Lecture 14

Public Key Certification and Revocation
CertificationTree / Hierarchy

Logical tree of CA-s

root

CA1

CA2

CA3

CA4

PK_{root}

[PK_{CA1}]SK_{root}

[PK_{CA2}]SK_{CA1}

[PK_{CA3}]SK_{root}

[PK_{CA4}]SK_{CA3}
Hierarchical Public Key Infrastructure (PKI)

Example

CAs

End users

UCI

UCSB

UCSD

UCR
Hierarchical PKI Example

State Govt.

Root CA
Upper level CAs
CAs
End users
Cross Certificate Based PKI Example

CAs

End users
Cross Certificate Based PKI Example

CAs

End users

UC System, UMass, UTexas

Cross certificates
Hybrid PKI Example

Note that no cross arrows down or up!
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
- Identity and attribute certificates
- 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 Sample X.509 Certificate

Certificate:
Data:
Version: 3 (0x2)
Serial Number: 28 (0x1c)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=US, O=Globus, CN=Globus Certification Authority
Validity
   Not Before: Apr 22 19:21:50 2010 GMT
   Not After : Apr 22 19:21:50 2020 GMT
Subject: C=US, O=Globus, O=University of Southern California, ou=ISI, CN=bonair.isi.edu
Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   RSA Public Key: (1024 bit)
      Modulus (1024 bit):
         00:bf:4c:9b:ae:51:e5:ad:ac:54:4f:12:52:3a:69:
         <snip>
         b4:e1:54:e7:87:57:b7:d0:61
      Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
   <snip>
A Sample Certificates in Practice (1/3)
### A Sample Certificates in Practice (2/3)

#### 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

- **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

- **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

---

[Image of a certificate details window]
A Sample Certificates in Practice (3/3)

-----BEGIN CERTIFICATE-----
MIIDTzCCAvmgAwIBAgIBATANBgkqhkiG9w0BAQQFADBcMSEwHwYDVQQKEkhFdXJv
cGVhbiBJQ0UtVEVMIHByb2p1Y3QxIzAhBqNVBAstTGlYzLUN1cnRpZmljYXRpb24g
QXV0aG9yaXR5MRIwEAYDVQQHEw1EYXJtc3RhZHqwhHcNOCtwNDAyMTc5WhcN
OTgwNDAYMTc5NTU5WjBrMSEwHwYDVQQKEkhFdXJvCGVhbiBJQ0UtVEVMIHByb2p1
Y3QxIzAhBqNVBAstTGlYzLUN1cnRpZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQQH
Ew1EYXJtc3RhZHqwhDTAIBgNVAMBFVTRVIFWTAKBgRVCAEBAgICAANLADBIAKEar
qKhTY0bk8PDC2yIEVXefmri+VKg3Gk1xMi/HeExqM7kqSmFmYoVmt72L+GQ9F9e
BHw9HbcPA453Dq+PqRhiwIDAQABo4ImDCCAZQwHwYDVRR0jBBgwFoAUfnLy+DqG
nEKINDRmdcPU/NGiETMwHQYDVRR0BBYEFJfc4B8gjSoRmLUX4Sg/ucIYiMrPMA4G
A1UdDwEB/wQEAwIBBDAcBgnVHSABAf8EEjAQMAAYgBCoDBAUwBglECQgHkJDBBkB
HREEPA6gRV1c2VyQGRhcm1zdGFDk5nbWQuZGWIWh0dHA6Ly93d3cuZGFybXN0
YWR0LmdtZC5kZS9+dXN1cJCBsQYDVRR0SBIGmQXnbWRjYUBnWQuZGWEWh0
YWR0LmdtZC5kZaRcMSEwHwYVQQKEkhFdXJvCGVhbiBJQ0UtVEVMIHByb2p1Y3QxIzAhBq
NVBAstTGlYzLUN1cnRpZmljYXRpb24gQXV0aG9yaXR5MRIwEAYDVQHEw1EYXJtc3Rh
ZHSHDDE0MS4xMi42
Mi4yNjAMBgNVHRMBAf8EAEaAAMB0GA1UdHwQMBQwEeAoCA6BDGdtZGNhQGdtZC5k
ZTANBgqkhkIG9w0BAQQFAANBAGkM4ben8tj76GnA803rSEGIK3oxtvxABu34LPW
DIEDzsNqPsfkJCSkkmTCg4MGQ1MoBwkehJr3b2Ob1JmDlqQ=
-----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 Scenario

What if:

• Bob’s CA goes berserk?
• Bob forgets his private key?
• Someone steals Bob’s private key?
• Bob loses his 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 reissued periodically, even if no activity!

• More on revocation later ...
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 periodically (re-issued)
  • Alice has a fresh certificate ➔ Alice not revoked
  • No need to distribute/publish revocation info

• Explicit
  • Only revoked certificates are periodically announced
  • Alice’s certificate not listed among the revoked ➔ Alice not revoked
  • Need to distribute/publish revocation info
Revocation Methods

• CRL - Certificate Revocation List
  • 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

• 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 can be huge
  • How to distribute CRLs reliably?
X.509 CRL Format
PKI and Revocation

• On January 29 and 30, 2001, VeriSign, Inc. issued two certificates for Authenticode Signing to an individual fraudulently claiming to be an employee of Microsoft Corporation.

• Any code signed by these certificates appears to be legitimately signed by Microsoft.

• Users who try to run code signed with these certificates will generally be presented with a warning dialog, but who wouldn't trust a valid certificate issued by VeriSign, and claimed to be for Microsoft?

• Certificates were very soon placed in a CRL, but:
  • code that checks signatures for ActiveX controls, Office Macros, and so on, didn't do any CRL processing.

• According to Microsoft:
  • since the certificates don't include a CRL Distribution Point (DP), it's impossible to find and use the CRL!
Certificate Revocation Tree (CRT)

• Proposed by P. Kocher (1998)

• Based on hash trees
  • Hash trees first proposed by R. Merkle in another context in 1979 (one-time signatures)
  • Improvement to Lamport-Diffie one time signature (OTS) scheme

• Based on the following idea:
  • A wants to sign (in the future) 1 bit of information
  • A gives B the image Y produced as Y=F(X)
  • To sign, A reveals the pre-image: X
  • B checks that: Y=F(X)
Merkle Hash Trees: I

- Authenticate a sequence of data values $D_0, D_1, ..., D_N$
- Construct binary tree over data values
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 $T_0 = \text{H}( \text{H}(T_3 \ || \ \text{H}(D_2 \ || \ D_3)) \ || \ T_2)$
CRT Contd.

• Express ranges of SN of PKC’s as tree leaf labels:
  • E.g., (5--12) means: 5 and 12 are revoked, the others larger than 5 and smaller 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

query: Is 67 revoked?

Signed root ($N_{3,0}$) → HASH

$N_{2,0}$ → HASH

$N_{1,1}$

$N_{1,0}$ → HASH

$N_{0,0}$ → HASH (-∞ to 7)

$N_{0,1}$ → HASH (7 to 23)

$N_{0,2}$ → HASH (23 to 27)

$N_{0,3}$ → HASH (27 to 37)

$N_{0,4}$ → HASH (37 to 49)

$N_{0,5}$ → HASH (49 to 54)

$N_{0,6}$ → HASH (54 to 88)

$N_{0,7}$ → HASH (88 to +∞)
Characteristics of CRT

• Each response 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 tree root (can be done off-line)
Explicit Revocation: OCSP

- OCSP = On-line Certificate Status Protocol (RFC 2560) - June 1999

- In place of or, as a supplement to, checking CRLs

- Obtain instantaneous status of a PKC

- OCSP may be used in sensitive, volatile settings, e.g., stock trades, electronic funds transfer, military
OCSP Players

1. Cert request

2. Bob

3. Transaction + request

4. OCSP request

5. OCSP response / Error message

6. Transaction response
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?
Implicit Revocation: Certificate Revocation System (CRS)

- Proposed by Micali (1996)
- Aims to improve CRL communication costs
- Basic idea: CA periodically refreshes valid certificates
- Uses off-line/on-line signature scheme to reduce update cost
One-Way Hash Chains

• Versatile cryptographic primitive

• Construction:
  1. Pick random $Y_N$ and public hash function $H()$
  2. Compute all values $Y_{N-1}, ..., Y_0$ such that $Y_{i-1} = H(Y_i)$
  3. Secret $\text{ROOT}=Y_N$, public $\text{ANCHOR}=Y_0$

\[
\begin{align*}
Y_0 & \quad \overset{H}{\rightarrow} \quad Y_1 \quad \overset{H}{\rightarrow} \quad Y_2 \quad \overset{H}{\rightarrow} \quad \cdots \quad \overset{H}{\rightarrow} \quad Y_{N-1} \quad \overset{H}{\rightarrow} \quad Y_N
\end{align*}
\]

• Properties:
  • Use in reverse order of construction: $Y_0, Y_1, ..., Y_N$
  • Hard to compute $Y_i$ from $Y_j$ (if $j<i$), easy to compute $Y_j$ from $Y_i$
    • For example: easy to compute $Y_1$ from $Y_2$ since $Y_1=H(Y_2)$
    • But, Infeasible to compute $Y_2$ from $Y_1$
  • Verifier can efficiently authenticate $Y_i$ knowing $Y_i$ ($j<i$): by verifying whether $Y_j = H^{i-j}(Y_i) = H(H(...H(Y_i)...))$
  • This method is robust to missing values
CRS: Creation of a Certificate

• Two new parameters in PKC: $Y_0$ and $N$

$$Y_0 = H^{\text{MAX}}(Y_{\text{MAX}})$$

$$N_0 = H(N_1)$$

• $[Y_0, N_0]$ -- per-PKC secrets stored by CA

• $H()$ -- public one-way function, e.g., SHA-2
CRS Example:
Certificate issued for a year, refreshed daily

- If Alice's certificate is valid:
  - $\text{UPD}_i = Y_i$ and
  - $Y_0 = H(Y_i) \iff \text{verifier can easily check this}$
  - Also, note that: $Y_i = H^{\text{MAX}-i}(Y_{\text{MAX}})$

- If her certificate is revoked, $\text{UPD}_i = N_1$
- $Y_0$ and $N_0$ are distinct for each certificate

NOTE: $i=0$ at issuance date