OpenSSL

What is it?

OpenSSL is an application that is able to provide (by setting parameters on the command line) a set of cryptographic functions (hash functions, symmetric and asymmetric encryption, CA); at the same time OpenSSL can be used as a cryptographic library in your own applications. As the name suggests, originally the library offered primarily the SSL protocol support. This functionality has been included up to now, but the library is offering much wider portfolio of cryptographic functions nowadays.

Address, version, license

OpenSSL is downloadable in the form of source code at the URL http://www.openssl.org. The binary form is part of many operating systems or can be downloaded from other source (use Google). OpenSSL can be compiled under many operating systems including MS Windows and UNIX systems. The authors are Eric A. Young and Tim J. Hudson. The license is better (more "free") than GPL, the OpenSSL license is close to the Apache license and permits both noncommercial and commercial use AND MODIFICATION. Currently the latest version is 1.0.2d, 1.0.1p, 1.0.0s and 0.9.8zg, some time ago there used to be two the versions of the library – the „engine" version (which included support of cryptographic HW devices) and the non-engine version, today all the versions include the HW device support automatically.

Documentation

Documentation of the OpenSSL **program** is fairly good and is available in the form of man pages (directory openssl/doc/apps), documentation of the API of the library is not so good, some functions are sufficiently documented (openssl/doc/crypto, for SSL API it's openssl/doc/ssl), but many functions remain undocumented and for a correct use the inspiration in the source codes of particular modules of the program (openssl/apps) or header files (openssl/include/openssl) is necessary.

Advantages/Disadvantages

The advantages of OpenSSL include good licensing terms (commercial use of modified code is permitted), availability of source codes, platform independence and wide functionality. On the other hand the disadvantages include poor documentation, and poor code quality ("hacks") in some parts of the library.

Program

The functionality offered by the OpenSSL program (command line) is divided into following modules (further info on parameters „openssl *module* help”):

- asn1parse - ASN.1 parsing tool
- ca - sample minimal CA application
- crl2pkcs7 - Create a PKCS#7 structure from a CRL and certificates.
- crl - CRL utility
- dgst, md5, md4, md2, sha1, sha, mdc2, ripemd160 - message digests
- dhparam - DH parameter manipulation and generation
- dsa - DSA key processing
- dsaparam - DSA parameter manipulation and generation
- ec - EC key processing
- ecparam - EC parameter manipulation and generation
- enc - symmetric cipher routines
- gendsa - generate a DSA private key from a set of parameters
- genrsa - generate an RSA private key
- ocsp - Online Certificate Status Protocol utility
- passwd - compute password hashes
- pkcs7 - PKCS#7 utility
- pkcs8 - PKCS#8 format private key conversion tool
- pkcs12 - PKCS#12 file utility
- rand - generate pseudo-random bytes
- req - PKCS#10 certificate request and certificate generating utility.
- rsa - RSA key processing tool
- rsautl - RSA utility
- s_client - SSL/TLS client program
- s_server - SSL/TLS server program
- sess_id - SSL/TLS session handling utility
- smime - S/MIME utility
- speed - test library performance
- spkac - SPKAC printing and generating utility
- verify - Utility to verify certificates.
- x509 - Certificate display and signing utility
- x509v3_config - X509 V3 certificate extension configuration format

OpenSSL supports algorithms RC4, RC2, DES3, DES, CAST, BLOWFISH, AES, (to get the full list of modes use „openssl list-cipher-commands“), MD2, MD4, MD5, RMD160, SHA, SHA1, RSA, DSS, DH (see openssl ciphers -v).

Usage of the program

Let's look at one of the typical uses of the OpenSSL program i.e. at the certificate requesting, signing and revoking (CRL generation).

1. Generate the RSA key of the CA
    - **`openssl genrsa -out ca.key 2048`**
2. Creation of the self-signed certificate of the CA
    - **`openssl req -new -x509 -days 365 -key ca.key -out ca.crt [-md5] [-sha1] [-sha256] [-sha224] [-sha384] [sha512]`**
    - **`openssl x509 -in ca.crt -text -noout`**
3. Generate RSA key of a user (e.g. server)
    - **`openssl genrsa -out server.key 2048`**
4. Generate the certificate request
    - **`openssl req -new -key server.key -out server.csr`**
    - **`openssl req -verify -in server.csr -text -noout`**
5. Certificate signature
    - https://www.openssl.org/docs/manmaster/apps/ca.html
    - `ca.config`
    - **`openssl -in server.csr -out server.crt`**
    - **`openssl x509 -req -in server.csr -CA ca/ca.crt -CAkey ca/ca.key -CAcreateserial -out server.crt -days 365`**
6. Prepare the PKCS#12 file (certificate plus the private key)
    - **`openssl pkcs12 -export -out server.p12 -in server.crt -inkey server.key`**
7. Issue the CRL (it's empty now)
    - **`openssl ca -gencrl -config ca.config -keyfile ca.key -cert ca.crt -out ca1.crl [-md md5/sha1/sha224/...]`**
8. Revoke the user's certificate
    - **`openssl ca -config ca.config -revoke server.crt -keyfile ca.key -cert ca.crt`**
9. Issue a new CRL
    - **`openssl ca -gencrl -config ca.config -keyfile ca.key -cert ca.crt -out ca2.crl`**
10. Let's examine the CRLs
    - **`openssl crl -in ca1.crl -noout -text`**
    - **`openssl crl -in ca2.crl -noout -text`**

Next we can focus on the symmetric encryption:

1. file encryption
    - **`openssl enc -aes-256-cbc -salt -in file.txt -out file.enc`**
2. file decryption
    - **`openssl enc -d -aes-256-cbc -in file.enc`**