Harokopio University
Department of Informatics and Telematics

Security

Certificate Validation Process

Professor: Dr. Panagiotis Rizomiliotis
Date: 06/02/2022

By: Georgara Vasiliki 21107

What is a certificate?

Certificates are digital files in every respect, which means that they need to follow a file format to store information (e.g. signatures, keys, issuers, etc.). While private PKI configurations can implement any format for their certificates, publicly trusted PKIs (i.e. those trusted by the browsers) must conform to RFC 5280, which requires the use of the X.509 v3 format.

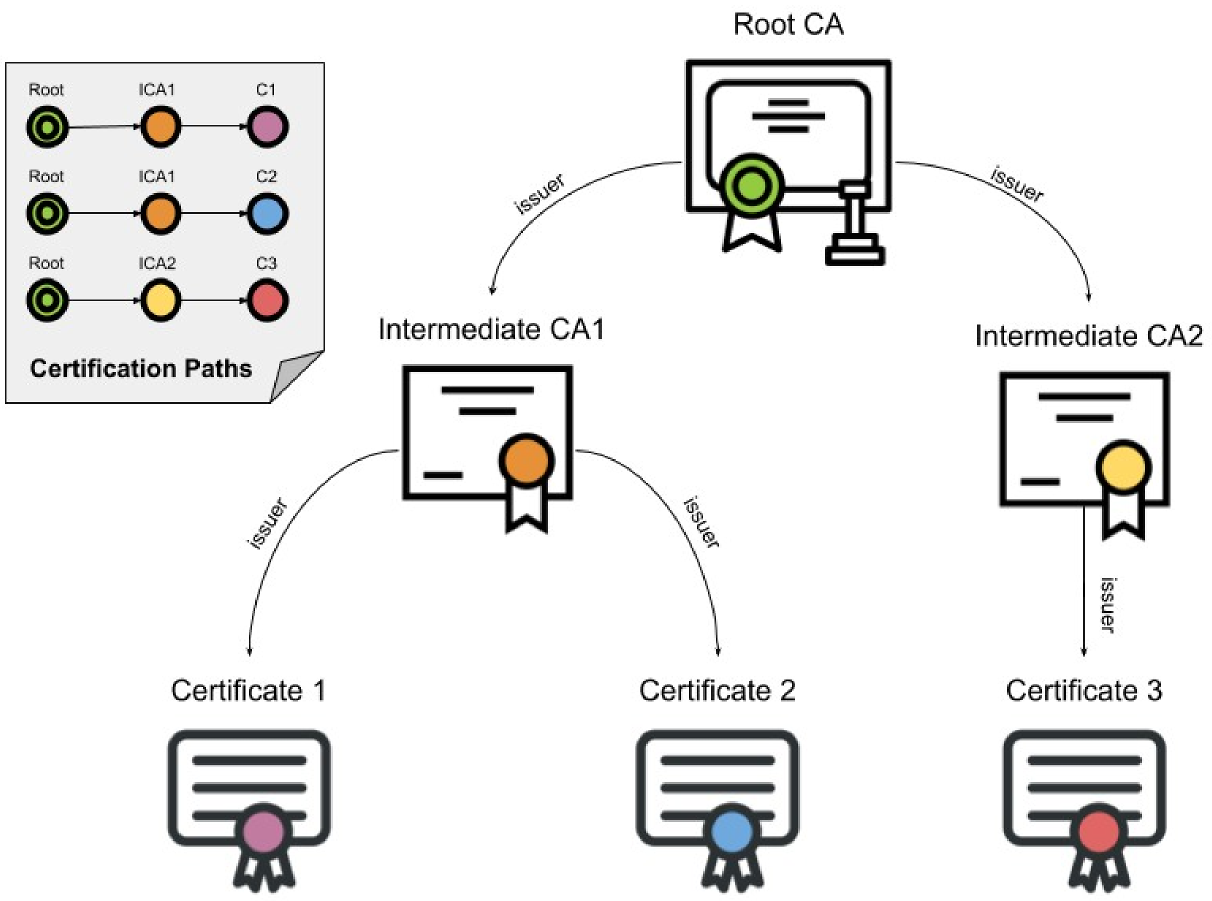
What is an x.509 certificate?

X.509 is a standard format for public key certificates, digital documents that securely associate cryptographic key pairs with identities such as websites, individuals, or organizations. One common application of x.509 certificates is SSL/TLS and HTTPS for authenticated and encrypted web browsing.
Basically, X.509 v3 allows certificates to include additional data, such as usage constraints or policy information, as extensions, with each extension being either critical or non-critical. Browsers can ignore invalid or unrecognized non-critical extensions, but they are required to process and validate all critical ones.

What is a CA?

A certificate authority (CA), also sometimes referred to as a certification authority, is a company or organization that acts to validate the identities of entities (such as websites, email addresses, companies, or individual persons) and bind them to cryptographic keys through the issuance of electronic documents known as digital certificates.
A digital certificate provides:
Authentication, by serving as a credential to validate the identity of the entity that it is issued to.
Encryption, for secure communication over insecure networks such as the Internet.
Integrity of documents signed with the certificate so that they cannot be altered by a third party in transit.
CAs use a private key to cryptographically sign all issued certificates. Such signatures can irrevocably prove that a certificate was issued by a specific CA and that it was not modified after it was signed.

CAs establish ownership of their signing key by holding a self-issued certificate (called the root) for the corresponding public key. CAs have to observe tightly controlled and audited procedures to create, manage and utilize a root, and to minimize exposure will normally use a root to issue intermediate certificates. These intermediates can then be used to issue their customers’ certificates.



Browsers are shipped with a built-in list of trusted roots. (These are roots from CAs who have passed the browser's stringent criteria for inclusion.) To verify a certificate, a browser will obtain a sequence of certificates, each one having signed the next certificate in the sequence, connecting the signing CA's root to the server's certificate.
This sequence of certificates is called a certification path. The path's root is called a trust anchor and the server's certificate is called the leaf or end entity certificate.

Path Construction
Oftentimes browsers have to consider multiple certification paths until they can find a valid one for a given certificate. Even though a path may contain certificates that "chain" together properly to a known anchor, the path itself may be rejected due to restrictions on path length, domain name, certificate usage, or policy.
Constructing and evaluating all possible paths is an expensive process performed for every new certificate a browser encounters. Browsers have implemented various optimizations to minimize the number of rejected candidate paths, but delving into such details is well beyond the scope of this article.

Path Validation
After a candidate certification path is constructed, browsers validate it using information contained in the certificates. A path is valid if browsers can cryptographically prove that, starting from a certificate directly signed by a trust anchor, each certificate's corresponding private key was used to issue the next one in the path, all the way down to the leaf certificate.

What is the process of verifying a certificate?

The steps that a browser performs to check the certificate are the following:

1. <u>The browser verifies the certificate’s integrity.</u>

The signature on the certificate can be verified using normal public key cryptography. If the signature is invalid, then the certificate is considered to be modified after its issuance and is therefore rejected.

2. <u>The browser verifies the certificate’s validity.</u>

A certificate’s validity period is the time interval during which the signing CA warrants that it will maintain information about its status. Browsers reject any certificates with a validity period ending before or starting after the date and time of the validation check.

3. <u>The browser checks the certificate’s revocation status.</u>

When a certificate is issued, it is expected to be in use for its entire validity period. Of course, various circumstances may cause a certificate to become invalid before it naturally expires.
Such circumstances might include a subject changing their name or a suspected compromise of their private key. In cases like this, a CA needs to revoke the corresponding certificate, and users also place their trust in a CAs to notify browsers of their certificates’ revocation status.
RFC 5280 recommends that CAs use revocation lists for this purpose.

Certificate Revocation Lists (CRL)

CAs periodically issue a signed, time-stamped list of revoked certificates called a certificate revocation list (CRL). CRLs are distributed in publicly available repositories, and browsers can acquire and consult the CA’s latest CRL when validating a certificate.
One flaw of this method is that the time granularity of revocation is limited to the CRL issue period. A browser will be notified of a revocation only after all currently issued CRLs are scheduled to be updated. Depending on the signing CA’s policy, this might take an hour, day or even up to a week.

Online Certificate Status Protocol (OCSP)

There are other alternative methods to acquiring revocation status information, with the most popular being the Online Certificate Status Protocol (OCSP).
Described in standard document RFC6960, OCSP lets a browser request a specific certificate’s revocation status from an online OCSP server (also called a responder). When properly configured, OCSP is much more immediate and avoids the CRL update-latency issue mentioned above. In addition, OCSP Stapling improves performance and speed.

4. <u>The browser verifies the issuer.</u>

Certificates are normally associated with two entities:
The issuer, which is the entity owning the signing key and
The subject, which refers to the owner of the public key that the certificate authenticates.
Browsers check that a certificate’s issuer field is the same as the subject field of the previous certificate in the path. For added security, most PKI implementations also verify that the issuer’s key is the same as the key that signed the current certificate. (Note that this is not true for the trust anchor, since roots are self-issued – i.e. they have the same issuer and subject.)

5. <u>The browser checks name constraints.</u>

A privately owned (but publicly trusted) intermediate CA with the appropriate name constraints can provide an organization with fine-grained control over certificate management and issuance. Certificates can be limited to a specific domain or domain tree (i.e. including subdomains) for a company or organization’s domain name. Name constraints are often used for intermediate CA certificates purchased from a publicly trusted CA to prevent the intermediate CA from issuing perfectly valid certificates for third-party domains (e.g. google.com).

6. The browser checks policy constraints.
A certificate policy is a legal document published by a CA, officially detailing the procedures they follow to issue and manage their certificates. CAs might issue a certificate under one or more policies, and links to these are included in each certificate issued so that relying parties can evaluate these policies before deciding to trust that certificate.
For legal and operational reasons, certificates can impose restrictions on which policies they can be subject to. If a certificate is found to contain critical policy constraints, browsers must validate them before proceeding. (However, critical policy constraints are rarely encountered in the real world and so will be disregarded for the rest of this article.)

7. The browser checks basic constraints (a.k.a. path length).
The X.509 v3 format allows issuers to define the maximum path length a certificate can support. This provides control over how far each certificate can be placed in a certification path. This is actually important – browsers used to disregard the certification path length until a researcher demonstrated, in a 2009 presentation, how he used his web site's leaf certificate to forge a valid certificate for a large e-commerce web site.

8. The browser verifies key usage.
The "key usage" extension states the purpose of the key contained in the certificate. Examples of such purposes include encipherment, signatures, certificate signing and so on. Browsers reject certificates violating their key usage constraints, such as encountering a server certificate with a key meant only for CRL signing.

9. The browser continues to process all remaining critical extensions.
After processing the extensions mentioned above, browsers proceed to validate all remaining extensions that the current certificate designates as critical, before moving on to the next. If a browser reaches a path's leaf certificate without error, then the path is accepted as valid. If any errors are produced the path is marked as invalid and a secure connection is not established.

Acknowledgments

[What Is an X.509 Certificate?]
[What Is a Certificate Authority (CA)?]
[Browsers and Certificate Validation]