Contents

X.509

In cryptography, **X.509** is an ITU-T standard for a public key infrastructure (PKI) and Privilege Management Infrastructure (PMI). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.

History and usage

X.509 was initially issued on July 3, 1988 and was begun in association with the X.500 standard. It assumes a strict hierarchical system of certificate authorities (CAs) for issuing the certificates. This contrasts with web of trust models, like PGP, where anyone (not just special CAs) may sign and thus attest to the validity of others' key certificates. Version 3 of X.509 includes the flexibility to support other topologies like bridges and meshes (RFC 4158). It can be used in a peer-to-peer, OpenPGP-like web of trust[*citation needed*], but was rarely used that way as of 2004. The X.500 system has only ever been implemented by sovereign nations for state identity information sharing treaty fulfillment purposes, and the IETF's Public-Key Infrastructure (X.509), or **PKIX**, working group has adapted the standard to the more flexible organization of the Internet. In fact, the term *X.509 certificate* usually refers to the IETF's PKIX Certificate and CRL Profile of the X.509 v3 certificate standard, as specified in RFC 5280, commonly referred to as PKIX for ***Public Key Infrastructure (X.509)***.[*citation needed*]

Certificates

In the X.509 system, a certification authority issues a certificate binding a public key to a particular distinguished name in the X.500 tradition, or to an *alternative name* such as an e-mail address or a DNS entry.[*citation needed*]

An organization's trusted root certificates can be distributed to all employees so that they can use the company PKI system. Browsers such as Internet Explorer, Firefox, Opera, Safari and Chrome come with a predetermined set of root certificates pre-installed, so SSL certificates from larger vendors will work instantly; in effect the browsers' developers determine which CAs are trusted third parties for the browsers' users.[*citation needed*]

X.509 also includes standards for certificate revocation list (CRL) implementations, an often neglected aspect of PKI systems. The IETF-approved way of checking a certificate's validity is the Online Certificate Status Protocol (OCSP). Firefox 3 enables OCSP checking by default along with versions of Windows including Vista and later.[*citation needed*]

Structure of a certificate

The structure foreseen by the standards is expressed in a formal language, namely Abstract Syntax Notation One.

The structure of an X.509 v3 digital certificate is as follows:

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm

        - Subject Public Key
    - Issuer Unique Identifier (optional)
    - Subject Unique Identifier (optional)
    - Extensions (optional)
        - ...
- Certificate Signature Algorithm
- Certificate Signature

Each extension has its own id, expressed as Object identifier, which is a set of values, together with either a critical or non-critical indication. A certificate-using system MUST reject the certificate if it encounters a critical extension that 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.[1]

The structure of Version 1 is given in RFC 1422 [2].

ITU-T introduced issuer and subject unique identifiers in version 2 to permit the reuse of issuer or subject name after some time. An example of reuse will be when a CA goes bankrupt and its name is deleted from the country's public list. After some time another CA with the same name may register itself, even though it is unrelated to the first one. However, IETF recommends that no issuer and subject names be reused. Therefore, version 2 is not widely deployed in the Internet.[*citation needed*]

Extensions were introduced in version 3. A CA can use extensions to issue a certificate only for a specific purpose (e.g. only for signing digital object). Each extension can be critical or non-critical. If an extension is critical and the system processing the certificate does not recognize the extension or cannot process it, the system MUST reject the entire certificate. A non-critical extension, on the other hand, can be ignored while the system processes the rest of the certificate.[*citation needed*]

In all versions, the serial number MUST be unique for each certificate issued by a specific CA (as mentioned in RFC 2459).

Extensions informing a specific usage of a certificate

RFC 5280 (and its predecessors) defines a number of certificate extensions which indicate how the certificate should be used. Most of them are arcs from the `joint-iso-ccitt(2) ds(5) id-ce(29)` OID. Some of the most common, defined in section 4.2.1, are:

- Basic Constraints, `{ id-ce 19 }`, are used to indicate whether the certificate belongs to a CA.
- Key Usage, `{ id-ce 15 }`, provides a bitmap specifying the cryptographic operations which may be performed using the public key contained in the certificate; for example, it could indicate that the key should be used for signatures but not for encipherment.
- Extended Key Usage, `{ id-ce 37 }`, is used, typically on a leaf certificate, to indicate the purpose of the public key contained in the certificate. It contains a list of OIDs, each of which indicates an allowed use. For example, `{ id-pkix 3 1 }` indicates that the key may be used on the server end of a TLS or SSL connection; `{ id-pkix 3 4 }` indicates that the key may be used to secure email.

In general, if a certificate has several extensions restricting its use, all restrictions must be satisfied for a given use to be appropriate. RFC 5280 gives the specific example of a certificate containing both keyUsage and extendedKeyUsage: in this case, both must be processed and the certificate can only be used if both extensions are coherent in specifying the usage of a certificate. For example, NSS uses both extensions to specify certificate usage.[3]

Certificate filename extensions

Common filename extensions for X.509 certificates are:[citation needed]

- `.pem` – (Privacy-enhanced Electronic Mail) Base64 encoded DER certificate, enclosed between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----"
- `.cer`, `.crt`, `.der` – usually in binary DER form, but Base64-encoded certificates are common too (see `.pem` above)
- `.p7b`, `.p7c` – PKCS#7 SignedData structure without data, just certificate(s) or CRL(s)
- `.p12` – PKCS#12, may contain certificate(s) (public) and private keys (password protected)
- `.pfx` – PFX, predecessor of PKCS#12 (usually contains data in PKCS#12 format, e.g., with PFX files generated in IIS)

PKCS#7 is a standard for signing or encrypting (officially called "enveloping") data. Since the certificate is needed to verify signed data, it is possible to include them in the SignedData structure. A `.P7C` file is a degenerated SignedData structure, without any data to sign.[citation needed]

PKCS#12 evolved from the *personal information exchange* (PFX) standard and is used to exchange public and private objects in a single file.[citation needed]

Sample X.509 certificates

This is an example of a decoded X.509 certificate for `www.freesoft.org`, generated with OpenSSL—the actual certificate is about 1 kB in size. It was issued by Thawte (since acquired by VeriSign and now owned by Symantec), as stated in the Issuer field. Its subject contains many personal details, but the most important part is usually the common name (CN), as this is the part that must match the host being authenticated. Also included is an RSA public key (modulus and public exponent), followed by the signature, computed by taking a MD5 hash of the first part of the certificate and signing it (applying the encryption operation) using Thawte's RSA private key.

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

To validate this certificate, one needs a second certificate that matches the Issuer (Thawte Server CA) of the first certificate. First, one verifies that the second certificate is of a CA kind; that is, that it can be used to issue other certificates. This is done by inspecting a value of the *CA* attribute in the *X509v3 extension* section. Then the RSA public key from the CA certificate is used to decode the signature on the first certificate to obtain a MD5 hash, which must match an actual MD5 hash computed over the rest of the certificate. An example CA certificate follows:

```
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number: 1 (0x1)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Aug  1 00:00:00 1996 GMT
           Not After : Dec 31 23:59:59 2020 GMT
       Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                   68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                   85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                   6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                   6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                   29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                   6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                   5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                   3a:c2:b5:66:22:12:d6:87:0d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
        a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
        3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
        4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
        8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
        e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
        b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
        70:47
```

This is an example of a self-signed certificate, as the issuer and subject are the same. There's no way to verify this certificate except by checking it against itself; instead, these top-level certificates are manually stored by web browsers. Thawte is one of the root certificate authorities recognized by both Microsoft and Netscape. This certificate comes with the web browser and is trusted by default. As a long-lived, globally trusted certificate that can sign anything (as there are no constraints in the *X509v3 Basic Constraints* section), its matching private key has to be closely guarded.

Security

There are a number of publications about PKI problems by Bruce Schneier, Peter Gutmann and other security experts.[4][5]

Certificate complexity

A devastating majority of Internet users, either business or social, currently lack the basic ability, knowledge, or even willingness to effectively use cryptographic applications in a way that can successfully deter imminent threats. The complexity of this task is the Achilles' heel of public key cryptography. A lack of user friendliness and overall usability thus affects solution efficiency.[6] To deal with such issues, software companies have included a bundle of root certificates, which have been audited for security purposes, into user browsers and operating systems. For the sake of user friendliness and interoperability, all web browsers and operating systems currently contain this audited *Trusted Root Store* of certificate issuing authorities. Certificates issued by these organizations, or their subordinate authorities, are transparently trusted by relying entities. These certificates are automatically deemed as secure and trustworthy, as opposed to those issued by “unknown” issuers, which a relying party is warned not to trust. This essentially interprets into certificates published by all authorities that have not been included in the root store. This approach attempts to make the provision of system security automatic and transparent, and essentially removes from the end user the decision making process about the trustworthiness of web entities.

The X.509 standard was primarily designed to support the X.500 structure, but today's use cases center around the web. Many features are of little or no relevance today. The X.509 specification suffers from being over-functional and underspecified and the normative information is spread across many documents from different standardization bodies. Several profiles were developed to solve this, but these introduce interoperability issues and did not fix the problem.

Architectural weaknesses

- Use of blacklisting invalid certificates (using CRLs and OCSP) instead of whitelisting,
- CRLs are notably a poor choice because of large sizes and convoluted distribution patterns,
- Ambiguous OCSP semantics and lack of historical revocation status,
- Revocation of root certificates is not addressed,
- Aggregation problem: Identity claims (authenticate with an identifier), attribute claims (submit a bag of vetted attributes), and policy claims are combined in a single container. This raises privacy, policy mapping, and maintenance issues,
- Delegation problem: CAs cannot technically restrict subordinate CAs from issuing certificates outside a limited namespaces or attribute set; this feature of X.509 is not in use. Therefore a large number of CAs exist on the Internet, and classifying them and their policies is an insurmountable task. Delegation of authority within an organization cannot be handled at all, as in common business practice.
- Federation problem: Certificate chains that are the result of subordinate CAs, bridge CAs, and cross-signing make validation complex and expensive in terms of processing time. Path validation semantics may be ambiguous. The hierarchy with a third-party trusted party is the only model. This is inconvenient when a bilateral trust relationship is already in place.

Problems with certificate authorities

- The subject, not the relying party, purchases certificates. The subject will often utilize the cheapest issuer, so quality is not being paid for in the competing market. This is partly addressed by Extended Validation certificates.
- Certification authorities deny almost all warranties to the user (including subject or even relying parties).
- The expiration date should be used to limit the time the key strength is deemed sufficient. This parameter is abused by certification authorities to charge the client an extension fee. This places an unnecessary burden on the user with key roll-over.
- "Users use an undefined certification request protocol to obtain a certificate which is published in an unclear location in a nonexistent directory with no real means to revoke it."

Implementation issues

Implementations suffer from design flaws, bugs, different interpretations of standards and lack of interoperability of different standards. Some problems are:[*citation needed*]

- Many implementations turn off revocation check:
  - Seen as obstacle, policies are not enforced
  - If it was turned on in all browsers by default, including code signing, it would probably crash the infrastructure.
- DNs are complex and little understood (lack of canonicalization, internationalization problems, ..)
- rfc822Name has two notations
- Name and policy constraints hardly supported
- Key usage ignored, first certificate in a list being used
- Enforcement of custom OIDs is difficult
- Attributes should not be made critical because it makes clients crash.
- Unspecified length of attributes lead to product-specific limits

Exploits

- MD2-based certificates were used for a long time and were vulnerable to preimage attacks. Since the root certificate already had a self-signature, attackers could use this signature and use it for an intermediate certificate.
- In 2005, Arjen Lenstra and Benne de Weger demonstrated "how to use hash collisions to construct two X.509 certificates that contain identical signatures and that differ only in the public keys", achieved using a collision attack on the MD5 hash function.
- In 2008, Alexander Sotirov and Marc Stevens presented at the Chaos Communication Congress a practical attack that allowed them to create a rogue Certificate Authority, accepted by all common browsers, by exploiting the fact that RapidSSL was still issuing X.509 certificates based on MD5.
- X.509 certificates based on SHA-1 had been deemed to be secure up until very recent times. In April 2009 at the Eurocrypt Conference [7], Australian Researchers of Macquarie University presented "Automatic Differential Path Searching for SHA-1" [8]. The researchers were able to deduce a method which increases the likelihood of a collision by several orders of magnitude.[9]
- Domain-validated certificates ("Junk certificates") are still trusted by web browsers, and can be obtained with little effort from commercial CAs.[*citation needed*]
- EV-certificates are of very limited help, because Browsers do not have policies that disallow EV-certificates, Zusman and Sotirov Blackhat 2009 [10]
- There are implementation errors with X.509 that allow e.g. falsified subject names using null-terminated strings Marlinspike Blackhat 2009 [11] or code injections attacks in certificates.
- By using illegal[12] 0x80 padded subidentifiers of Object Identifiers, wrong implementations or by using integer-overflows of the client's browsers, an attacker can include an unknown attribute in the CSR, which the CA will sign, which the client wrongly interprets as "CN" (OID=2.5.4.3). Dan Kaminsky at the 26th Chaos Communication Congress "Black OPs of PKI"

PKI standards for X.509

- PKCS7 (Cryptographic Message Syntax Standard - public keys with proof of identity for signed and/or encrypted message for PKI)[*citation needed*]
- Secure Sockets Layer (SSL) - cryptographic protocols for Internet secure communications[*citation needed*]
- Online Certificate Status Protocol (OCSP) / Certificate Revocation List (CRL) - this is for validating proof of identity[*citation needed*]
- PKCS12 (Personal Information Exchange Syntax Standard) - used to store a private key with the appropriate public key certificate[*citation needed*]

Certification authority

A certification authority (CA) is an entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many public key infrastructure (PKI) schemes.[*citation needed*]

There are many commercial CAs that charge for their services. Institutions and governments may have their own CAs, and there are free CAs.[*citation needed*]

Public-Key Infrastructure (X.509) Working Group

The The Public-Key Infrastructure (X.509) working group (PKIX) was a working group of the Internet Engineering Task Force dedicated to creating RFCs and other standard documentation on issues related to public key infrastructure based on X.509 certificates. PKIX was established in Autumn 1995 in conjunction with the National Institute of Standards and Technology. The Working Group was closed in November 2013.

Protocols and standards supporting X.509 certificates

- TLS/SSL
- S/MIME (Secure Multipurpose Internet Mail Extensions)
- IPsec
- SSH
- Smart card
- HTTPS
- EAP
- LDAP
- Trusted Computing Group (TNC TPM NGSCB)
- CableLabs (North American Cable Industry Technology Forum)
- WS-Security
- XMPP
- Microsoft Authenticode
- OPC UA

References

[1] retrieved 12 February 2013 (http://tools.ietf.org/html/rfc5280#section-4.2,)
[2] http://www.ietf.org/rfc/rfc1422
[3] All About Certificate Extensions (http://www.mozilla.org/projects/security/pki/nss/tech-notes/tn3.html)
[4] Top 10 PKI risks (http://hackerproof.org/technotes/pki/pki_risks.pdf)
[5] Peter Gutmann, PKI: it's not dead, just resting (http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1023787)
[6] Zissis, D. & Lekkas, D., “Trust coercion in the name of usable Public Key Infrastructure”, Security And Communication Networks, John Wiley & Sons, (2013)
[7] http://www.iacr.org/conferences/eurocrypt2009/
[8] http://eurocrypt2009rump.cr.yp.to/837a0a8086fa6ca714249409ddfae43d.pdf
[9] SHA-1 Collision Attacks Now 2^{52} (http://www.secureworks.com/research/blog/index.php/2009/6/3/sha-1-collision-attacks-now-252/)
[10] http://www.blackhat.com/presentations/bh-usa-09/SOTIROV/BHUSA09-Sotirov-AttackExtSSL-PAPER.pdf
[11] http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf
[12] Rec. ITU-T X.690, clause 8.19.2

General

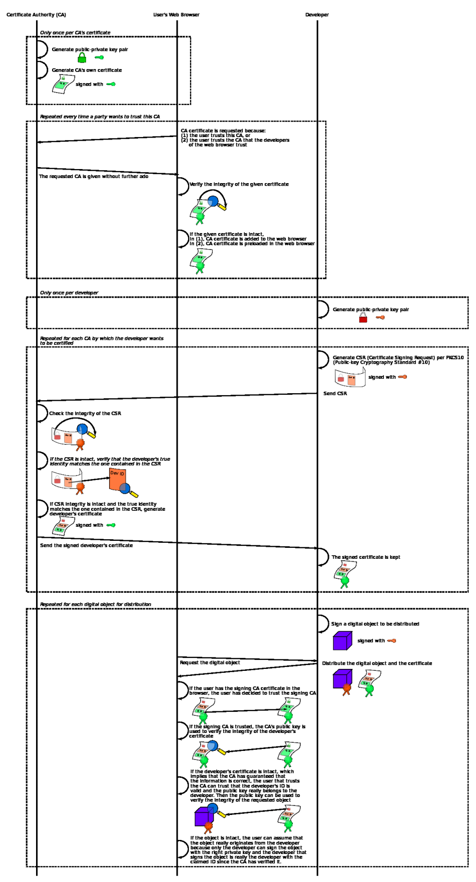
- ITU-T Recommendation X.509 (http://www.itu.int/rec/T-REC-X.509) (2005): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 08/05.
- C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure: Certificate Management Protocols", RFC 2510, March 1999
- Housley, R., W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 3280, April 2002. Obsoleted by RFC 5280, Obsoletes RFC 2459/ updated by RFC 4325, RFC 4630.
- Housley, R., W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 2459, January 1999. Obsoleted by RFC 3280.
- Arjen Lenstra, Xiaoyun Wang and Benne de Weger, On the possibility of constructing meaningful hash collisions for public keys, full version, with an appendix on colliding X.509 certificates, 2005 (http://www.win.tue.nl/~bdeweger/CollidingCertificates/) (see also (http://eprint.iacr.org/2005/067)).

External links

- ITU-T Recommendation X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks (http://www.itu.int/rec/T-REC-X.509/en)
- Peter Gutmann's articles, an overview of PKI (http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf), X.509 implementation notes X.509 Style Guide (http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt)
- PKIX website (http://www.ietf.org/html.charters/pkix-charter.html)
- Enterprise Trust Integration and Web Services Security standards and demos (http://www.trustedwebservices.org)
- FAQ from RSA Labs (http://www.rsasecurity.com/rsalabs/node.asp?id=2155)
- Sun Inc. - Secure code guidelines (http://java.sun.com/security/seccodeguide.html)
- RFC 4158 - Internet X.509 Public Key Infrastructure: Certification Path Building
- CSR Decoder and Certificate Decoder (http://certlogik.com/decoder) - can be used to decode and examine an encoded CSR or certificate.
- SSL Checker (http://certlogik.com/sslchecker/) - can be used to test a certificate and that it has been installed correctly
- phpseclib: X.509 Decoder (http://phpseclib.sourceforge.net/x509/decoder.php) - decodes to an associative array whose keys correspond to X.509's ASN.1 description
- Certificate Lookup (http://www.ssltools.com/certificate_lookup) - ssl tool that can be used to verify and troubleshoot X.509 certificate installations or examine their contents
- SSLTools Manager for Windows (http://www.ssltools.com/manager) - Windows utility to create X.509 certificates for IIS or Exchange.

Public key certificate

In cryptography, a **public key certificate** (also known as a **digital certificate** or **identity certificate**) is an electronic document that uses a digital signature to bind a public key with an identity — information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual.


Certificate Authority (CA)
User's Web Browser
Developer
Only once per CA's certificate
Generate public-private key pair
Generate CA's own certificate
signed with
Repeated every time a party wants to trust this CA
CA certificate is requested because:
(1) the user trusts this CA, or
(2) the user trusts the CA that the developers of the web browser trust
The requested CA is given without further ado
Verify the integrity of the given certificate
If the given certificate is intact,
in (1), CA certificate is added to the web browser
in (2), CA certificate is preloaded in the web browser
Only once per developer
Generate public-private key pair
Repeated for each CA by which the developer wants to be certified
Generate CSR (Certificate Signing Request) per PKCS10 (Public-key Cryptography Standard #10)
signed with
Send CSR
Check the integrity of the CSR
If the CSR is intact, verify that the developer's true identity matches the one contained in the CSR
Dev ID
If CSR integrity is intact and the true identity matches the one contained in the CSR, generate developer's certificate
signed with
Send the signed developer's certificate
The signed certificate is kept
Repeated for each digital object for distribution
Sign a digital object to be distributed
signed with
Request the digital object
Distribute the digital object and the certificate
If the user has the signing CA certificate in the browser, the user has decided to trust the signing CA
If the signing CA is trusted, the CA's public key is used to verify the integrity of the developer's certificate
If the developer's certificate is intact, which implies that the CA has guaranteed that the information is correct, the user that trusts the CA can trust that the developer's ID is valid and the public key really belongs to the developer. Then the public key can be used to verify the integrity of the requested object
If the object is intact, the user can assume that the object really originates from the developer because only the developer can sign the object with the right private key and the developer that signs the object is really the developer with the claimed ID since the CA has verified it.
Diagram of an example usage of digital certificate

In a typical public key infrastructure (PKI) scheme, the signature will be of a certificate authority (CA). In a web of trust scheme, the signature is of either the user (a self-signed certificate) or other users ("endorsements"). In either case, the signatures on a certificate are attestations by the certificate signer that the identity information and the public key belong together.

For provable security, this reliance on something external to the system has the consequence that any public key certification scheme has to rely on some special setup assumption, such as the existence of a certificate authority.[1]

Operating systems

Certificates can be created for Unix-based servers with tools such as OpenSSL's `ca` [2] command.[3] or SuSE's `gensslcert`. These may be used to issue unmanaged certificates, certification authority (CA) certificates for managing other certificates, and user and/or computer certificate requests to be signed by the CA, as well as a number of other certificate related functions.

Similarly, Microsoft Windows 2000 Server and Windows Server 2003 contain a certification authority (CA) as part of Certificate Services for the creation of digital certificates. In Windows Server 2008 the CA may be installed as part of Active Directory Certificate Services. The CA is used to manage and centrally issue certificates to users and/or computers. Microsoft also provides a number of different certificate utilities, such as SelfSSL.exe for creating unmanaged certificates, and Certreq.exe for creating and submitting certificate requests to be signed by the CA, and certutil.exe for a number of other certificate related functions.

Mac OS X comes with the Keychain Access application, which, along with keeping track of the user's logins and credentials, is able to perform various certificate-related services.

Contents of a typical digital certificate

Serial Number: Used to uniquely identify the certificate.

Subject: The person, or entity identified.

Signature Algorithm: The algorithm used to create the signature.

Signature: The actual signature to verify that it came from the issuer.

Issuer: The entity that verified the information and issued the certificate.

Valid-From: The date the certificate is first valid from.

Valid-To: The expiration date.

Key-Usage: Purpose of the public key (e.g. encipherment, signature, certificate signing...).

Public Key: The public key.

Thumbprint Algorithm: The algorithm used to hash the public key certificate.

Thumbprint (also known as fingerprint) : The hash itself, used as an abbreviated form of the public key certificate.

Classification

Vendor defined classes

VeriSign uses the concept of classes for different types of digital certificates:[4]

- Class 1 for individuals, intended for email.
- Class 2 for organizations, for which proof of identity is required.
- Class 3 for servers and software signing, for which independent verification and checking of identity and authority is done by the issuing certificate authority.
- Class 4 for online business transactions between companies.
- Class 5 for private organizations or governmental security.

Other vendors may choose to use different classes or no classes at all as this is not specified in the PKI standards.

Usage in the European Union

The EU Directive 1999/93/EC on a Community framework for electronic signatures defines the term *qualified certificate* as "a certificate which meets the requirements laid down in Annex I and is provided by a certification-service-provider who fulfils the requirements laid down in Annex II":

> **Annex I**: Requirements for qualified certificates
>
> Qualified certificates must contain:
>
> (a) an indication that the certificate is issued as a qualified certificate;
>
> (b) the identification of the certification-service-provider and the State in which it is established;
>
> (c) the name of the signatory or a pseudonym, which shall be identified as such;
>
> (d) provision for a specific attribute of the signatory to be included if relevant, depending on the purpose for which the certificate is intended;
>
> (e) signature-verification data which correspond to signature-creation data under the control of the signatory;
>
> (f) an indication of the beginning and end of the period of validity of the certificate;
>
> (g) the identity code of the certificate;
>
> (h) the advanced electronic signature of the certification-service-provider issuing it;
>
> (i) limitations on the scope of use of the certificate, if applicable; and

(j) limits on the value of transactions for which the certificate can be used, if applicable.

Annex II Requirements for certification-service-providers issuing qualified certificates

Certification-service-providers must:

(a) demonstrate the reliability necessary for providing certification services;

(b) ensure the operation of a prompt and secure directory and a secure and immediate revocation service;

(c) ensure that the date and time when a certificate is issued or revoked can be determined precisely;

(d) verify, by appropriate means in accordance with national law, the identity and, if applicable, any specific attributes of the person to which a qualified certificate is issued;

(e) employ personnel who possess the expert knowledge, experience, and qualifications necessary for the services provided, in particular competence at managerial level, expertise in electronic signature technology and familiarity with proper security procedures; they must also apply administrative and management procedures which are adequate and correspond to recognised standards;

(f) use trustworthy systems and products which are protected against modification and ensure the technical and cryptographic security of the process supported by them;

(g) take measures against forgery of certificates, and, in cases where the certification-service-provider generates signature-creation data, guarantee confidentiality during the process of generating such data;

(h) maintain sufficient financial resources to operate in conformity with the requirements laid down in the Directive, in particular to bear the risk of liability for damages, for example, by obtaining appropriate insurance;

(i) record all relevant information concerning a qualified certificate for an appropriate period of time, in particular for the purpose of providing evidence of certification for the purposes of legal proceedings. Such recording may be done electronically;

(j) not store or copy signature-creation data of the person to whom the certification-service-provider provided key management services;

(k) before entering into a contractual relationship with a person seeking a certificate to support his electronic signature inform that person by a durable means of communication of the precise terms and conditions regarding the use of the certificate, including any limitations on its use, the existence of a voluntary accreditation scheme and procedures for complaints and dispute settlement. Such information, which may be transmitted electronically, must be in writing and in readily understandable language. Relevant parts of this information must also be made available on request to third-parties relying on the certificate;

(l) use trustworthy systems to store certificates in a verifiable form so that:

- only authorized persons can make entries and changes,
- information can be checked for authenticity,
- certificates are publicly available for retrieval in only those cases for which the certificate-holder's consent has been obtained, and
- any technical changes compromising these security requirements are apparent to the operator.

Certificates and web site security

The most common use of certificates is for HTTPS-based web sites. A web browser validates that a TLS (Transport Layer Security) web server is authentic, so that the user can feel secure that his/her interaction with the web site has no eavesdroppers and that the web site is who it claims to be. This security is important for electronic commerce. In practice, a web site operator obtains a certificate by applying to a certificate provider (a CA that presents as a commercial retailer of certificates) with a certificate signing request. The certificate request is an electronic document that contains the web site name, contact email address, company information and the public key (for security reasons the private key is not part of the request and is not sent to the certificate authority). The certificate provider signs the request, thus producing a public certificate. During web browsing, this public certificate is served to any web browser that connects to the web site and proves to the web browser that the provider believes it has issued a certificate to the owner of the web site.

Before issuing a certificate, the certificate provider will request the contact email address for the web site from a public domain name registrar, and check that published address against the email address supplied in the certificate request. Therefore, an https web site is only secure to the extent that the end user can be sure that the web site is operated by someone in contact with the person who registered the domain name.

As an example, when a user connects to `https://www.example.com/` with his browser, if the browser does not give any certificate warning message, then the user can be theoretically sure that interacting with `https://www.example.com/` is equivalent to interacting with the entity in contact with the email address listed in the public registrar under "example.com", even though that email address may not be displayed anywhere on the web site. No other surety of any kind is implied. Further, the relationship between the purchaser of the certificate, the operator of the web site, and the generator of the web site content may be tenuous and is not guaranteed. At best, the certificate guarantees uniqueness of the web site, provided that the web site itself has not been compromised (hacked) or the certificate issuing process subverted.

Extended Validation

Certificate providers issue "higher security" certificates that require further security checks, and incur higher fees. This is called an Extended Validation. These security checks cross reference the owner of the domain name with the owner of the legal entity that claims to operate under it. (These checks may involve the presentation of utility bills, passports, etc.) The difference between these higher security certificates and regular certificates are that the browser URL bar changes to a different color, usually green. This improved security assumes the user knows the meaning of the colors, and would choose to navigate away from the site if the color code was not commensurate with the purpose of the web site. To be clear, for a https:// web site URL, the difference between having no certificate, and having a regular certificate is that the browser will *refuse* to access the site without confirming with the user. By comparison, the difference between a regular certificate and an extended validation certificate is merely a change in color.

Weaknesses

A web browser will give no warning to the user if a web site suddenly presents a different certificate, even if that certificate has a lower number of key bits, even if it has a different provider, and even if the previous certificate had an expiry date far into the future.[*citation needed*] However a change from an EV certificate to a non-EV certificate will be apparent as the green bar will no longer be displayed. Where certificate providers are under the jurisdiction of governments, those governments may have the freedom to order the provider to generate any certificate, such as for the purposes of law enforcement. Subsidiary wholesale certificate providers also have the freedom to generate any certificate.

All web browsers come with an extensive built-in list of trusted root certificates, many of which are controlled by organizations that may be unfamiliar to the user. Each of these organizations is free to issue any certificate for any web site and have the guarantee that web browsers that include its root certificates will accept it as genuine. In this instance, end users must rely on the developer of the browser software to manage its built-in list of certificates and on the certificate providers to behave correctly and to inform the browser developer of problematic certificates. While uncommon, there have been incidents, in which fraudulent certificates have been issued: in some cases, the browsers have detected the fraud; in others, some time passed before browser developers removed these certificates from their software.

The list of built-in certificates is also not limited to those provided by the browser developer: users (and to a degree applications) are free to extend the list for special purposes such as for company intranets. This means that if someone gains access to a machine and can install a new root certificate in the browser, that browser will recognize websites that use the inserted certificate as legitimate.

Usefulness versus unsecured web sites

In spite of the limitations described above, certificate-authenticated SSL is considered mandatory by all security guidelines whenever a web site hosts confidential information or performs material transactions. This is because, in practice, in spite of the serious flaws described above, web sites secured by public key certificates are still more secure than unsecured http:// web sites.

References

[1] Ran Canetti: Universally Composable Signature, Certification, and Authentication. CSFW 2004, http://eprint.iacr.org/2003/239
[2] http://www.openssl.org/docs/apps/ca.html
[3] OpenSSL: Documentation ca(1) (http://www.openssl.org/docs/apps/ca.html)
[4] VeriSign Class definitions (https://www.verisign.com/support/roots.html)

External links

- RFC 5280 (http://www.ietf.org/rfc/rfc5280.txt) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
- SSLTools Certificate Lookup (http://www.ssltools.com/certificate_lookup/www.wikipedia.org) Wikipedia.org Public key certificate details

Article Sources and Contributors

X.509 *Source*: http://en.wikipedia.org/w/index.php?oldid=581792847 *Contributors*: Abdull, Aerowolf, Alan J Shea, Alan U. Kennington, Alberge204, AlistairMcMillan, Altermike, Andreiko, Angeltoribio, Angusmca, Armando, Azollman, Baccala@freesoft.org, Barcex, Bender235, Billinghurst, Bomazi, Buttysquirrel, C960657, Captain panda, Cjcollier, Commander, Corti, Craiged, Cryptoki, D.w.chadwick, Daniel.Cardenas, Darac, Darth Sitges, David Woolley, DeTreville, Digi-cs, Dkgdkg, DokReggar, Drondent, Dysprosia, Dzlabing, E2eamon, EagleOne, Eus Kevin, EvgeniGolov, Eyreland, Feezo, Fenring, Foxj, Frap, Glenn, Gorgonzilla, Grawity, Grendelkhan, Gunn1, Gurch, Hoylen, Hymek, Iida-yosiaki, Imran, Inkling, JTN, Jasuus, Jeff J. Snider, JidGom, John Vandenberg, JonHarder, Jonash, Judesba, Jwilkinson, KAtremer, Kdz, Kenschry, Kku, Klamation, Klausner, Kusma, Kw27, Libraequilibra, LightStarch, LilHelpa, Magioladitis, MarkWahl, Matt Crypto, Mesoderm, Michael Hardy, Minghong, MoraSique, Mortense, MrOllie, Mrbakerfield, Msct, Narayan82es, Nealmcb, Neustradamus, Nono64, Nopherox, NuclearWizard, Nuwewsco, Orborde, OverlordQ, Pastore Italy, Paul Foxworthy, Pdejuan, Peaceray, PerryTachett, Pgan002, Q Chris, Racaille, Ram Moskovitz, Rasmus Faber, Rhoerbe, Rich Farmbrough, Robert Illes, Rochus, Rohanuk, Rsrikanth05, Samuel J. Howard, Saqib, Schnolle, Scorpiuss, Sega381, Sid.shetye, Sievebrain, Siryendor, SkyRocketMedia, Sligocki, Smyth, Stevag, Stewartadcock, The Thing That Should Not Be, Thierrytung, Tnshibu, Toshiro92, Tuntable, Unixplumber, Vandevenp, Vegarwe, Vspaceg, Wgd, Whmac, Whophd, Wiarthurhu, Widefox, Wik, WikiReviewer.de, Wrs1864, Ww, Xandi, YUL89YYZ, Zeroshell, Zundark, 184 anonymous edits

Public key certificate *Source*: http://en.wikipedia.org/w/index.php?oldid=582164405 *Contributors*: Acyclopedic, Aerowolf, Alexei Kojenov, Anon lynx, Armando, ArnoldReinhold, Aryaniae, BD2412, BKoteska, BWikime, BazookaJoe, Blutrot, Bryan Derksen, Chatfecter, ClementSeveillac, Codelux, Cody Cooper, Daniel.Cardenas, Davish Krail, Gold Five, Ddas, DeekGeek, Delta 51, Doberek, Downwards, Dpaking, Dvorak.typist, Ecemaml, Egret, Enix150, Eus Kevin, FlippyFlink, Flyer22, Folajimi, Fuzzy Logic, Gr8dude, Grawity, Gryszkalis, Guysha, HDCase, Haakon, Haakon K, Haggis, Het, Hlkii, Hundred and half, Hut 8.5, Intgr, Isnow, Itahmed, Ixfd64, JTN, Javidjamae, Jdthood, JidGom, Jlascar, Jonathanbenari, Jordav, Julesd, Kenschry, Klausner, Kokamomi, Kpatelseo, Leobold1, LimoWreck, Linktolonkar, Loxlie, MER-C, Magioladitis, Matt Crypto, Mekirkpa, Mfreidgeim, Michael Hardy, Mindmatrix, Mnemeson, Mohsens, Moscow Connection, Mr alibebe, Mydogategodshat, Nabieh, Nailed, Nixdorf, Notmicro, Nurg, Paulsheer, PhilipMW, Pmetzger, Pxma, Q Chris, R.javanmard, RP459, Rasmus Faber, Rcbarnes, Remember the dot, Rentzepopoulos, Robertbowerman, Runningboffin, SPS1962W, Schnolle, Shinton, Sinar, Smaines, Steven Weston, Strait, Sunnycomes, Sydius, TOMTOM1974, Tabletop, Teemuk, TonyW, Trammell, Ulrich Müller, Uppfinnarn, Wrs1864, Ww, Xanga, Yaronf, Zarutian, ZimZalaBim, 179 anonymous edits

Image Sources, Licenses and Contributors

License