Einführung in OpenSSL und X.509-Zertifikate

Martin Kaiser
http://www.kaiser.cx/

Über mich

- Elektrotechnik-Studium Uni Karlsruhe
- System-Ingenieur UNIX und IP-Netze (2001-2003)
- Embedded Software-Entwicklung Digital-TV (seit 2003)
  - Schwerpunkt Pay-TV (DVB Common Interface)
  - Pay-TV Standard CI+

Inhalt

- OpenSSL
- Zertifikate
  - Einsatzbereich
  - Aufbau
  - Codierung
- Erstellen und Verifizieren von Zertifikaten
- Anwendungsbeispiel
- Programmierung

OpenSSL

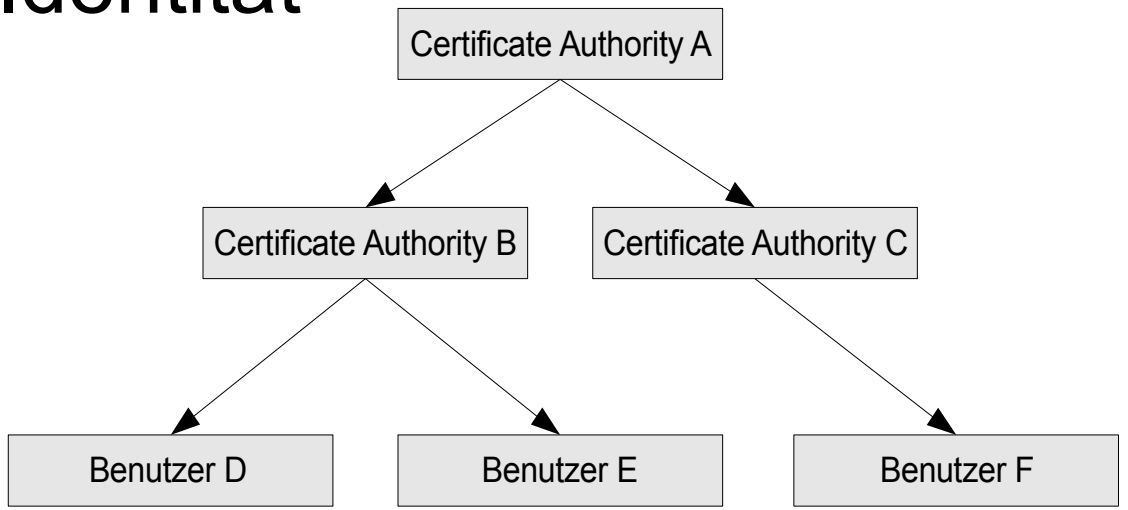
- Implementierung der SSL/TLS Protokolle
- Open source, Lizenz nicht kompatibel zur GPL
- libssl, libcrypto
  - Symmetrische und Public-Key Verschlüsselungsverfahren
  - Hashes
  - X.509 Zertifikate
  - ASN.1
  - BIO
  - BIGNUM
  - Primzahlen
  - ...
- Kommandozeilentools

Public-Key Kryptographie

- Schlüsselpaar
  - öffentlicher Schlüssel
  - privater Schlüssel
- Verschlüsselung
  - A verschlüsselt mit Bs öffentlichem Schlüssel
  - B entschlüsselt mit seinem privaten Schlüssel
- Signatur
  - A signiert mit seinem privaten Schlüssel
  - B verifiziert mit As öffentlichem Schlüssel

Zertifikat

- Datenstruktur, die einen öffentlichen Schlüssel mit Informationen über dessen Inhaber verknüpft
- Aussteller signiert das Zertifikat, bestätigt damit die Identität des Inhabers
- Revocation-Liste, um Zertifikate zu widerrufen



Aufbau eines X.509 Zertifikats (I)

- Version
- Seriennummer
- Signatur-Algorithmus
- Issuer
- Gültigkeit (von, bis)
- Subject

eindeutige Nummer, vom Aussteller vergeben

begrenzte zeitliche Gültigkeit

Namen entsprechend X.501

```
C  = DE,
ST = Hessen,
L  = Frankfurt,
O  = Test GmbH,
OU = Personalabteilung,
CN = Martin Kaiser
```

Aufbau eines X.509 Zertifikats (II)

- Subject Public Key Info
  - Public Key Algorithmus
  - Subject Public Key
- Eindeutige ID von Aussteller und Inhaber (optional)
- Erweiterungen (optional)
- Signatur-Algorithmus
- Signatur

meistens RSA

Aufbau
- ID
- Flag: kritisch?
- Inhalt

Beispiele
- Basic constraints
- Key usage
- Private Erweiterungen

über alle Felder von Version bis Erweiterungen

Textdarstellung

```
openssl x509 -in myCert.pem -noout -text
```

Codierung, Speicherung

- Zertifikat wird codiert in ASN.1
  - Tag (Datentyp), Length, Value
  - Datentyp z.B. Sequenz, Objekt, Bitstring, Integer
- ASN.1 Darstellung des Signatur-Algorithmus

```
askja:/etc/ssl/guug# openssl asn1parse -in ./myCert.pem
...
956:d=1 hl=2 l=13 cons: SEQUENCE
958:d=2 hl=2 l= 9 prim: OBJECT :sha256WithRSAEncryption
969:d=2 hl=2 l= 0 prim: NULL
```

- Speicherung
  - direkt als ASN.1 Bytefolge (DER) oder als PEM

Selbstsigniertes Zertifikat

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

```
openssl req -x509 -out myCert.pem \
  -newkey rsa:2048 -keyout myKey.pem \
  -nodes -sha256 -days 1000
```

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

Certificate Authority (CA)

- Aufgaben
  - Erzeugen von Benutzerzertifikaten durch Signieren von Certificate Requests
  - Revocation-Liste
- Grundkonfiguration
  - *openssl.cnf*: Algorithmen, Gültigkeitsdauer
  - Root-Zertifikat, privater Schlüssel

```
askja:/etc/ssl/guug# mkdir newcerts private
askja:/etc/ssl/guug# chmod 700 private
askja:/etc/ssl/guug# touch index.txt
askja:/etc/ssl/guug# echo 1001 > serial
```

CA (II)

- Benutzer erstellt seinen CSR

```
openssl req -new -keyout clientKey.pem \
  -out clientCsr.pem
```

- CA signiert den Benutzer-CSR

```
openssl ca -in clientCsr.pem -notext
```

  - Benutzerzertifikat *newcerts/<serial>.pem*
  - Nummer in der Datei *serial* wird inkrementiert
  - Neuer Eintrag in *index.txt*

```
V  101113173819Z  1007  unknown
   /C=DE/ST=Hessen/L=...
```

Revocation-Liste, Verifikation

- Zertifikat auf die Revocation-Liste setzen

```
openssl ca -revoke newcerts/1007.pem
```

  - *index.txt:* Flag R, Widerrufszeitpunkt

- Revocation-Liste erzeugen aus *index.txt*

```
openssl ca -gencrl -out myList.crl
```

- Zertifikat verifizieren

```
openssl verify -crl_check myCert.pem
```

  - */etc/ssl/certs/*: Root-Zertifikate, Revocation-Listen
  - *c_rehash*: erzeugt Symlinks für **.pem*, **.crt*

Apache (I)

- Apache Webserver
  - Apache-Modul *mod_ssl* nutzt OpenSSL
  - X.509 Zertifikate für SSL-Verbindungen (HTTPS)
    - Server- und Client-Zertifikate
- Server-Zertifikat

```
SSLCertificateFile /path/to/apacheServerCert.pem
SSLCertificateKeyFile /path/to/apacheServerKey.pem
```

  - Browser verifiziert das Zertifikat des Webservers
    - Browser enthält eine Anzahl von Root-Zertifikaten
    - Import des eigenen Root-Zertifikats zur Verifikation von selbst erstellten Zertifikaten

Apache (II)

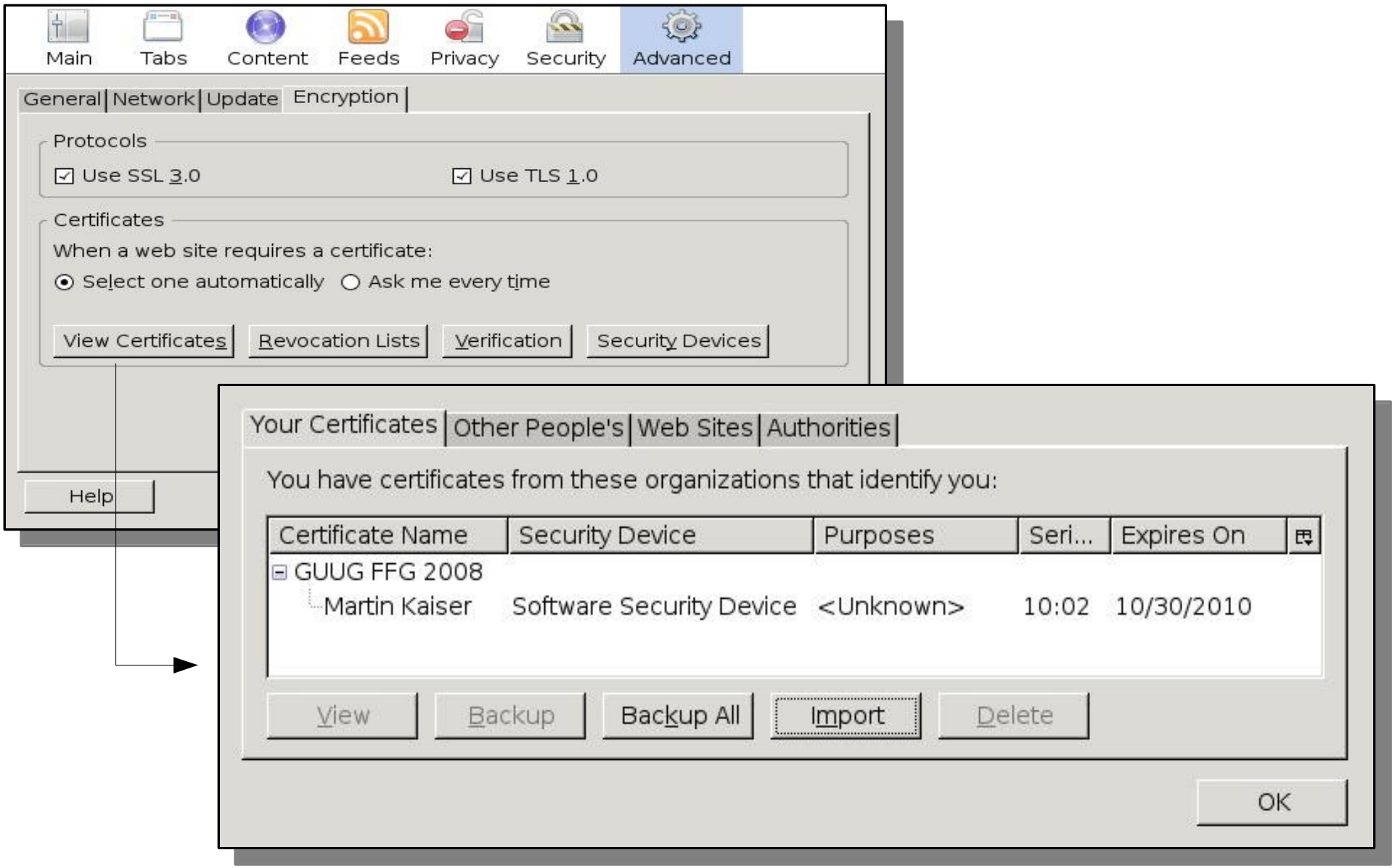
- Client-Zertifikat, Revocation-Liste
  - nur Clients mit gültigem Zertifikat dürfen zugreifen

```
SSLVerifyClient require

SSLCACertificateFile /path/to/myCaCert.pem

SSLVerifyDepth 1

SSLCARevocationFile /path/to/myCrl.pem
```

  - Import von Client-Zertifikat und Private Key in den Browser als PKCS12

```
openssl pkcs12 -export -clcerts \
  -in myCert.pem -inkey myKey.pem \
  -out myClient.p12
```

Import von Zertifikaten in Firefox



Programmierung (I)

- Beispiel: Verifikation eines X.509-Zertifikats
- Struktur *X509*: ein Zertifikat

```
typedef struct x509_st {
  X509_CINF        *cert_info;
  X509_ALGOR       *sig_alg;
  ASN1_BIT_STRING *signature;
  ...
} X509;
```

- Struktur *X509_STORE*: Parameter für eine Verifikation

```
typedef struct x509_store_st {
  STACK_OF(X509_OBJECT) *objs; /* cert, crl, pubkey */
  X509_VERIFY_PARAM *param;    /* depth, flags, ... */
  int (*verify_cb)(int ok,X509_STORE_CTX *ctx);
  ...
} X509_STORE;
```

Programmierung (II)

- Struktur *X509_STORE_CTX*: Store + Zertifikate

```
typedef struct x509_store_ctx_st {
  X509_STORE *ctx;
  X509 *cert;
  STACK_OF(X509) *untrusted;
  ...
  X509 *current_cert;
  X509 *current_issuer;
} X509_STORE_CTX;
```

- Verifikation eines Zertifikats

```
int X509_verify_cert(X509_STORE_CTX *ctx)
```

- Callback-Funktion für jedes zu prüfende Zertifikat

```
int myCb(int ok, X509_STORE_CTX *ctx)
```

  - Rückgabewert ist der neue Fehlerzustand

Danke für Ihr Interesse

Fragen?

Folien zum Download -> http://www.kaiser.cx/