Understanding OpenSSL

OpenSSL (<https://www.openssl.org/>) is a cryptography library that provides various security tools such as the Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), and Secure Sockets Layer (SSL) protocols. OpenSSL is used by numerous operating systems to provide essential security and cryptography tools and functions. Understanding OpenSSL can help many people gain a better idea of various cryptography concepts and the importance of this single library.  
  
OpenSSL supports Linux, OS X, \*BSD, Solaris, OpenVMS, Windows, ReactOS, and many Unixoid systems. Developers can contribute to OpenSSL via GitHub (<https://github.com/openssl/openssl>).  
  
**Supported Tools**  
OpenSSL is a library that provides various functions to programs that require the OpenSSL library. OpenSSL offers the functions and code needed to support the below listed cryptography tools.  
  
**Ciphers**

* AES
* Blowfish (4 different modes - ecb, cbc, cfb, and ofb)
* Camellia
* SEED
* CAST-128
* DES (15 variations)
* IDEA (4 different modes - ecb, cbc, cfb, and ofb)
* RC2 (4 different modes - ecb, cbc, cfb, and ofb)
* RC4
* RC5
* Triple DES
* GOST 28147-89

**Certificates**

* X.509v3 (encoding/decoding ASN1 and PEM)

**Cryptographic hash functions**

* MD5
* MD4
* MD2
* SHA-1
* SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256)
* RIPEMD-160
* MDC-2
* GOST R 34.11-94

**Public-key cryptography**

* RSA
* DSA
* Diffie–Hellman key exchange
* Elliptic curve
* GOST R 34.10-2001

libssl.a provides support for client and server SSLv2, SSLv3, and TLSv1.  
libcrypto.a provides general encryption and X.509 v1/v3.  
  
**Forks**  
Various forks of OpenSSL have been made. OpenSSL itself is a fork of the SSLeay library developed by Eric A. Young and Tim J. Hudson.

* Agglomerated SSL was made in 2009 by Marco Peereboom (an OpenBSD developer). This fork aims to provide a simple interface.
* BoringSSL was made in June 2014 by Google.
* LibreSSL ([http://www.libressl.org/](https://www.libressl.org/)) was made in 2014 by the OpenBSD developers. The goals of this fork is to modernize and improve security and the library's code.

**Generating a Private Key**  
A private key is used to encrypt data that is decrypted by the public key and vice versa. The public key and private key are different (but related) cryptography keys that work together. To create a self-signed SSL certificate, generate the private key. Many people use RSA, 1024 bits, and TripleDES. However, 4096 bits is more secure, but slower to encrypt/decrypt.

openssl genrsa -des3 -out server.key 1024

**Generating a CSR Certificate**  
A Certificate Signing Request (CSR) is an unsigned copy of a SSL certificate. A CSR contains the public key, X.509 certificate attributes, and other information. Certificate Authorities (CA) require that server admins create a CSR before being given a digital certificate.

openssl req -key server.key -out server.csr

**Generate a SSL Certificate**  
Creating your own SSL certificate is known as a self-signing certificate and is legal and free or charge/cost. To successfully complete the below command, the user must already have a CSR (\*.csr file) and a private key (\*.key file). The below command will generate the SSL certificate as a \*.crt file.

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

**NOTE:** Be sure to run "chmod 400" on the created key and certificate files. This ensures that only Root can read the files.  
  
**OpenSSL Commands**  
Below are some commands to perform various actions using OpenSSL. Remember to replace "server", "key", "cert", and other place-holders with the proper values for your particular needs and situations.

* Creating a CSR (Certificate Signing Request) - openssl req -new -key server.key -out server.csr
* Creating RSA private keys - openssl genrsa -des3 -out server.key 1024
* Creating self-signed certificates - openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
* Creating self-signed certificates - openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
* Install a certificate - cp server.crt /usr/local/apache/conf/ssl.crt
* Install a private key - cp server.key /usr/local/apache/conf/ssl.key
* Removing the pass-phrase from the key - cp server.key server.key.org && openssl rsa -in server.key.org -out server.key

**Further Reading**

* Linux.org Cryptography Reading Guide - [http://www.linux.org/threads/cryptography-reading-guide.6231/](https://www.linux.org/threads/cryptography-reading-guide.6231/)
* Linux.org Security Reading Guide - [http://www.linux.org/threads/security-reading-guide.6356/](https://www.linux.org/threads/security-reading-guide.6356/)
* Intro to Cryptography - [http://www.linux.org/threads/intro-to-cryptography.8018/](https://www.linux.org/threads/intro-to-cryptography.8018/)
* Intro to Encryption - [http://www.linux.org/threads/intro-to-encryption.4376/](https://www.linux.org/threads/intro-to-encryption.4376/)
* Basic Security - [http://www.linux.org/threads/basic-security.4226/](https://www.linux.org/threads/basic-security.4226/)