OpenSSL in der Praxis

UUGRN FIXME-Treffen
6. März 2009

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

- Einführung in OpenSSL
- Grundlegende Funktionen
- Symmetrische Verschlüsselungsverfahren
- Hashes
- Public-Key Verfahren
- Zertifikate
- SSL/TLS
- Programmierung mit OpenSSL

OpenSSL

- Implementierung der SSL / TLS Protokolle
- BSD-Lizenz mit Advertising Clause,
  nicht GPL-kompatibel
- Bibliotheken libssl, libcrypto
- Kommandozeilentools
  openssl <Funktion> <Parameter>
- Interfaces zu allen möglichen Programmiersprachen

OpenSSL (II)

- apache, exim, opera, ... verwenden libssl und libcrypto
  openssh verwendet libcrypto
- unterstützt Unix, Windows, OS/2, QNX, vxWorks, ...
  - eigenes Configure-Skript, kein autoconf
- Assembler-Teile
- Engines
  - Interface zu Crypto-Hardware
- FIPS 140-2 Zertifizierung

Funktionsumfang

- Zufallszahlen
- BIGNUM
- Primzahlen
- BIO
- ASN.1
- Symmetrische und Public-Key Verschlüsselung
- Hashes, MACs
- X.509 Zertifikate
- SSL, TLS, DTLS
- S/MIME
- PKCS12

Grundlegende Funktionen

- Konfigdatei finden

```
mk@askja:~$ openssl version -d
OPENSSLDIR: "/usr/lib/ssl"
```

- einfacher Benchmark

```
mk@askja:~$ openssl speed md5
Doing md5 for 3s on 16 size blocks: 3186606 md5's in 2.99s
Doing md5 for 3s on 64 size blocks: 2703175 md5's in 2.99s
[...]
```

- Zufallszahlen

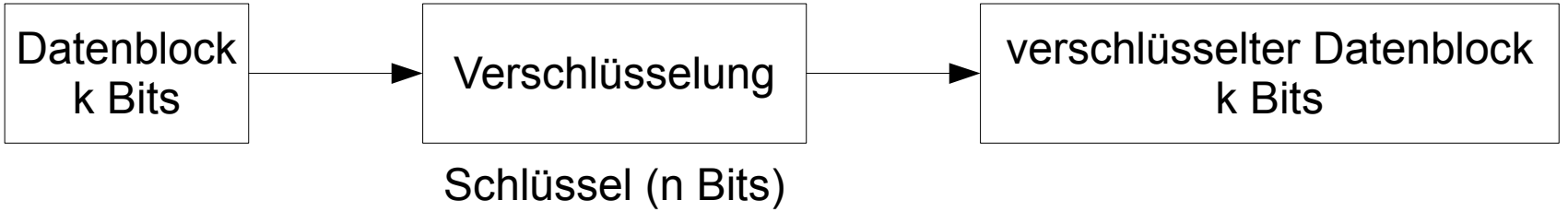
```
mk@askja:~$ openssl rand 20 -base64
FHZFFyXDfbx8vXb83N5RcHIY/NY=
```

- verschlüsselte Passwörter (crypt oder MD5)

```
mk@askja:~$ openssl passwd -1 hallo
$1$MI59Eu39$47oueKwEWWdpRSY3qUQ1O1
```

Symmetrische Verschlüsselung

- symmetrisch = derselbe Schlüssel zum Ver- und Entschlüsseln
  (nicht exakt dieselbe Berechnung)
- Blockverschlüsselung



  - Blocklänge k bits , Schlüssellänge n Bits
  - Padding
  - Verkettung der Eingangsdatenblöcke
    - ECB
    - CBC, braucht einen Init-Vektor
  - Beispiele: DES, AES, IDEA
- Stromverschlüsselung

Beispiele

- Datei verschlüsseln

```
openssl enc -aes-256-cbc -a -salt \
   -in file.txt -out file.enc -pass pass:hallo
```

  - Schlüssel und IV werden aus Passwort und salt generiert

- Datei entschlüsseln

```
openssl enc -d -aes-256-cbc -a -in file.txt
```

  - Eingabeaufforderung für das Passwort

- verschlüsseltes tar-Archiv

```
tar cvz mydir | openssl enc \
                -aes128 -salt -out mydir.tgz.enc
```

- entschlüsseln

```
openssl enc -d -aes128 -in mydir.tgz.enc | tar xvzf -
```

Hash-Funktionen

- Beispiele: MD5, SHA1, SHA256
- SHA3 contest, ähnlich AES
- eine Anwendung: Prüfsumme einer Datei berechnen

```
mk@askja:~$ openssl dgst -sha1 file.txt
SHA1(file.txt)= e1a9de5dc7f97cc18cade55d04ea0b3dd52ac4f0
```

Public-Key Kryptographie

- Schlüsselpaar: öffentlicher und 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
- bekanntestes Verfahren ist RSA
  - unterstützt Verschlüsselung und Signatur
- Public-Key Verfahren deutlich langsamer als symmetrische Verfahren

RSA mit OpenSSL

- privaten Schlüssel erzeugen

```
openssl genrsa -out test1.key 2048
```

  - zusätzlich mit Passphrase verschlüsseln

```
openssl genrsa -out test1.key -aes128 2048
```

- Textdarstellung des privaten Schlüssels

```
openssl rsa -in test1.key -noout -text
```

- öffentlichen Schlüssel aus privatem Schlüssel erzeugen

```
openssl rsa -in test1.key -pubout -out test1.pub \
  -passin pass:hallo
```

RSA mit OpenSSL (II)

- Signieren

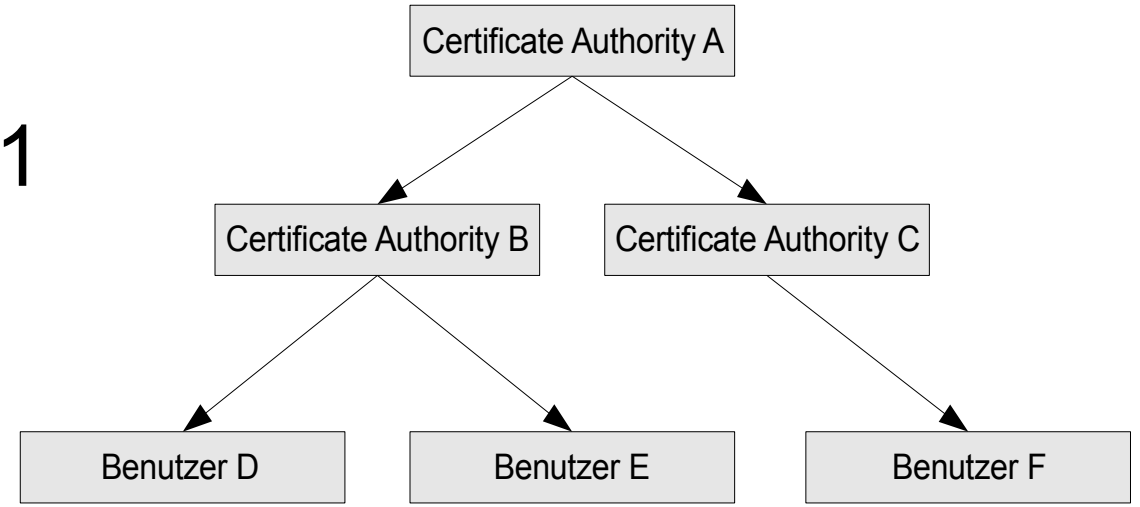
```
openssl rsautl -in data1.txt -inkey test1.key \
   -pass pass:hallo -sign -out data1.sig
```

- Signatur prüfen

```
openssl rsautl -in data1.sig -inkey test1.pub \
   -pubin -verify
```

X.509 Zertifikate

- Zertifikat = Datenstruktur, die einen öffentlichen Schlüssel (meistens RSA) mit Informationen über dessen Inhaber verknüpft
- Aussteller signiert das Zertifikat, bestätigt damit die Identität des Inhabers
- Codierung in ASN.1
- Speicherung im PEM oder DER-Format



Beispiele zu Zertifikaten

- Textdarstellung eines Zertifikats

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

- selbst-signiertes Zertifikat und privaten Schlüssel erstellen

```
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

Beispiele zu Zertifikaten (II)

- Zertifikat-Request erstellen und signieren

```
mk@askja:~$ openssl req -new -out ./testReq.pem \
                -newkey rsa:2048 -keyout test1.key

mk@askja:~$ echo 1000 > serial.txt

mk@askja:~$ openssl x509 -req -in ./testReq.pem \
            -out testCert.pem \
            -CA ./myRootCert.pem -CAkey ./myRootKey.pem \
            -CAserial ./serial.txt
```

- Zertifikat verifizieren

```
openssl verify -CAfile ./myRootCert.pem ./testCert.pem
```

SSL / TLS

- die häufigste Anwendung für X.509-Zertifikate
- Protokoll zur gesicherten Kommunikation zwischen zwei Partnern
  - Authentifizierung der Gegenseite
  - verschlüsselte und signierte Datenübertragung
  - Unterstützung für „Sessions“
    - vereinfachter Aufbau weiterer Verbindungen mit bereits ausgehandelten Parametern
  - SSLv3 wurde von Netscape spezifiziert
  - TLS 1.0 ist definiert in RFC2246

SSL auf der Kommandozeile

- s_client, s_server: einfacher Client, Server für SSL/TLS, sehr nützlich zum Testen
- Webserver mit Status-Seite
  - Port 4433
  - Client-Zertifikat erforderlich, maximale Pfadlänge 2

```
openssl s_server -tls1 \
  -cert ./serverCert.pem -key serverKey.pem \
  -Verify 2 -CApath /etc/ssl/certs -www
```

- Verbindung zu einem POP3-SSL-Server

```
openssl s_client -connect mail.server:995
```

  - Umschaltung auf SSL mit STARTTLS

```
openssl s_client -connect mail.server:110 -starttls pop3
```

SSL auf der Kommandozeile (II)

- Benchmark für SSL-Verbindungsaufbau
  - 30s möglichst viele Verbindungen aufbauen, mit jeweils neuer Session
  - 30s mit gecachter Session

```
openssl s_time -connect <servername>:443
```

Firefox und X.509-Zertifikate

- Firefox verwendet nicht OpenSSL
  - deshalb auch nicht die Root-CA-Zertifikate in OPENSSLDIR/certs/
- Firefox verwaltet folgende Zertifikate
  - Root-CA-Zertifikate, gegen die ein Server-Zertifikat authentifiziert wird
  - Client-Zertifikate mit zugehörigem privaten Schlüssel, um sich bei Websites zu authentifizieren, die ein Client-Zertifikat verlangen
    - Import im PKCS12-Format

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

Programmierung mit OpenSSL

- Beispiel: ein einfacher Client, der eine html-Seite liest
- Datentypen
  - SSL_CTX : Verbindungsparameter
  - SSL : eine SSL-Verbindung
  - BIO : eine Datenverbindung (TCP/IP, shared mem, file, filter, ...)
- Verbindungsaufbau
  - SSL_CTX erzeugen, Parameter setzen
  - SSL daraus ableiten
  - BIO-Verbindung aufbauen
  - SSL mit BIO verknüpfen
  - SSL-Handshake
  - read() und write() auf den SSL

Danke für Ihr Interesse

Fragen?

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