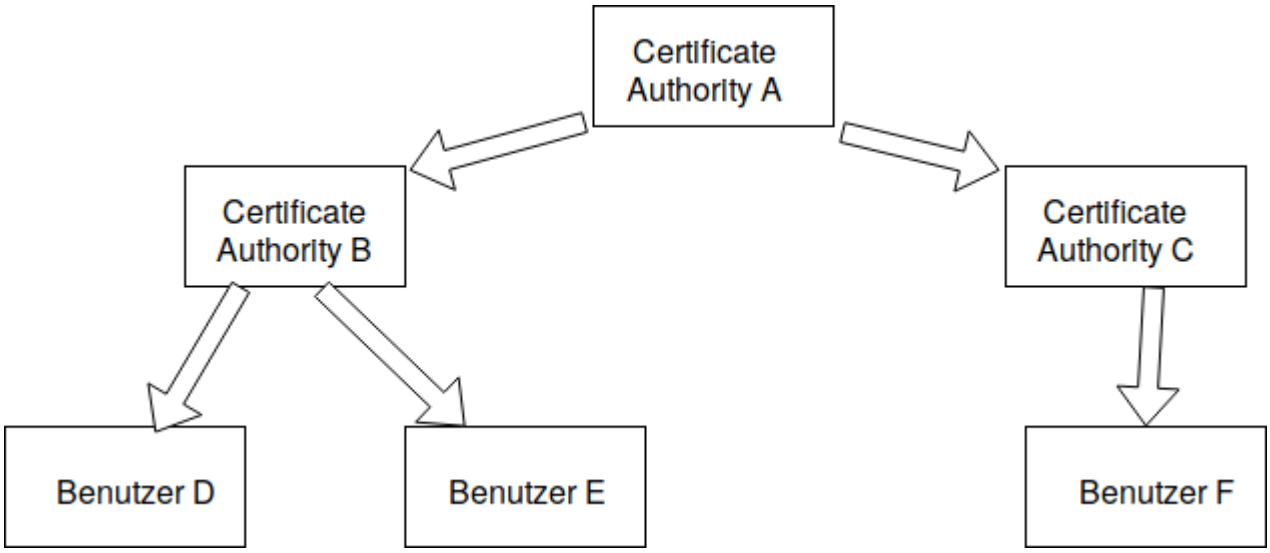
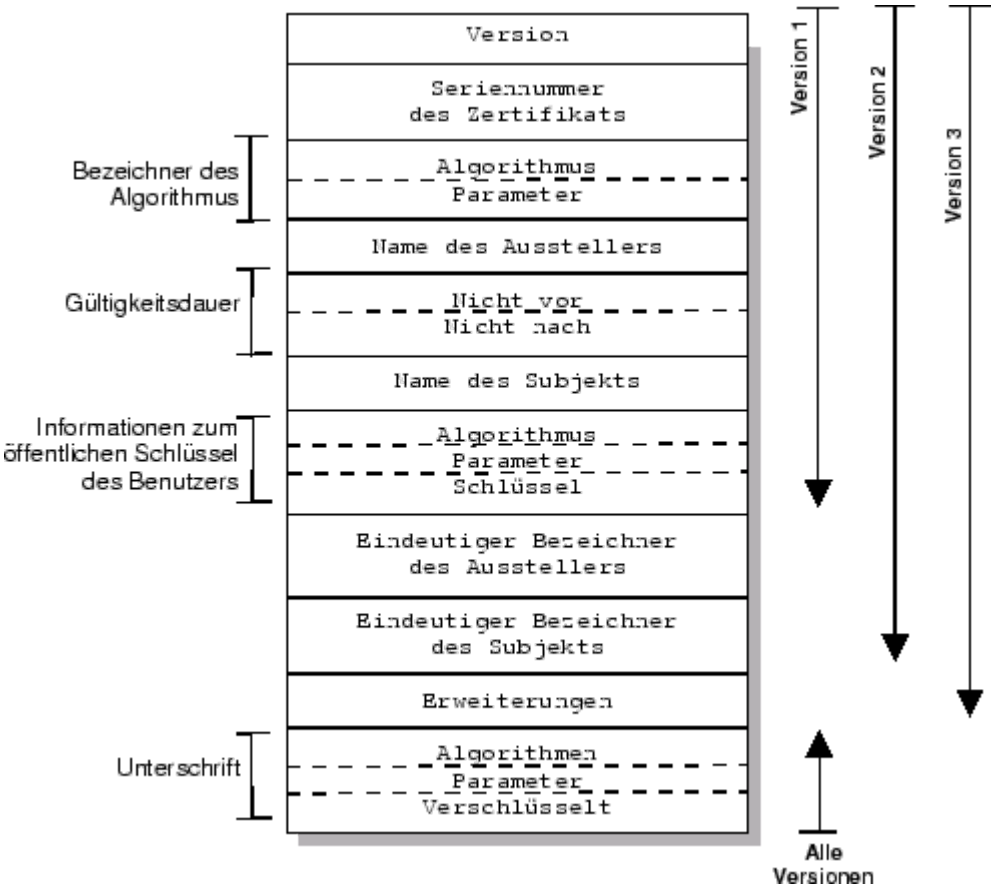
How do X509 Certificates work and how are they used in Internet Applications

Zertifikat

- Datenstruktur, die Informationen über dessen Inhaber und einen öffentlichen Schlüssel enthält.
- Aussteller (Certificate Authority) bestätigt die Identität des Inhabers durch das ausstellen.
- Revocation-Liste zum wiederrufen von Zertifikaten



Aufbau Zertifikat



- Version: Versionsnummer des Zertifikates
- Seriennummer: bildet zusammen mit dem Ausstellernamen ein eindeutiges Kennzeichen für jedes Zertifikat
- Name der Aussteller: Namen der Zertifizierungsstelle (CA)
- Name des Subjekts: Eigentümer des Zertifikats

Außerdem noch ein Gültigkeitsintervall des Zertifikates

Dateiformate

- PEM
- DER
- PFX oder P12
- P7B
- P7C
- CSR
- CRL

Dateiformat PEM

wird von den meisten Tools am besten unterstützt

häufig von Zertifizierungsstellen benutzt

Der Name Privacy Enhanced Mail stammt aus einer gescheiterten Methode für sichere E-Mails

Base64 kodiert

weitere Dateiendungen .cert, .cer, .crt, .key

Dateiformat DER

Distinguished Encoding Rules

binäre Form der Base64 Kodierten .pem Datei

Dateiformate PFX/P12

binäres Format

Datei lässt sich Passwordgeschützt speichern

oft zum Import/Export von Zertifikaten und privaten Schlüsseln unter Windows verwendet

Dateiformat CRL

Certificate Revocation List

Zertifikate können für ungültig erklärt werden

z.B: wenn der private Schlüssel kompromitiert wurde

wird von der CA verwaltet

Teilnehmer rufen die CRL regelmäßig ab und können damit die Gültigkeit der Zertifikate anderer Teilnehmer überprüfen

Wo wird es benutzt ?

- TLS

Nachfolger von SSL

gewähleistet damit sichere Datenübertragung im Internet

umgangssprachlich wird auch SSL Zertifikat gesagt

Wo wird es außerdem benutzt

- E-Mail Verschlüsselung/Signierung

Erstellung Zertifikat

- Selbstsigniertes Zertifikat
- Signiertes Zertifikat von einer CA

Selbstsigniertes Zertifikat

- Schlüsselpaar erzeugen
- Certificate Request (CSR) erzeugen
- Certificate Request signieren

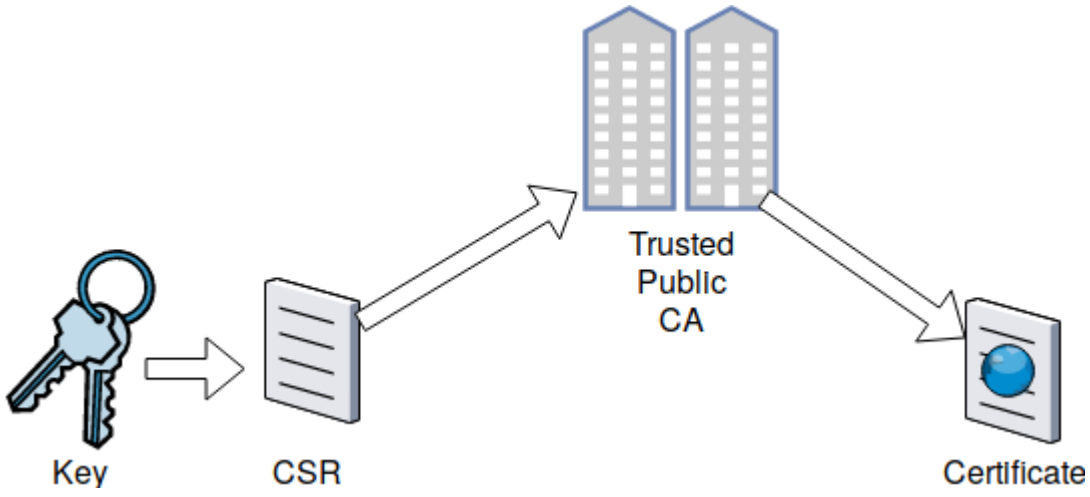
In [1]:

```
openssl req -x509 -out myCert.pem -newkey rsa:2048 -keyout myKey.pem -nodes
 -sha256 -days 1000 -subj "/C=DE/ST=Berlin/L=Berlin/O=HTW/OU=AI/CN=example.
com"
```

```
Generating a 2048 bit RSA private key
................+++
..............................+++
writing new private key to 'myKey.pem'
-----
```

- myKey.pem: enthält den privaten RSA-Schlüssel
- myCert.pem: enthält das selbstsignierte Zertifikat

Erstellung Zertifikat mithilfe CA



Rolle Client

Erstellen eines RSA Key mit 4096 Bit

In [2]:

```
openssl genrsa -out example.com.key 4096
```

```
Generating RSA private key, 4096 bit long modulus
.....................................................................
++
.....................................................................
++
e is 65537 (0x10001)
```

Erstellen eines Certificate Signing Requests dies sind Anträge

Dieser CSR kann nun selbst unterschrieben werden oder von einer Certificate Authority (CA)

In [3]:

```
openssl req -new -sha256 -key example.com.key -out example.com.csr -subj "/
C=DE/ST=Berlin/L=Berlin/O=HTW/OU=AI/CN=example.com"
```

Rolle CA

Aufgaben:

- Erzeugen von Benutzerzertifikaten durch Signieren von Certificate Requests
- Revocation-Liste verwalten

erstellen des RSA Key für die CA

In [4]:

```
openssl genrsa -out CA.key 4096
```

```
Generating RSA private key, 4096 bit long modulus
.........................++
.............................++
e is 65537 (0x10001)
```

Erstellen des CSR für die CA

In [5]:

```
openssl req -new -sha256 -key CA.key -out CA.csr -subj "/C=DE/ST=Berlin/L=B
erlin/O=HTW/OU=CA/CN=CA.com"
```

Erstellen einer Konfigurations Datei für die CA

In [12]:

```
echo "[ ca ]
# X509 extensions for a ca
keyUsage                = critical, cRLSign, keyCertSign
basicConstraints        = CA:TRUE, pathlen:0
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer:always

[ server ]
# X509 extensions for a server
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
basicConstraints        = critical,CA:FALSE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer:always" > x509.ext
```

Erstellen des Zertifikates für die CA

In [7]:

```
openssl x509 -req -sha256 -extfile x509.ext -extensions ca -in CA.csr -sign
key CA.key -days 1095 -out CA.pem
```

```
Signature ok
subject=/C=DE/ST=Berlin/L=Berlin/O=HTW/OU=CA/CN=CA.com
Getting Private key
```

Erstellen des Zertifikates für den Client

In [8]:

```
openssl x509 -req -sha256 -CA CA.pem -CAkey CA.key -days 730 -CAcreateseria
l -CAserial CA.srl -extfile x509.ext -extensions server -in example.com.csr
 -out example.com.pem
```

```
Signature ok
subject=/C=DE/ST=Berlin/L=Berlin/O=HTW/OU=AI/CN=example.com
Getting CA Private Key
```

Verifizierung

In [10]:

```
openssl verify -CAfile CA.pem example.com.pem
```

```
example.com.pem: OK
```

Übersprüfen der md5 Werte

In [9]:

```
openssl rsa -noout -modulus -in example.com.key| openssl md5
openssl x509 -noout -modulus -in example.com.pem| openssl md5
```

```
(stdin)= 12a4101301c7cd92a27039ec8e6aa28e
(stdin)= 12a4101301c7cd92a27039ec8e6aa28e
```

Noch Fragen ?

Folien: https://github.com/mjohnki/talks (https://github.com/mjohnki/talks)