8EAjAAMB0GA1UdHwQWMBQwEqaQoA6BDGdtZGNhQGdtZC5k
ZTANBgkqhkiG9w0BAQQFAANBAGkM4ben8tj76GnAE803rSEGIk3oxtvxBAu34LPW
DIEDzsNqPsfNJCskkmTCg4MGQlMObwkehJr3b2OblJmD1qQ=
```

-----END CERTIFICATE-----

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)

Certificate Revocation Scenarios

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 in public announcements
 - Published in public 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 "Bob" generated before he realized his key is lost?

Certificate is a capability

- Certificate revocation needs to occur when:
 - certificate holder key compromise/loss
 - CA key compromise
 - end of contract (e.g., certificates for employees)
- Certificate Revocation List (CRL) lists certificates that are not yet naturally expired but revoked
- CRL should be reissued periodically, even there if no new revocation activity! WHY?

Requirements for Revocation

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

Types of Revocation

- **Implicit**

- Each certificate is frequently/periodically re-issued
- Alice has a current valid certificate → Alice is not revoked
- No need to distribute/publish revocation info

- **Explicit**

- Only revoked certificates are periodically announced
- Alice's certificate is not listed among the revoked → Alice is not revoked
- Need to distribute/publish revocation info

Revocation Methods

Explicit:

- CRL - Certificate Revocation List
 - Sources: CRL-DP, indirect CRL, dynamic CRL-DP
 - Delta-CRL, windowed CRL, etc.
 - Certificate Revocation Tree (CRT) and other Authenticated Data Structures
- OCSP – On-line Certificate Status Protocol

Implicit:

- CRS - Certificate Revocation System

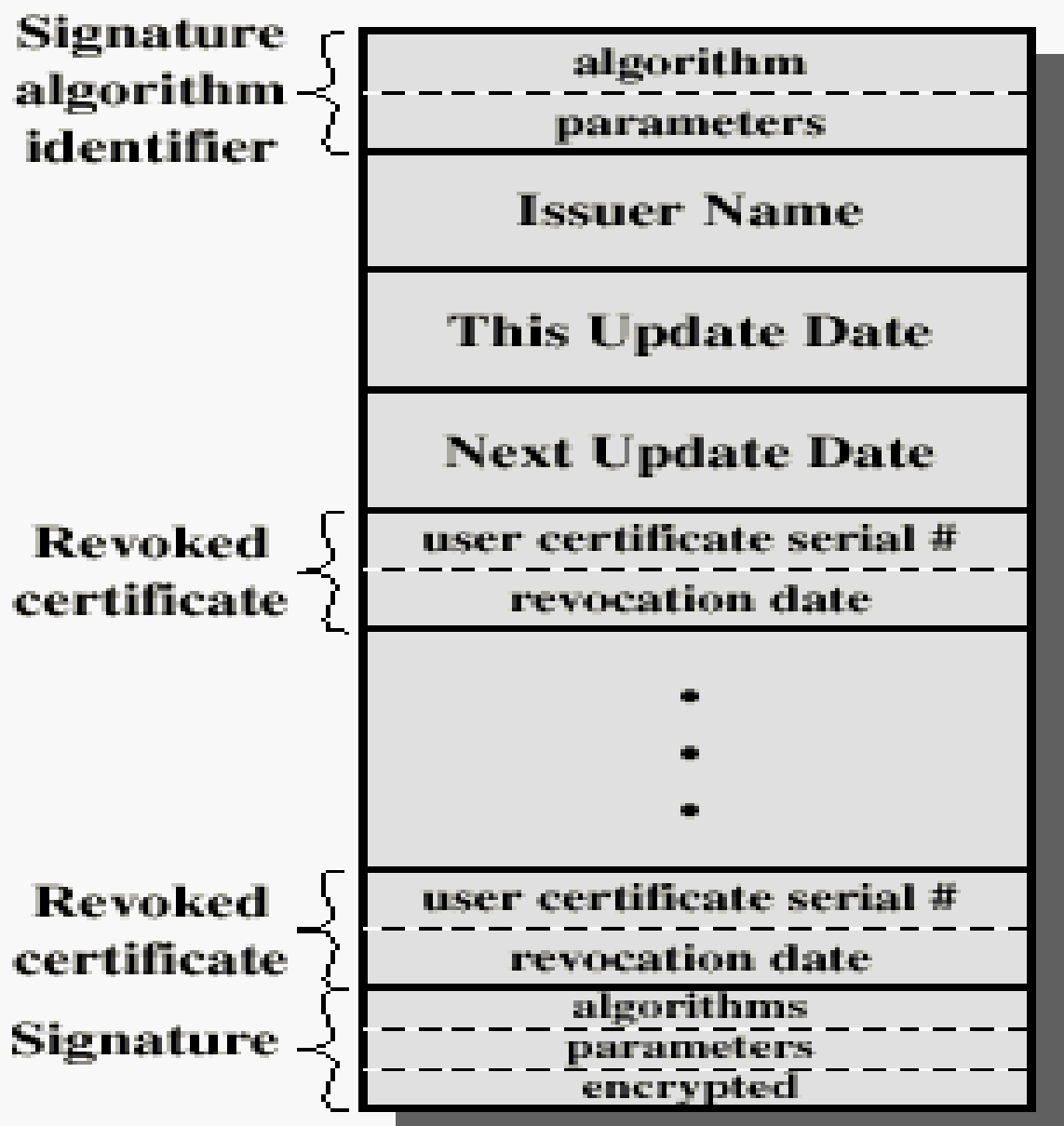
Certificate Revocation List (CRL)

- Off-line mechanism
- CRL = list of revoked certificates (e.g., SNs) signed by a revocation authority (RA)
- RA not always CA that issued the revoked PKC
- Periodically issued: daily, weekly, monthly, etc.

Pros & Cons of CRLs

- Pros
 - Simple
 - Does not need secure channels for CRL distribution
- Cons
 - Timeliness: “window of vulnerability”
 - CRLs grow and can become huge
 - How to distribute CRLs reliably?

X.509 CRL Format

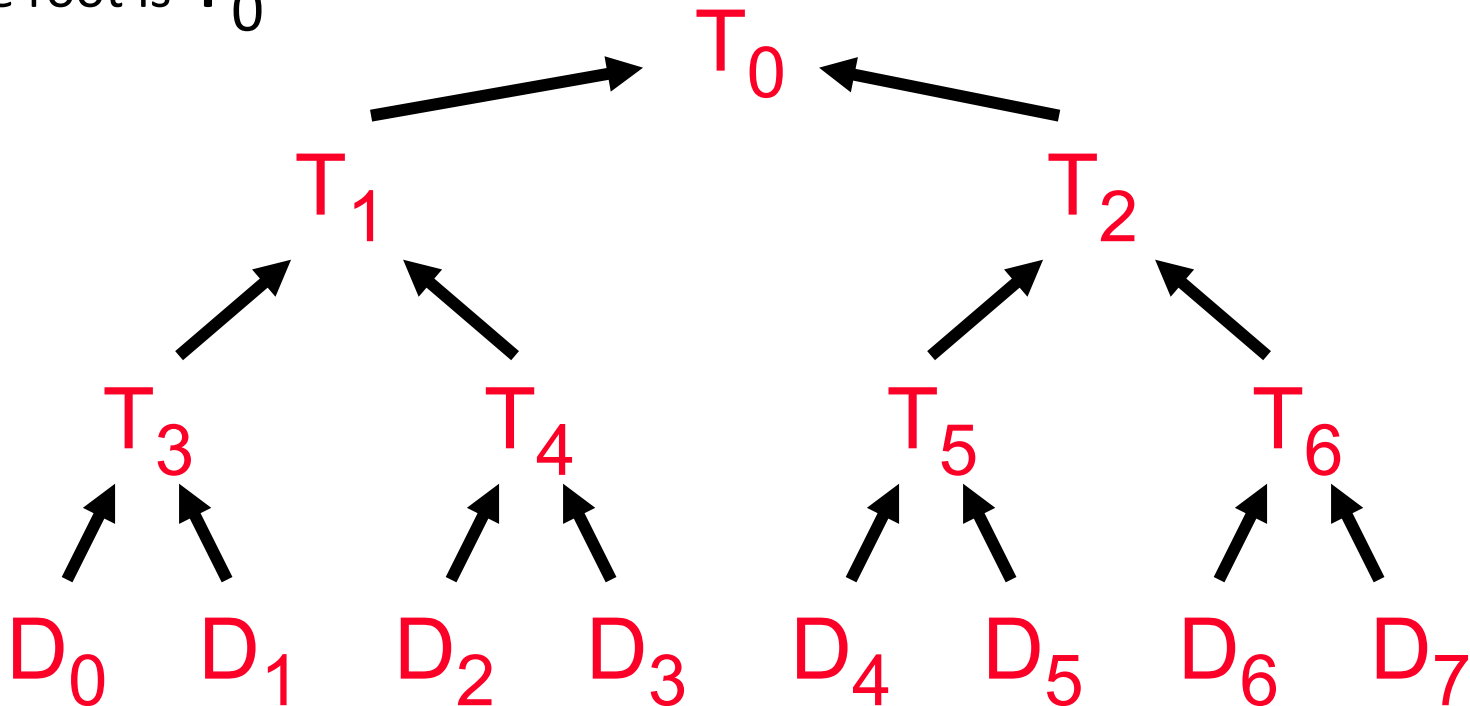


Certificate Revocation Tree (CRT)

- Proposed by in 1998 by P. Kocher
- Based on so-called hash trees
 - Hash trees first proposed by R. Merkle in another context in 1979 (one-time signatures)

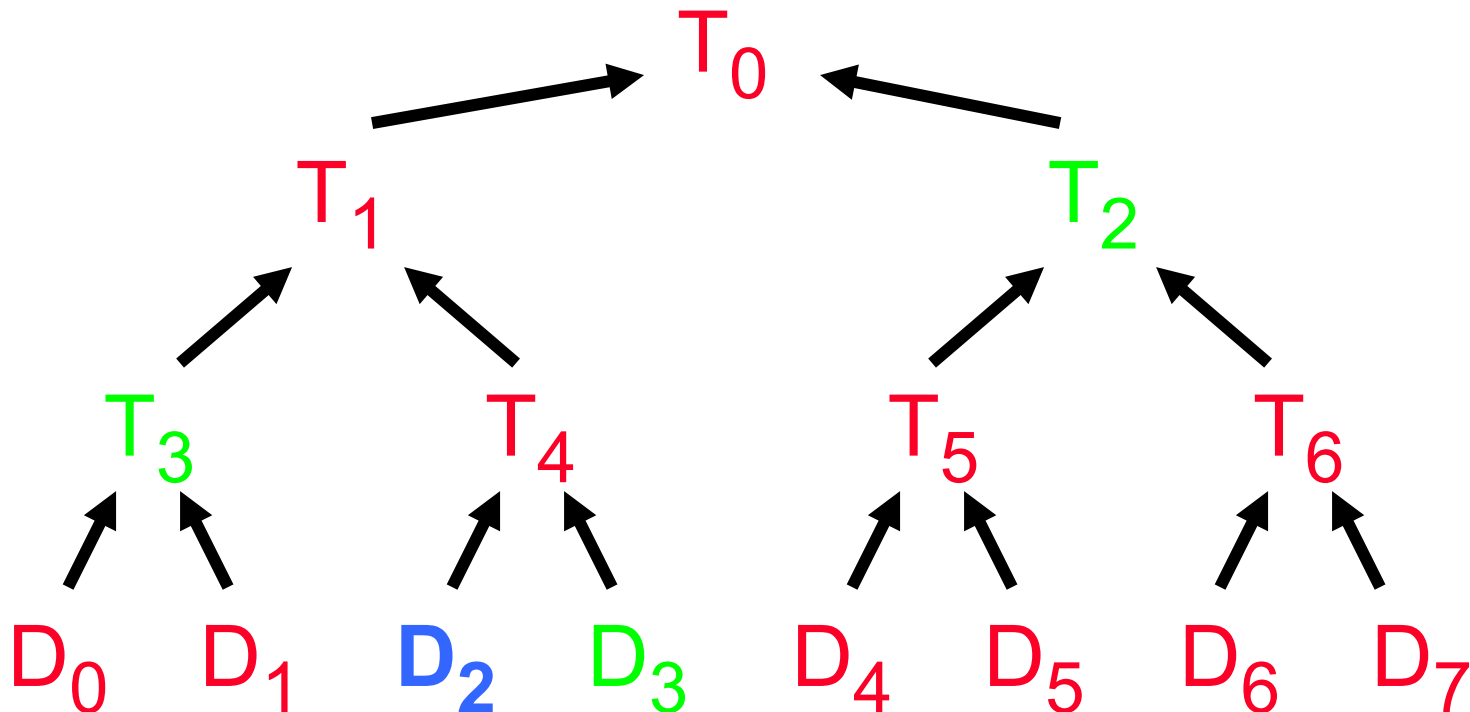
Merkle Hash Tree Example

- Need to authenticate a sequence of values: D_0, D_1, \dots, D_N
- Construct binary tree over data values
- Arrows represent hashing, e.g., $T_4 = H(D_2, D_3)$
- The root is T_0



Merkle Hash Trees: II

- Verifier knows T_0
- How can verifier authenticate tree leaf D_i ?
- Solution: re-compute T_0 using D_i
- Example: to authenticate D_2 , send D_2 and co-path=[D_3, T_3, T_2]
- Verify $T_0 = H(H(T_3 || H(D_2 || D_3)) || T_2)$

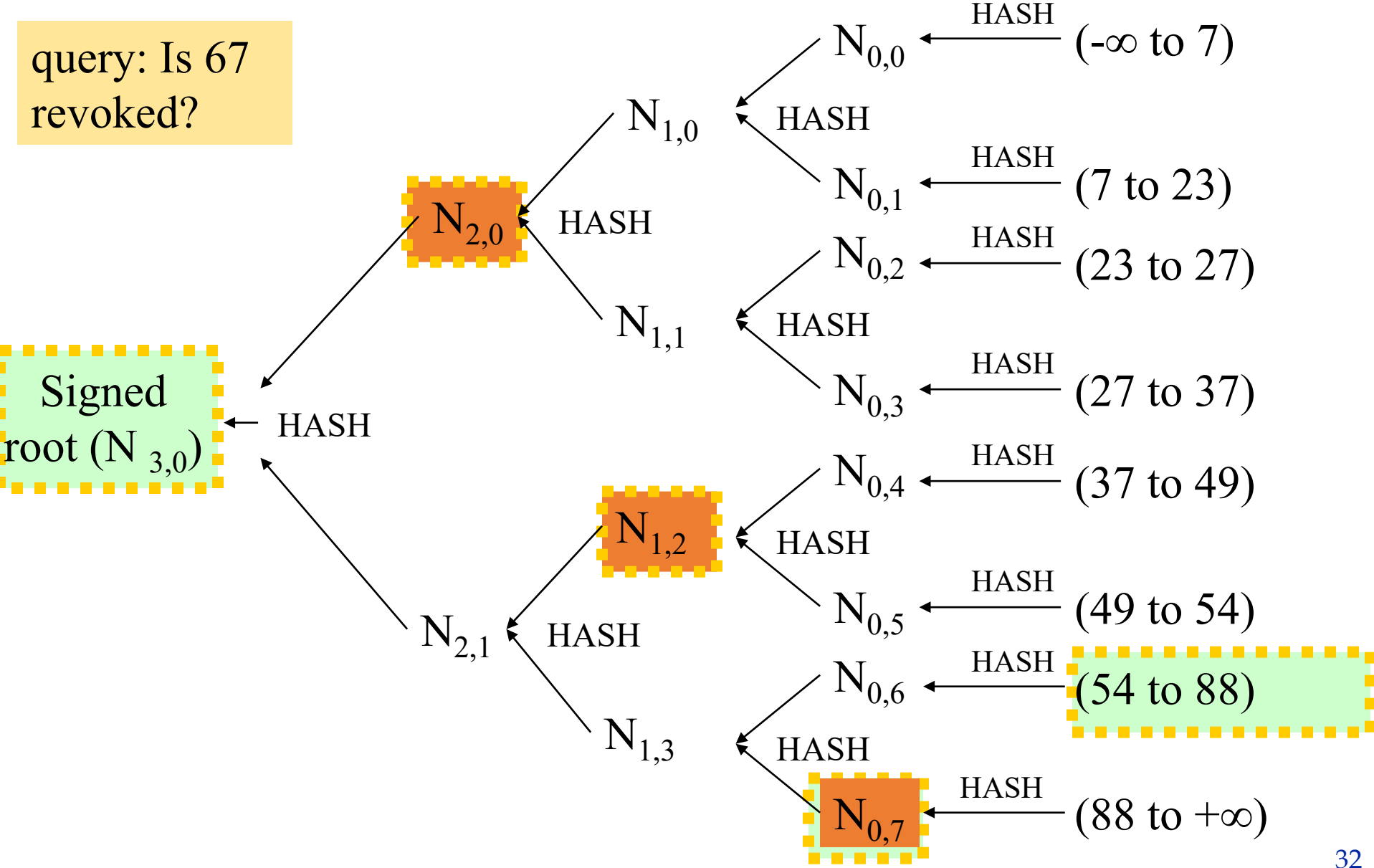


CRT Contd.

- Express ranges of SN of PKC's as tree leaf labels:
 - E.g., (5—12) means: 5 and 12 are revoked, those larger than 5 and less than 12 are okay
 - Place the hash of the range in the leaf
- Response includes the corresponding tree leaf, the necessary hash values along the path to the root, the signed root
- The CA periodically updates the structure and distributes to untrusted servers called Confirmation Issuers

Example of CRT: each leaf = range of valid certificates

query: Is 67 revoked?



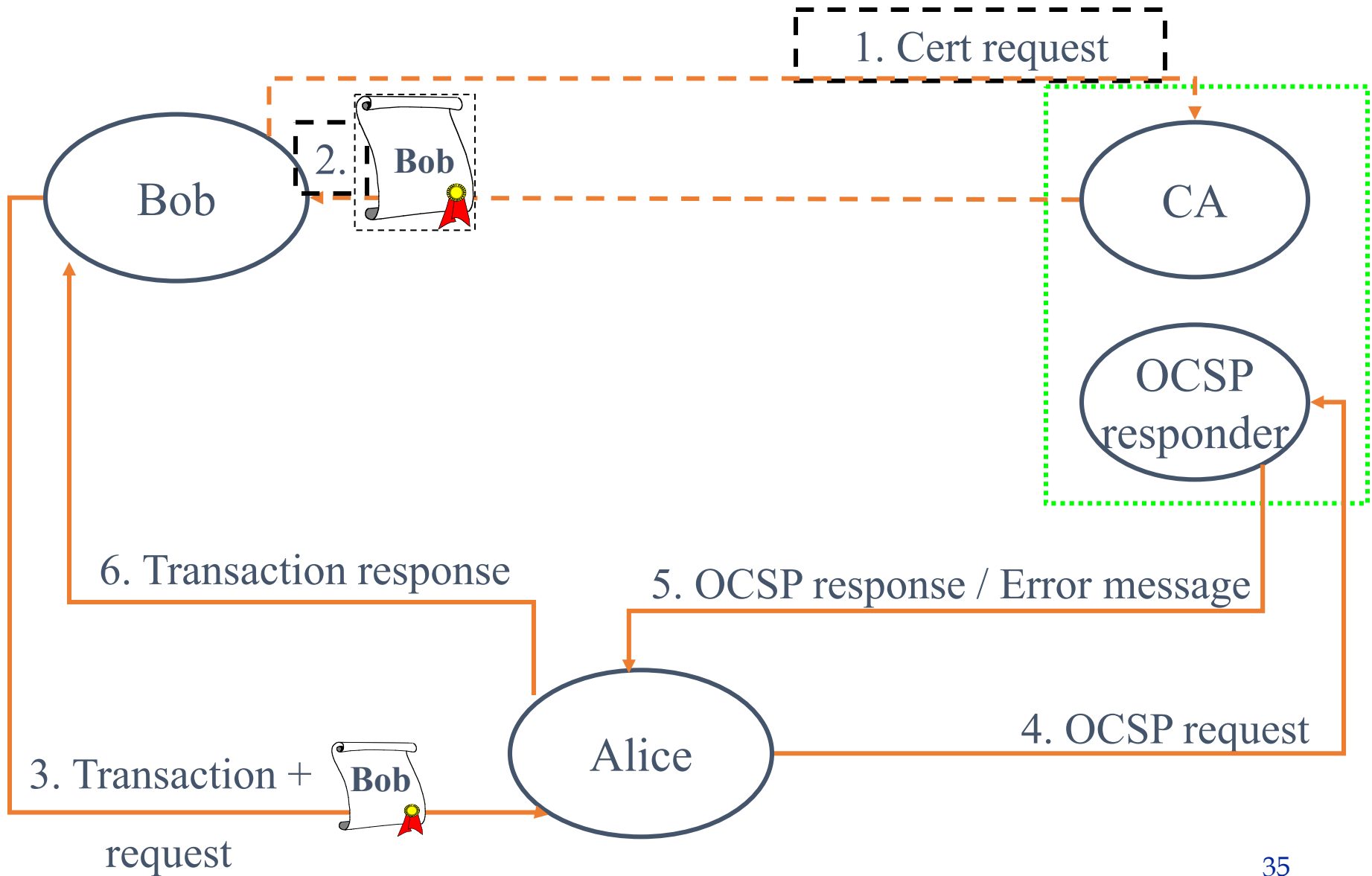
Characteristics of CRT

- Each response (leaf + co-path) represents a proof
- Length of proof is: $O(\log n)$
 - Much shorter than CRL which is $O(n)$
 - Where n is # of revoked certificates
- Only one “real” signature for the whole tree – over the root

Explicit Revocation: OCSP

- OCSP = On-line Certificate Status Protocol (RFC 2560) - June 1999
- Used in place of or, as a supplement to, checking CRLs
- Conveys instantaneous status of a PKC
- Especially suitable for sensitive, volatile settings, e.g., stock trades, electronic funds transfer, military

OCSP Players



OCSP Definitive Response

- All definitive responses have to be signed:
 - either by issuing CA
 - or by a Trusted Responder (OCSP client trusts the TR's PKC)
 - or by a CA Authorized Responder which has a special PKC (issued by the CA) saying that it can issue OCSP responses on CA's behalf

Responses for Each Certificate

- Response format:
 - target PKC SN
 - PKC status:
 - good - positive answer
 - revoked - permanently/temporarily (on-hold)
 - unknown - responder doesn't know about the certificate being requested
 - response validity interval
 - optional extensions

Special Timing Fields

- A response contain three timestamps:
 - `thisUpdate` - time at which the status being indicated is known to be correct
 - `nextUpdate` - time at or before which newer information will be available
 - `producedAt` - time at which the OCSP responder signed this response. Useful for response **pre-production**

Security Considerations

- On-line method
- DoS vulnerability
 - flood of queries + generating signatures!
 - unsigned responses → false responses
 - pre-computing responses offers some protection against DoS, but...
- Pre-computing responses allows replay attacks (since no nonce included)
 - but OCSP signing key can be kept off-line

Open Questions

- Consistency between CRL and OCSP responses
 - It is possible to have a certificate with two different statuses.
- If OCSP is more timely and provides the same information as CRLs, do we still need CRLs?
- Which method should come first - OCSP or to CRL?