Public key infrastructure (PKI)
Public key certificates (X.509)
Public key infrastructure (PKI)

Public key certificate binds a public key to an identity

- Main actors:
  - User (subject, subscriber, end-entity)
  - Certificate authority (trusted third party, issuer)
  - Verifier (relying party)

- The passport analogy
X.509 certificate

Subject: “John Smith”
Issuer: Verisign, Inc. Root CA
Public key: BC F7 C6 74 F5 32 D0 34 ...
Serial #: 11:21:56:2D:2E
Valid: from 2020.01.01 15:00 to 2022.01.01 15:00
Other data: ...

Signed: CA’s Signature
PKI use cases

- Server authentication
- Digital signatures (eIDAS)
- Code signing
- E-mail (S/MIME)
Certificate Authority (CA)

Certificate Authority – **Trusted** Third Party. Where does the trust come from?

- Software vendors decide on our behalf (Mozilla, Google)
- EU Regulation 910/2014 (eIDAS)

How to get our CA trusted:

- Compliance audit (WebTrust, ETSI TS)
  - Ernst & Young or KPMG (15k EUR/year)
- Liability insurance (required by eIDAS)
  - Insurance industry reluctant (3k EUR/year)
- Use of Hardware Security Module (HSM)

Certificate hierarchy

- Root CA – self-signed certificate (trust anchor)
- CA can delegate trust to subordinate/intermediate CAs
- Useful for risk limitation
X.509 certificate

Certificate ::= SEQUENCE {
  tbsCertificate TBSCertificate,
  signatureAlgorithm AlgorithmIdentifier,
  signatureValue BIT STRING }

TBSCertificate ::= SEQUENCE {
  version [0] EXPLICIT Version DEFAULT v1(0),
  serialNumber INTEGER,
  signature AlgorithmIdentifier,
  issuer Name,
  validity Validity,
  subject Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  extensions [3] EXPLICIT Extensions OPTIONAL -- v3(2) only }

Validity ::= SEQUENCE {
  notBefore UTCTime,
  notAfter UTCTime }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
  extnID OBJECT IDENTIFIER,
  critical BOOLEAN DEFAULT FALSE,
  extnValue OCTET STRING }

X.509 certificate

- tbsCertificate – DER structure to be signed by CA
- version – X.509v1 or X.509v3 used
  - X.509 v3 introduces certificate extensions
- serialNumber – unique for every certificate issued by CA
- signature – AlgorithmIdentifier from outer Certificate sequence
- issuer – identity of CA who signed the certificate
- validity – period in which certificate should be assumed valid
- subject – identity of a subject whose public key in the certificate
- subjectPublicKeyInfo – subject’s public key
- extensions – optional extensions providing more information
Distinguished Name (DN) in X.509 Certificate

The issuer and subject field is defined as the X.501 type Name:

Name ::= RDNSequence
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
  type OBJECT IDENTIFIER,
  value ANY -- DEFINED BY type
}

- Yet another notation for unique identifiers
- Used in LDAP and related protocols
- Example: CN=John Doe, OU=Helpdesk, O=Burgers Inc., C=US
Distinguished Name (DN) in X.509 Certificate

2 74: SEQUENCE {
4 11:   SET {
6 9:     SEQUENCE {
8 3:       OBJECT IDENTIFIER countryName (2 5 4 6)
13 2:       PrintableString 'US'
:     }
:   }
17 18:   SET {
19 16:     SEQUENCE {
21 3:       OBJECT IDENTIFIER organizationName (2 5 4 10)
26 9:       UTF8String 'Burgers Inc.'
:     }
:   }
37 20:   SET {
39 18:     SEQUENCE {
41 3:       OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
46 11:       UTF8String 'Helpdesk'
:     }
:   }
59 17:   SET {
61 15:     SEQUENCE {
63 3:       OBJECT IDENTIFIER commonName (2 5 4 3)
68 8:       UTF8String 'John Doe'
:     }
:   }
: }

(2 5 4 4) : surname (SN)
(2 5 4 42) : givenName (GN)
(2 5 4 5) : serialNumber
(2 5 4 7) : localityName (L)
(2 5 4 8) : stateOrProvinceName (ST)
(1 2 840 113549 1 9 1) : emailAddress
Certificate extensions (X.509v3 only)

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
               extnID OBJECT IDENTIFIER,
               critical BOOLEAN DEFAULT FALSE,
               extnValue OCTET STRING }

• Every extension has an OID
• RFC 5280 defines several standard extensions

"Each extension in a certificate is designated as either critical or non-critical. A certificate-using system MUST reject the certificate if it encounters a critical extension it does not recognize or a critical extension that contains information that it cannot process. A non-critical extension MAY be ignored if it is not recognized, but MUST be processed if it is recognized."

Certificate extensions (X.509v3 only)

• Key usage – limits the purpose of the key contained in the certificate

\[
\text{KeyUsage ::= BIT STRING }
\begin{array}{l}
\text{digitalSignature } (0), \\
\text{nonRepudiation } (1), \text{ -- contentCommitment } \\
\text{keyEncipherment } (2), \\
\text{dataEncipherment } (3), \\
\text{keyAgreement } (4), \\
\text{keyCertSign } (5), \\
\text{cRLSign } (6), \\
\text{encipherOnly } (7), \\
\text{decipherOnly } (8) \\
\end{array}
\]

• If extension is not present the key may be used for all purposes

• Extended key usage – indicates a more specific purpose of the key

\[
\text{ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId}
\]

\[
\text{KeyPurposeId ::= OBJECT IDENTIFIER}
\]

\[
\text{id-kp-serverAuth OBJECT IDENTIFIER ::= } \{ 1 3 6 1 5 5 7 3 1 \} \\
\text{id-kp-clientAuth OBJECT IDENTIFIER ::= } \{ 1 3 6 1 5 5 7 3 2 \} \\
\text{id-kp-codeSigning OBJECT IDENTIFIER ::= } \{ 1 3 6 1 5 5 7 3 3 \} \\
\text{id-kp-emailProtection OBJECT IDENTIFIER ::= } \{ 1 3 6 1 5 5 7 3 4 \} \\
\]

• Usage must be consistent with the key usage extension
Certificate extensions (X.509v3 only)

- Basic constraints – identifies whether subject is CA
  - For CA certificate identifies maximum subordinate CAs it may have
    
    id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
    BasicConstraints ::= SEQUENCE {
        cA BOOLEAN DEFAULT FALSE,
        pathLenConstraint INTEGER (0..MAX) OPTIONAL }

    - If cA is TRUE, the key usage extension must be absent or must have keyCertSign bit set

- Certificate policies – contains pointer to policy information
  - URL to certificate practice statement (CPS)
  - OID of the CPS document version
  - Explicit notice text
Certificate extensions (X.509v3 only)

- **Subject alternative name**
  - Used to alternatively identify a subject
  - Can include email, DNS name, IP addresses, URI, etc.
  - Latest standards promote use of this extension

- **Authority key identifier and subject key identifier**
  - Uniquely identifies subject and issuer

- **CRL distribution points**
  - Includes URI where CRL is available (HTTP or LDAP)

- **Authority information access**
  - Indicates how to access information about CA services

- **Subject information access**
  - Indicates how to access information about subject

Extensions may include a picture of the subject, attributes, roles, etc.
Use in HTTPS (TLS)

- TLS server certificates – the most popular use case
- What does the browser verify before the connection is considered secure?
  - Certificate signed by a trusted CA
  - Host name in the address bar matches the CN in the certificate
  - Validity date, extensions, etc.
## Server certificate

### Certificate

<table>
<thead>
<tr>
<th>Subject Name</th>
<th>DigiCert SHA2 High Assurance Server CA</th>
<th>DigiCert High Assurance EV Root CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>.facebook.com</td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>Facebook, Inc.</td>
<td></td>
</tr>
<tr>
<td>State/Province</td>
<td>California</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>US</td>
<td></td>
</tr>
<tr>
<td>Locality</td>
<td>Menlo Park</td>
<td></td>
</tr>
</tbody>
</table>

### Issuer Name

<table>
<thead>
<tr>
<th>Common Name</th>
<th>DigiCert SHA2 High Assurance Server CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Digicert Inc</td>
</tr>
<tr>
<td>DNS Name</td>
<td><a href="http://www.digicert.com">www.digicert.com</a></td>
</tr>
<tr>
<td>Country</td>
<td>US</td>
</tr>
</tbody>
</table>

### Validity

- **Not Before**: Tue, 12 Jul 2022 00:00:00 GMT
- **Not After**: Mon, 10 Oct 2022 23:59:59 GMT

### Subject Alt Names

- **DNS Name**: .facebook.com
- **DNS Name**: .facebook.net
- **DNS Name**: .fbcdn.net
- **DNS Name**: .pbbx.com
- **DNS Name**: .m.facebook.com
- **DNS Name**: .messenger.com
- **DNS Name**: .xx.fbcdn.net
- **DNS Name**: .yy.fbcdn.net
- **DNS Name**: .zz.fbcdn.net
- **DNS Name**: facebook.com
- **DNS Name**: messenger.com
Server certificate

$ openssl x509 -in facebook-com.pem -text
Version: 3 (0x2)
Serial Number:
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
Validity
   Not Before: Jul 12 00:00:00 2022 GMT
   Not After : Oct 10 23:59:59 2022 GMT
Subject: C = US, ST = California, L = Menlo Park, O = "Facebook, Inc.", CN = *.facebook.com
Subject Public Key Info:
   Public Key Algorithm: id-ecPublicKey
   Public-Key: (256 bit)
   pub:
      39:01:68:cd:66:66:00:0d:c0:ed:ab:5e:59:
      b0:8d:7c:5b:73
   ASN1 OID: prime256v1
   NIST CURVE: P-256
X509v3 extensions:
   X509v3 Basic Constraints: critical
   CA:FALSE
   X509v3 Key Usage: critical
   Digital Signature
   X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
   X509v3 Subject Alternative Name:
      DNS:*.facebook.com, DNS:*.facebook.net, DNS:*.fbcdn.net, DNS:*.fbsbx.com, DNS:*.m.facebook.com
Signature Algorithm: sha256WithRSAEncryption
Identity verification

• Domain Validation (DV): $20/year $0/year
  Checks whether you control the domain

• Organization Validation (OV): $100/year
  Checks whether you operate the organization

• Extended Validation (EV): $200/year
  Checks whether you operate the organization x2
Domain Validation (DV) vs Organization Validation (OV)
### Trusted CAs

[Image of Certificate Manager]

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC Camerfirma S.A.</td>
<td></td>
</tr>
<tr>
<td>Chambers of Commerce Root - 2008</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Global Chambersign Root - 2008</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>AC Camerfirma SA CIF A82743287</td>
<td></td>
</tr>
<tr>
<td>Camerfirma Chambers of Commerce R...</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Camerfirma Global Chambersign Root</td>
<td>Builtin Object Token</td>
</tr>
</tbody>
</table>

[Link](https://ccadb-public.secure.force.com/mozilla/CACertificatesInFirefoxReport)
Certificate Transparency (CT)

https://crt.sh/
Certificate signing request (CSR)

$ openssl genpkey -algorithm RSA -out priv.pem -pkeyopt rsa_keygen_bits:2048
$ openssl req -new -key priv.pem -out example.com.csr

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]:US
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:example.com
Email Address []:

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:asdasd
An optional company name []:

-----BEGIN CERTIFICATE REQUEST-----
MIICfzCCAWcCAQAwIzELMAkGA1UEBhMCVVMxFDASBgNV
MIIBIjANBgkqhkiG9w0BAQEFAAOCQAQ0AMIIBCgKCAQE
FqwKhDHTcgHyWCNgzBI7L7e9IsQ6E9pfUU5daEHFrF
PEuUNwobGw4oYS/vUv6u0a0/YTqtVu2M9N4bU/YKk
xzASSJ4qKs0F543RmKipvveiE7aoMkrxnQ5sh2nWj
ulkxBIGo9sQysl9Vpr9TDJUpiqPeYiK2fRK06umA0
iVkgHl7FoL1+lfloZDkM70BFu3L3Ye62kuxw5
HMQVeZQfnNUQNPS0910tXywOE2inAo9dgCajPnioKc
KBpgzAkff7H9mQb72a2MVYTESo9jT4xU1DJuW+P
fYr1UMIRiKyH9glbq1cnbtpPpQ==
-----END CERTIFICATE REQUEST-----
Certificate signing request (CSR)

$ openssl req -in example.com.csr -text

Certificate Request:
   Data:
      Version: 1 (0x0)
      Subject: C = US, CN = example.com
      Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
         RSA Public-Key: (2048 bit)
         Modulus:
            00:d0:0b:d6:34:0a:6e:cc:38:fb:7d:fd:5f:5a:a9:
            87:03
         Exponent: 65537 (0x10001)
      Attributes:
         challengePassword :asdasd
      Signature Algorithm: sha256WithRSAEncryption
Certificate signing request (CSR)

```bash
$ openssl req -in example.com.csr -outform der -out example.com.csr.der
$ dumpasn1 example.com.csr.der
0 639: SEQUENCE {
  4 359: SEQUENCE {
    8 1: INTEGER 0
    11 35: SEQUENCE {
      13 11: SET {
        15 9: SEQUENCE {
          17 3: OBJECT IDENTIFIER countryName (2 5 4 6)
          22 2: PrintableString 'US'
            : }
        26 20: SET {
          28 18: SEQUENCE {
            30 11: OBJECT IDENTIFIER commonName (2 5 4 3)
            35 11: UTF8String 'example.com'
              : }
          48 290: SEQUENCE {
            52 13: SEQUENCE {
              54 9: OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
              65 0: NULL
                : }
            67 271: BIT STRING, encapsulates {
              72 266: SEQUENCE {
                76 257: INTEGER
                  : 00 D0 0B D6 34 0A 6E CC 38 FB 7D FD 5F 5A A9 88
                    [ Another 129 bytes skipped ]
                337 3: INTEGER 65537
                  : }
              342 3: [0] {
                344 21: SEQUENCE {
                  346 9: OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
                  357 8: SET {
                    359 6: UTF8String 'asdasd'
                      : }
                367 13: SEQUENCE {
                  369 9: OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
                    : }
              373 11: SEQUENCE {
                375 9: OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
                378 0: NULL
                  : }
              385 257: BIT STRING
                : 1A 23 45 C1 6E BA 57 49 DE 73 28 00 BD 8C 1E E8
                  29 FA 21 D9 7D 1A 22 DB 56 FE 1A 93 E1 E9 0E 82
                    D0 05 47 E9 F3 93 B4 89 F4 4D 7A FC 24 55 46 58
                      E0 B3 3D 80 AD 01 88 0D 50 5A F8 15 85 9E 78 B9
                        24 70 12 4F AB BE 80 5A 27 05 D2 AD DD D1 02 87
                          8F A8 6C 1A 1A D6 39 6A 61 38 8A C9 E3 2C 42 95
                            E5 34 B0 3E 5E FE 53 13 D2 DC 86 AA 02 94 C6 5C
                              D9 7A 53 76 7A 17 F0 91 B4 1D 5D 50 64 1C A0 EC
                                }

CertificationRequest ::= SEQUENCE {
  certificationRequestInfo CertificationRequestInfo,
  signatureAlgorithm AlgorithmIdentifier,
  signature BIT STRING
}

CertificationRequestInfo ::= SEQUENCE {
  version INTEGER v1(0),
  subject Name,
  subjectPKInfo SubjectPublicKeyInfo,
  attributes [0] IMPLICIT Attributes
}

PKCS#10: https://tools.ietf.org/html/rfc2986
```
Certificate enrollment

Verify Domain

Your certificate has been created and is ready for domain verification.

example.com

Congratulations, your SSL certificate is on its way! However, you need to verify ownership of your domain before installing your certificate. Please follow the steps below.

Verification Method for example.com

We need you to verify ownership of each domain in your certificate. Please select your preferred verification method and click "Next Step".

- Email Verification
  - Please select an email address below

    admin@example.com

  - How to use email verification?
    - After selecting an email, click "Next Step".

- DNS (CNAME)
- HTTP File Upload

Finalize
Task: Certificate issuer

Implement a utility that issues a TLS server certificate based on a certificate signing request.

$ ./issue_cert.py
usage: issue_cert.py CA_cert_file CA_private_key_file csr_file output_cert_file

$ ./issue_cert.py CA_cert.pem CA_priv.pem example.com.csr issued.pem
[+] Issuing certificate for "example.com"

$ openssl verify -CAfile CA_cert.pem -purpose sslserver issued.pem
issued.pem: OK

$ openssl verify -CAfile CA_cert.pem -purpose smimesign issued.pem
CN = example.com
error 26 at 0 depth lookup: unsupported certificate purpose
error issued.pem: verification failed

- Must support PEM/DER inputs, PEM output
- Sign subject’s certificate using the CA’s private key
- Use sha256WithRSAEncryption algorithm (1.2.840.113549.1.1.11)
Task: Certificate issuer

• Specify any certificate serial number you want
• Validity dates at least 3 month apart from today (may hardcode)
• Fetch issuer’s distinguished name from CA certificate
• Copy only the CN from the subject’s CSR DN (other fields must be ignored!)
• Fetch subject’s public key from CSR (subjectPublicKeyInfo)
• Certificate extensions (critical:TRUE):
  • basic constraints CA:FALSE
  • key usage: digitalSignature
  • extended key usage: id-kp-serverAuth
• No need to verify CSR’s signature
• Use your own DER encoder and pyasn1 for decoding
$ openssl x509 -in issued.pem -text
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 4138208570 (0xf6a80d3a)
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C = US, O = Trustworthy Inc, OU = IT dep, CN = Trustworthy Root CA
Validity
  Not Before: Jan 1 00:00:00 2022 GMT
  Not After : Jan 1 00:00:00 2023 GMT
Subject: CN = example.com
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
    RSA Public-Key: (2048 bit)
      Modulus:
    Exponent: 65537 (0x10001)
X509v3 extensions: 
  X509v3 Basic Constraints: critical
    CA:FALSE
  X509v3 Key Usage: critical
    Digital Signature
  X509v3 Extended Key Usage: critical
    TLS Web Server Authentication
Signature Algorithm: sha256WithRSAEncryption
Task: Hints

• **pyasn1** will fail to decode CSR if it contains no attributes (challenge password) since it expects implicit tagging:
  • Make sure your CSR contains a challenge password

• **pyasn1** can easily encode decoded substructures:
  
  ```python
  encoder.encode(decoder.decode(der)[0][0][5])
  ```

• You may want to implement `asn1_bitstring_der()` which takes a byte string as input (padding always 0x00)

• Read ASN.1 definitions or `dumpasn1` example certificates to find out the encoding of the certificate and its extensions
  
  ```bash
  openssl x509 -inform pem -in cert.pem -outform der -out cert.der
  ```

• For debugging use two windows to compare your `dumpasn1` output with reference output
Trust on first use (TOFU)

$ ssh user@example.com
The authenticity of host 'example.com (93.184.216.34)' can't be established.
RSA key fingerprint is SHA256:2x7va2E9JDr1xwWRemr5gYQrguFjBGikei9bXDl6K44.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'example.com,93.184.216.34' (RSA) to the list of known hosts.
user@example.com's password:

$ cat ~/.ssh/known_hosts
example.net,93.184.216.33 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA2SGQCzV/vcs1su6eSM52Skksu2n9J3zdFjmSfgBexample.com,93.184.216.34 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA+EuCKTMZU9LYhNqBLfz8KGqvLv90wiadUOAHv2

$ ssh user@example.com
user@example.com's password:

$ ssh user@example.com
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
SHA256:4J1lsTx1vbxHYaF6ALHD/dTkbx5N6ViZZQtNIiAKd04k.
Please contact your system administrator.
Web of trust (WOT)
Questions

• What does PKI and X.509 certificates solve?
• Which are the two most important fields in the X.509 certificate?
• Who defines trusted CAs for digital signature certificates?
• What is the Hardware Security Module useful for?
• What does the browser check in a certificate received from the server?
• Who defines trusted CAs for web server certificates?
• How are DV certificates different from OV certificates?
• How does a CA verify whether an entity owns the domain?