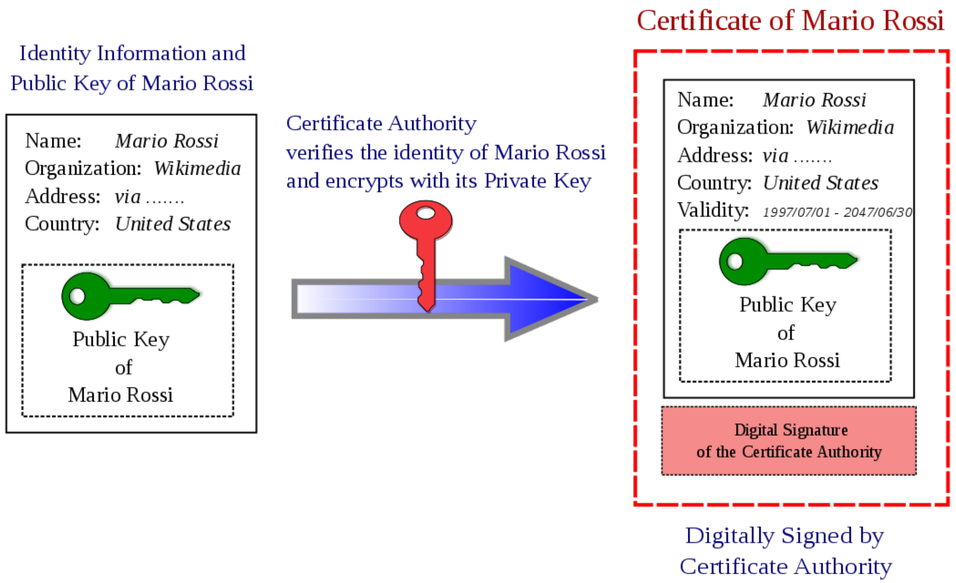
Certificates

Applied Cryptography

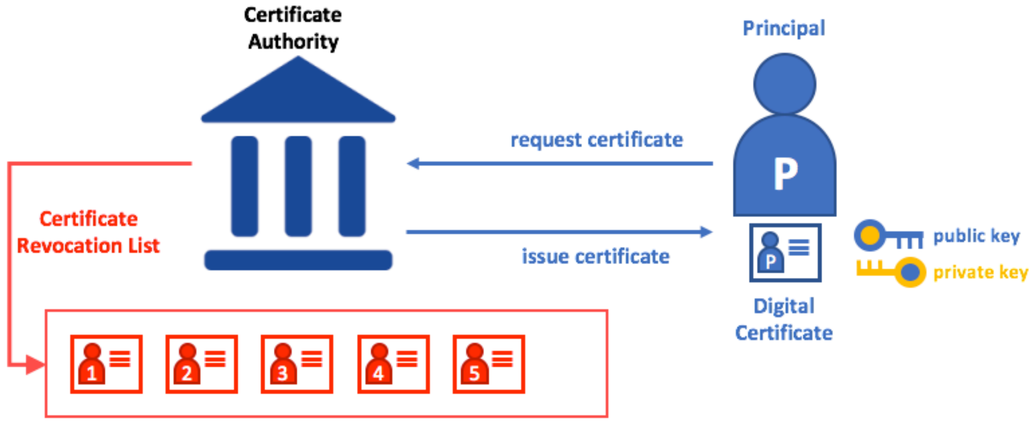
Public Key Certificates



https://en.wikipedia.org/wiki/Public_key_certificate

- A public key certificate (commonly just called a "certificate") is used to provide identification and trust by associating a public key with the subject identified by the certificate (the certificate holder)
- Certificates
  - contents are defined by the X.509 ITU standard
  - describe the identity of a subject, which contains several possible pieces of data including the Common Name (CN) which for a web certificate would be a domain name
  - can be tagged for various allowed uses, such as server certificates, client certificates, signing certificates, etc. with the difference being what the certificate is to be used for, and what is expected to be in the subject information stored in the certificate
  - include a signature from the CA that issued the certificate and guarantees it is valid
  - are tagged to show what identity verification has been done by the certificate issuer
  - have an expiry date and cannot be altered once issued, a new certificate must be issued when an old one expires in order to continue securing communications with that subject

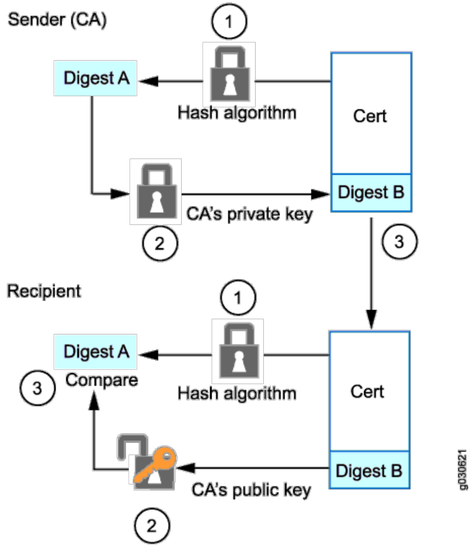
Authorities



https://hyperledger-fabric.readthedocs.io/en/release-2.0/identity/identity.html

1. Using a hash algorithm, the CA generates digest A from the certificate.
2. Using the private key, the CA encrypts digest A. The result is digest B, the digital signature.
3. The CA sends the digitally signed certificate to the person who requested it.

1. Using a hash algorithm, the recipient generates digest A from the certificate.
2. Using the CA's public key, the recipient decrypts digest B.
3. The recipient compares digest A with digest B. If they match, the recipient knows that the certificate has not been tampered with.



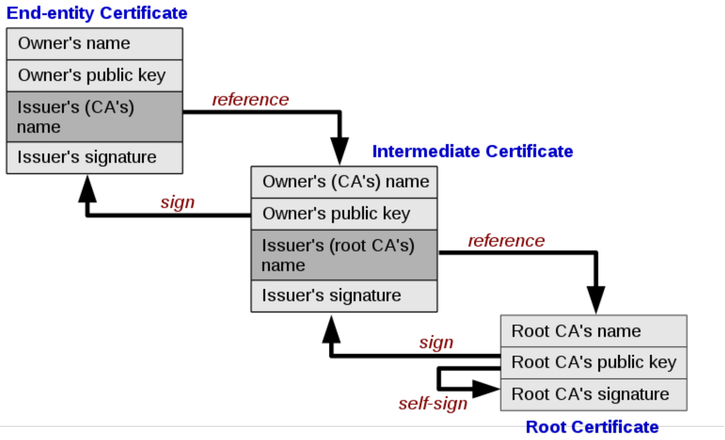
https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-digital-certificates-with-pki-overview.html

- Certificate Authorities (CAs) issue certificates in response to validated certificate signing requests
- Validation of a request may be done by a registration authority or may be done by the same organization that will issue the certificate to the requestor
- An issued certificate will be signed digitally by the CA issuing it so that the public can have confidence that the certificate is valid based on their trust of the CA

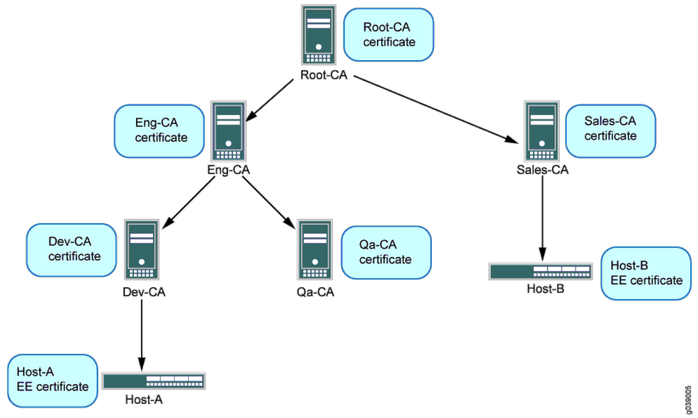
CA Certificates

- A set of CA certificates is included with the operating system
- Root certificates contain the identity information for the CA and the public key of the CA
- The public keys are used to verify CA signatures in other certificates
- CA root certificates are self-signed which is why they are pre-installed as trusted by the operating system and/or specific applications
- May use a chain of authority to increase confidence that the CA is trustworthy
- CAs may cross-sign each other's authority certificates

Chain of Trust



https://en.wikipedia.org/wiki/Public_key_certificate



https://www.juniper.net/documentation/en_US/junos/topics/example/security-pki-device-certificate-chain-configuring-mx.html

- CA may use a hierarchy of certificates
- Roots may sign intermediate authority certificates
- For a 2-level hierarchy, the intermediates sign the certificate requests that come from the registration authorities
- For a 3-level hierarchy, the intermediates sign further intermediates which sign certificate requests that come from the registration authorities

Certificate Store

- Root certificates and intermediate signing authority certificates are placed in a storage location on the system where they will be used called the certificate store
- Trusted self-signed certificates are placed in a certificate store (as in storage location, not as in buying certificates)
- The certificate store is created when the system is installed
- Where they are stored is operating-system specific, and can also be OS version specific
- openssl can be used to verify if a certificate is trusted

```
openssl verify -in certfilename.crt
```

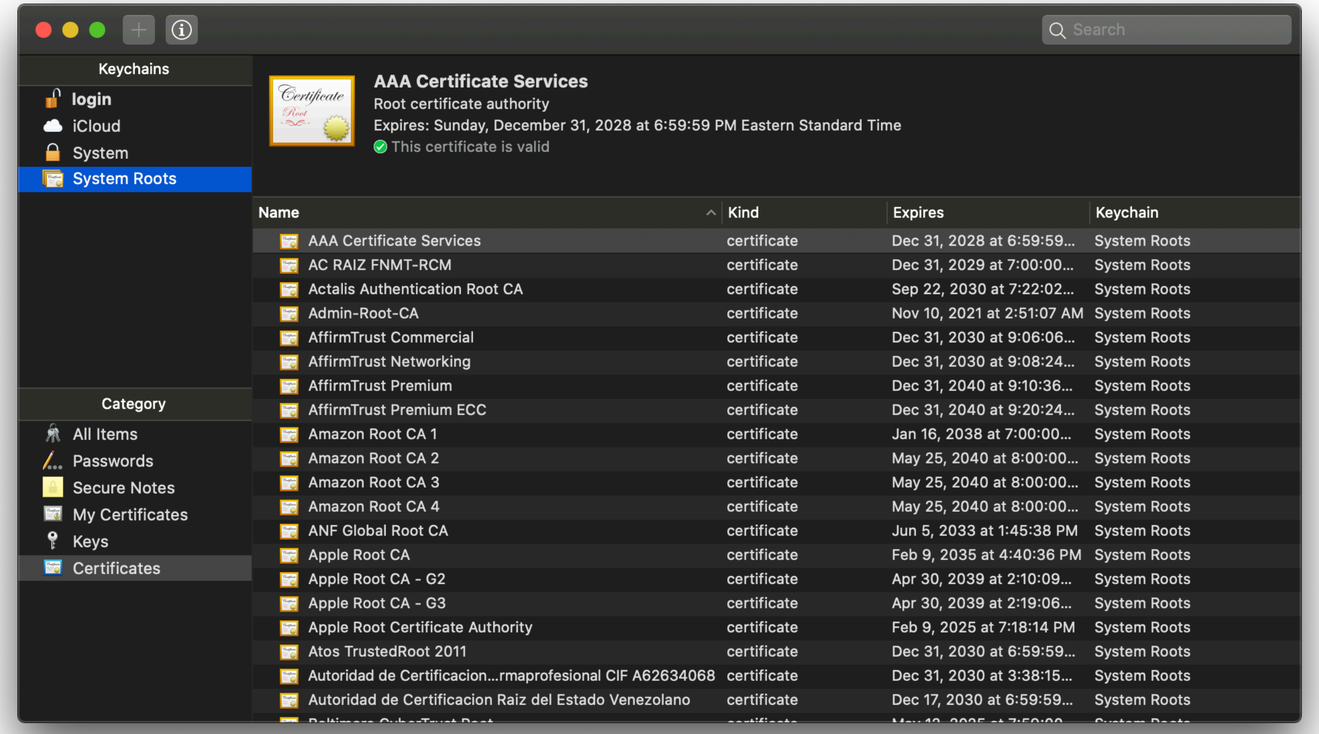
Linux Certificate Store

- Ubuntu installs them as part of the ca-certificates package
- The certificates are kept in /usr/share/ca-certificates
- Certificate files are stored in subdirectories below that to organize them based on how they were approved to be included
- Certificates can be added or removed at your discretion by adding or removing the files and managing the list of enabled certificates with dpkg-reconfigure ca-certificates
- Debian-derived distros use /etc/ssl (/etc/pki/tls for Redhat-derived distros) to hold certificate and key files
- There are links to the root certificates placed in a certs subdirectory, and you can add non-root certificates that are trusted to the certs subdirectory
- Private keys for use on a particular system are placed in a private subdirectory

Windows Certificate Store

- Windows stores them in multiple places dependent on multiple factors but there is a set of them stored in the registry
- If you want to add or remove a root certificate in Windows, you may have to do more than add it to, or remove it from, the registry
- The Windows certificate store is designed to be managed using GUI certificate tools, and is a bit tricky to try to work with, particularly if you are not just using a simple desktop machine
- There are multiple stores, and how they get used depends on the os version, the system config, active directory use, what browser you are using, etc.
- See https://docs.microsoft.com/en-us/windows/win32/seccrypto/system-store-locations for more information

MacOS Certificate Store



- MacOS has a single place where root certificates are stored like Linux, but stores them in keychain files, which are not intended to be directly manipulated
- The MacOS keychain utility is used to view, modify, and extract certificates from the certificate store
- You can drag and drop certificates in and out of the keychain utility

PKI Certificates

- Certificates themselves are not OS-specific, and openssl is a good cross-platform utility for working with them
- openssl has the x509 subcommand which allows us to work with certificate files
- The verify subcommand can be used to check trust for a certificate
- openssl's x509 command also can be used during the creation of certificates

```
cat certfile | openssl x509 -noout -pubkey
cat certfile | openssl x509 -noout -pubkey -subject
cat certfile | openssl x509 -noout -pubkey -subject -startdate -enddate
cat certfile | openssl x509 -noout -text

openssl s_client -connect api.stackexchange.com:443 </dev/null | openssl x509 -text -noout

sudo openssl s_server -cert /etc/ssl/certs/ssl-cert-snakeoil.pem -key /etc/ssl/private/ssl-cert-snakeoil.key

openssl s_client -connect server:4433 <<<"GET /"
```

Certificate Revocation

- Certificates can be revoked or held by the issuing CA
  - Revoked certificates are no longer valid
  - Held certificates may be released and become valid again
- Revoked certificates are added to a CRL by the CA that issued the certificate
- CRLs are published at the discretion of the issuing CA
- There are also ARLs, to revoke CAs which also invalidates every certificate signed by that CA
- CRL checking is supported but not enabled in most of the major browsers and browser suppliers support OCSP as an alternative

CRL

- Certificate Revocation List

ARL

- Authority Revocation List

OCSP

- Online Certificate Status Protocol

Online Certificate Status Protocol

- A lower-overhead request/response protocol, compared to querying CRLs used by web client programs, to check if a certificate is still valid
- Not all browsers use this - all the major ones except Chrome support it although it may not be enabled
- Not all certificate suppliers respond to this, some delegate authority to respond to third parties, and some only use it for EV certificates
- It is not encrypted so it is susceptible to data harvesting and man-in-the-middle attacks, as well as replay attacks
- Many clients simply ignore unresponsive OCSP responders

OCSP Stapling

- Both CRL and OCSP validation techniques involve the potential to generate large quantities of traffic requiring real-time responses
- A solution for this is to have the web server using a certificate make regular OCSP queries and save the responses
- Then if a client issues a TLS request with the Certificate Status Request extension enabled, the web server just includes its recent response with the request saving the web client from making separate OCSP requests
- Other than the potential for introducing delays in revoking certificates due to server caching of OCSP responses, this mostly solves the traffic problem
- Major web server software suppliers pretty much all have some form of stapling in place

PKI Certificate Types

- Web certificates are most commonly used to authenticate domain names
- Certificates can be self-signed, meaning they are not trusted to authenticate, unless a user has explicitly chosen to trust them, or the system administrator has installed them as a trusted certificate
- Certificates can be CA signed, in which case they are typically validated by one of several methods by the CA
  - DV certificates only indicate that the certificate holder has control of either the domain's DNS, the domain name's web server, or both
  - OV certificates indicate the CA has done diligence in verifying that the certificate was applied for by the organization described in the certificate
  - IV certificates indicate the CA has done diligence in verifying that the certificate was applied for by the individual identified in the certificate - these are not normally used for web servers but are used for personal messaging
  - EV certificates require more verification on the part of the CA to ensure the certificate's identity does indeed belong to the certificate holder

DV

- Domain Verified

OV/IV

- Organization/ Individual Verified

EV

- More Diligently Verified Identity

Obtaining Certificates

- Traditionally a manual process involving someone responsible for a domain name making a request for a certificate from a CA
- The requestor creates a CSR file and signs it with their own private key
- The requestor gathers documentation to demonstrate they are who they claim to be and that they have the authority to manage certificates for the domain name
- The requestor sends the CSR file, the documentation, and a cheque to the CA
- They all twiddle their thumbs for a significant time
- The requestor receives the certificate file from the CA and can now install it where their service software can find it

CSR

- Certificate Signing Request

ACME

- On the internet, the most common use of PKI certificates is to authenticate web servers' domain names
- The ACME protocol provides an automated method to verify domain control and issue certificates based on that verification
- There are 2 ways normally used to demonstrate domain control
    1. Verify control of DNS by adding a DNS record
    2. Verify control of web server for domain by adding a retrievable document to the web server
- See https://tools.ietf.org/pdf/rfc8555.pdf for an excellent discussion of the ACME protocol and how it fits in with the web's PKI

ACME

- Automatic Certificate Management Environment

Letsencrypt

- Letsencrypt is a free certificate authority which can be used to obtain DV certificates with the ACME protocol
- It bases certificate signing decisions on whether the applicant can demonstrate control of the domain's web service or DNS
- The process is fully automated at the letsencrypt server and there are many scripts and programs that can apply for, receive, and install letsencrypt certificates
- Firmware suppliers for routers and such are starting to include letsencrypt client capabilities
- When applying for a letsencrypt certificate, the letsencrypt servers provide you with data for a DNS TXT record, or content for a file to place on the domain web server
- When you ask for the certificate to be issued, the letsencrypt servers verify the web file or DNS TXT record are live and that is all the proof they require to believe that you control the domain name for the certificate
- Automation software often only provides the web method and assumes the web server is on the embedded device

Private CA for PKI

- There are a number of both free and paid software packages that can provide the functionality of a private CA
- They create a root private keypair, then a self-signed root certificate
- They can then either sign CSRs using their root private key, or create a keypair and CSR and sign it, which supplies the private key and matching certificate as an intermediate authority
- Any machines which should be part of the private PKI will need to add the private root certificate to their trusted certificate store
- VPN packages typically include a private CA capability
- Easy-rsa, CA.pl, and openssl can all be used to create a private CA