X.509 and TLS

X.509 is an ITU standard (but also RFC 5280).

- TLS servers (and sometimes clients) are identified by public key
- Public keys are *certified* by certificate authorities
- X.509 certificates are the format used for certificates
- Any certificate authority can certify keys for any domain

---

TLS is not the only major protocol using X.509!
There is also S/MIME for e-mail!
X.509 overview (reminder)
What is a certificate?

- A public-key certificate is a digitally signed statement that binds the identity of the entity to a public key.
- If $A$ trusts $B$, and knows $B$’s public key, then $A$ can learn $C$’s public key if $B$ issues a public key certification of $C$. 
Contents of X.509 certificates

- X.509 version
- CA serial number
- A digital signature algorithm identifier
- The identity of the signer
- Validity period
- The identity of the subject (common name, org. unit, org., state, country)
- The public key of the subject
- Optional: URL to revocation center (OCSP!)
- Auxiliary information (identity address, alternative names)
- The digital signature
Certificate Authorities

- Entities that *claim* to be trustworthy to verify identities and issuing public key certificates ("let’s encrypt")
- CAs can be organized into a directed graph
- X.509: Tree depth can be limited for a subtree
- X.509: Certificates of CAs signing intermediate-level CAs have the special "CA" bit set
Self-signed certificates

- Signer is self
- Allowed by TLS
- Used to sign CA tree roots
X.509 CA challenges

- Must trust a CA
  - Which one?
  - What is it trusted to do?
- Certificate bindings must be correct
  - Which John Smith is this?
  - Who authorizes attributes in a certificate?
  - How long are these values valid?
  - What process is used to verify the key holder?
CA: creates a self-signed certificate

# create certificate:
$ openssl req -x509 -out cert.pem -outform PEM -days 3650
# private key will now be in privkey.pem
# convert to certificate request:
$ openssl x509 -x509toreq -in cert.pem -out req.pem \
   -signkey privkey.pem
# generate config
$ cp /usr/lib/ssl/openssl.cnf .
# self-sign using:
$ openssl x509 -req -in req.pem -extfile openssl.cnf \
   -extensions v3_ca -signkey privkey.pem -out selfcert.pem
# view using:
$ openssl x509 -in cacert.pem -text -noout

PEM encoding is Base64 of DER bytestream with “begin certificate” and “end certificate” markers.
Client: creates a certificate request

# create private key using:
$ openssl genpkey -algorithm RSA -out key.pem  \
   -aes-128-cbc -pkeyopt rsa_keygen_bits:2048

# create CSR using:
$ openssl req -new -key key.pem -keyform PEM  \
   -out req.pem -outform PEM
CA: signs certificate request

# Prepare CA directory structure
$ wget https://grothoff.org/christian/teaching/ca.conf
$ mkdir dir certdir
$ touch dir/index.txt dir/index.txt.attr
$ echo 1 > dir/serial.txt
# sign CSR using:
$ openssl ca -in req.pem -out cert.pem -config ca.conf
X.509v3 subjectAltNames

X.509v3 certificates can specify many subjectAltNames:

- IP:192.168.2.0
- DNS:www.example.com
- email:user@example.com

Emails must be subjectAltNames and should not be used for the subject distinguished name (DN)!
X.509v3 crlDistributionPoints

version (v1 or v2)
signature
issuer
lastUpdate
nextUpdate
revokedCertificates
  SerialNumber
  RevocationDate
crlEntryExtensions(v2)
  SerialNumber
...
signatureAlgorithm
signature
Online Certificate Status Protocol (OCSP)

OCSP Request:
status of Kool CA #2 ?
optionally signed by Bodo

OCSP Reply:
Kool CA #2 good
signed by OCSP Server

frequent status updates e.g. via CRL

locally stored

Authentication

Antje

Kool CA

#0

Bodo

Kool CA

#2

Bodo

Kool CA

#3

OCSP

#0

OCSP

#0

OCSP

#0

Kool CA

#2

Kool CA

#0
basicConstraints

- CA:TRUE; critical
- CA:TRUE; pathLenConstraint = 0
- CA:FALSE
keyUsage

CA:
» certificateSign
» crlSign

Leaf:
» digitalSignature
» nonRepudiation
» keyEncipherment
» dataEncipherment
» keyAgreement
Extended Key Usage (EKU)

- serverAuth
- clientAuth
- codeSigning
- emailProtection
- timeStamping
- ocspSigning
Legal Aspects

For what is a CA liable?

- Certificate policies (CP) define rights, duties and obligations of each party in a PKI
- These documents usually have a legal effect
- The CP should be publicly exposed by CAs on their Web site and include:
  - Registration procedures
  - Revocation procedures
  - Liability issues
Acknowledgements

- Partially based on materials and inspiration taken from talks by Andreas Steffen (ITA)