

Building governmental Certification Authority using OpenSSL

Blerim Rexha, Ehat Qerimi, Valon Raça and Haxhi Lajqi

Faculty of Electrical and Computer Engineering

University of Prishtina

Prishtina, Kosova

blerim.rexha, ehat.qerimi, valon.raca and haxhi.lajqi {@fiek.uni-pr.edu}

Abstract

In this paper we present a different approach, by using command line scripts for building a governmental Certification Authority (CA) using OpenSSL tool.

OpenSSL is a project driven by volunteer programmers to develop an open source toolkit for implementing the SSL, TLS and general purpose cryptographic libraries. OpenSSL is a command line driven tool, therefore the challenge was creating (changing) the configuration file and writing a script for generating a X.509 certificate. The OpenSSL configuration file, which has the format of a standard INI file, with predefined sections, user defined sections, and values is used to generate private and public keys and also to sign the private based on the data on configuration file. Furthermore we use OpenSSL for signing the Certification Revocation List (CRL).

We will create the Root Certification Authority (Root CA) and Issuer CA. The Issuer CA shall be responsible for issuing ISO X.509 certificates to end users.

We compare, pro's and con's of OpenSSL with Microsoft's certificate Services, which are part of Windows Server since year 2000.

Key words: OpenSSL, SSL, TLS, PKI, CRL, RSA, digital signature, PKCS standards.

Public Key Infrastructure

Public Key Infrastructure (PKI) is the "set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public key certificates (PKC) based on public-key cryptography"[1]. PKI includes the certificate storage resources of a server, and also provides users a set of services and protocols for managing public keys. The main feature of PKI is the introduction of what are known as a Certification Authority (CA) and a Registration Authority (RA). PKI is a basic protocol on which relay Secure Multipurpose Internet Mail Extensions (S/MIME), Transport Layer Security (TLS), Internet Protocol Security (IPSec), Virtual Private Network (VPN), online shopping and online banking. These protocols provide the four main services of data security and privacy such as: confidentiality, integrity, authentication and non-repudiation.

A PKI consists of five types of component[2]:

- The Certification Authority (CA) is the highest (root) instance, which is trusted by participants to generate, assign and revoke public key certificate (PKC).
- The Registration Authority (RA) is an optional entity. Its responsibility is to verify that the subject's identity value matches the PKC request. The RA is also responsible for verifying that the subject possesses the private key as stated in the PKC request.
- PKC holders who get the issued certificates. PKC holders (also called end entities) can sign digital documents and decrypt documents using their private keys. A PKC is also known as an identity certificate (IC).
- PKI enabled applications that validate digital signatures and their certification paths from the known public key of a trusted CA. PKI enabled applications can encrypt documents using the public key from certificates of PKC holders.
- Repositories are public online resources that provide certificates and certificate status information.

Figure 1 presents a simplified view of the architectural model assumed by the PKIX14 Working Group[1].

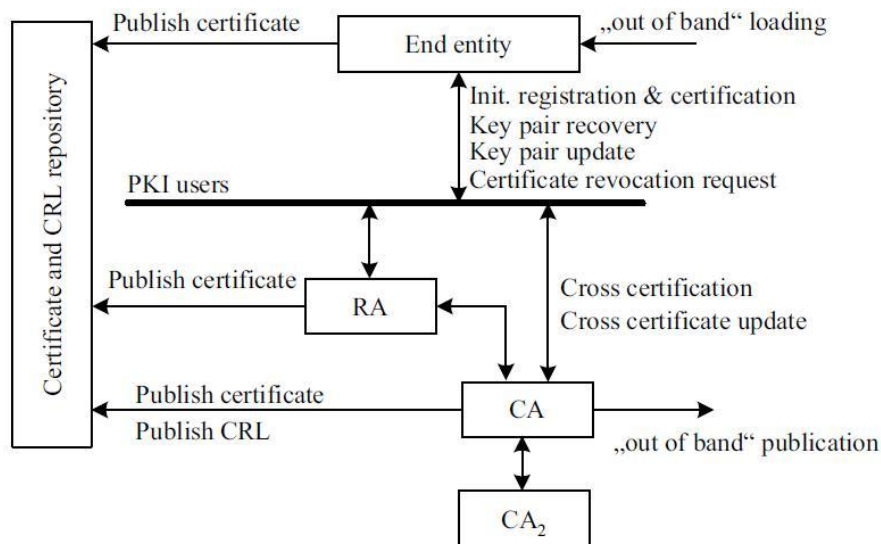


Figure 1: PKI entities

The end entity sends its certificate request to the RA (or in the case when RA is missing, direct to the CA) for approval. If it is actually approved, it is forwarded to the CA for signing. The Certification Authority verifies the certificate request and if it passes the verification, it is signed and the certificate is created. In governmental environment the end user digital certificate is issued by registration or issuing authority in manual mode and after successful validation is uploaded to public repository.

Digital Certificates

Digital certificates were first proposed by Loren Kohnfelder, in his master thesis at MIT in 1978, as a solution for efficient and authentic binding of private keys to end entities. This makes the use of public key cryptography practical for digital signatures. Digital certificate simply binds the public key with the corresponding name of some entity [3].

Digital certificates can be sent through insecure networks and can be stored in insecure media, because if any changes are made to the certificate the signature will be invalid. Digital certificates are usually not confidential and therefore can be freely stored in insecure public repositories. A digital certificate can be compared with passport in everyday life. A passport binds personal information such as photo, name, gender, address and birth date to a person and is valid for a finite period of time. All this personal information in the passport is verified by an entity like a government department; similarly the information in a digital certificate is verified by a CA[4].

In order to fulfill interoperability during the exchange of certificates in different systems the International Telecommunication Union (ITU) defined a standard in 1988 about X.509 Certificates as part of the X.500 directory recommendations [5]. The current version of X.509 certificates, (v3), was released in June 1996 [AFPS99]. The format of an X.509 v3 certificate is presented in Figure 1. It consists of a set of required and optional fields.

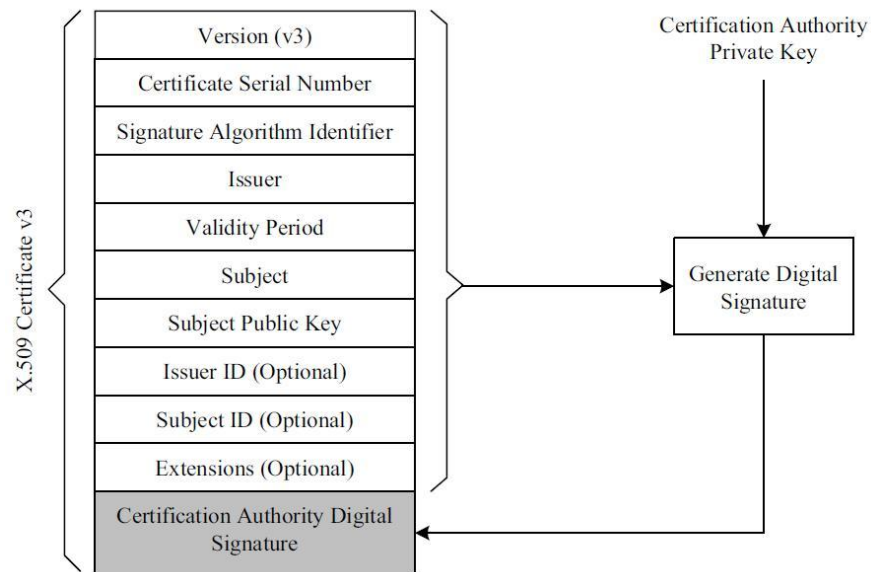


Figure 2: Format and content of X.509 v3 certificate

The format of an X.509 certificate is specified in Abstract Syntax One (ASN.1) and encoding follows in Distinguished Encoding Rules (DER). The certificate data structure fields so encoded are reduced to a list of more fundamental elements, whereby each element is either a primitive type or another list. The primitive types are then encoded in tag, length and value format (TLV). There has been many

suggestions from the community that the certificates should be encoded in XML, in order to avoid the problems with TLV encoding and to enjoy all the benefits of XML[6].

An X.509 certificate has a finite validity period. In the case of premature withdrawal of the certificate or if the associated private key is compromised, the question arises of how PKI enabled applications are to be informed about it. Certificate revocation lists (CRL) are mechanisms that are used to publish and distribute information about revoked certificates to PKI enabled applications. A CRL is data structure, digitally signed by the CA, which contains date and time of the CRL publication and the serial number of all revoked certificates[3]. The X.509 CRL format is an ITU and ISO standard with current version 2 (v2), a detailed description of all CRL fields is in[1]. The immediate question that arises about CRLs is how oft they should be updated. If PKI enabled applications do not check the CRL they are close to useless. Generally there are three methods that allow CRL propagation[7]:

- Polling of the current CRL, is a mechanism used by PKI enabled applications after each time the CA issues a new CRL. The update schedule is kept in the CRL data structure. The disadvantage of this approach is that the problem of having the actual CRL still exists. The update period of CRL can be kept short, but short is not tolerated by some time critical PKI enabled applications even if the CRL is updated hourly.
- Pushing is used by the CA as soon the CA revokes a certificate. The advantage of this approach is that the PKI enabled application always receives the current CRL. [FB01] describes the problems that appear in real PKI enabled applications, especially in the network infrastructure, using this approach.
- Online status checking is the most reliable method to determine the revocation status of the certificate. The advantage of this approach is that it does not require pushing or pulling a large amount of data over the network. However this approach requires that the CA should be available and reliable all the time.

The lifetime of a certificate depends on several factors and the optimal life time should be shorter as weeks. Applying this approach leads to the idea to use certificates only once and thus completely eliminate the need for CRLs.

OpenSSL

OpenSSL is a project driven by volunteer programmers to develop an open source toolkit for implementing the SSL, TLS and general purpose cryptographic libraries. OpenSSL is based on the SSLeay library developed by Eric A. Young and Tim J. Hudson. OpenSSL toolkit is free to use for commercial and non-commercial purposes[8]. OpenSSL is a command line driven tool, therefore the challenge was changing the configuration file and writing a script building the “Root CA”, “Issuing CA” and for generating a X.509 certificate.

Computer & Software Engineering¹ (CSE) LLC, is a local company in Kosova that has implemented a PKI and has its own Object Identifier (OID) assigned by IANA². CSE Trust Center has the Root CA that has its validity set to 20 years and Issuing CA for 10 years of validity. The Issuing CA is the only entity that issues digital certificates to end users, as is presented in Figure 3.

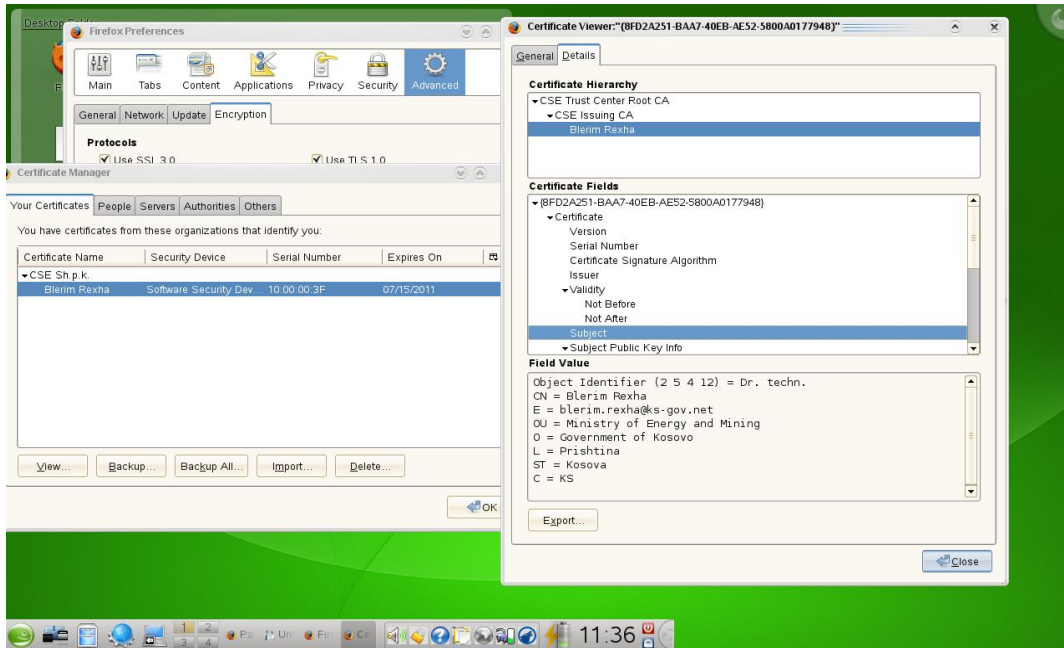


Figure 3: CSE governmental PKI structure

In order to use the OpenSSL toolkit we have developed the special purpose command line files for executing the “*openssl.exe*” application and “*CSECA.bat*” for using as command line file, as is presented in Figure 4. The configuration file is needed to set the path of the executables and the path on disk of the computer where the “*openssl.exe*” runs and are set the default values. The batch file enables to execute the “*openssl.exe*” in different modes such as: client, server, *crl* etc, as presented in Figure 4. It is also possible to fetch the batch file from other sources rather the keyboard. This increases the flexibility of the OpenSSL.

¹ CSE OID = 1.3.6.1.4.1.25106.1.1.1.0, www.cse-ks.com

² IANA – Internet Assigning Number Authority – www.iana.org

August 29, 2009

```

C:\Program Files\OpenSSL\cseca client
CSE Certification Authority utility
Written by Dr. Blerim Rexha, February 2006

Warning!
The content of the C:\RootCA\SubCA\temp\unc_client directory will be removed.
Press CTRL-C to break, or ENTER to continue...

Step 1: Generate the keys and the certificate request

Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'C:\RootCA\SubCA\temp\unc_client\client.key'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [Mosova]:
Locality Name (eg, city) [Pristina]:
Organization Name (eg, company) [CSE Sh.p.k.:Government of Kosova
Organizational Unit Name (eg, section) [PMI Department]:Ministry of ITC
Serial number [123456789]:
Your title [Dr. techn.]:
Common Name (eg, YOUR name) [Blerim Rexha]:Blerim Rexha

Echo CSE Certification Authority utility
Echo Written by Dr. Blerim Rexha, February 2006
Echo.
if %1==genRootCA. goto genRootCA
if %1==genSubCA. goto genSubCA
if %1==client. goto client
if %1==signreq. goto signreq
if %1==server. goto server
if %1==gencl. goto gencl

:help
Echo Usage:
Echo CSECA genRootCA - generate root CA certificate
Echo CSECA genSubCA - generate subordinate CA certificate
Echo CSECA server - generate and sign server certificate
Echo CSECA client - generate and sign client certificate
Echo CSECA signreq - sign certificate request
Echo CSECA gencl - generate csl file
Echo.
goto exit

:genRootCA
if not exist C:\Program Files\OpenSSL\openssl.exe goto ssldoesntexist
if not exist C:\RootCA goto genca2
Echo Warning!
Echo The content of the C:\RootCA directory will be removed.
Echo Press CTRL-C to break, or ENTER to continue...
Echo.
pause > nul
Echo Removing C:\RootCA...
rmdir /S /Q C:\RootCA > nul
rem deltree /Y C:\RootCA > nul
goto genca3

```

Figure 4: Executing and the view of the batch file for Certification Authority

Microsoft Certificate Services

Certificate Services are included in Windows Server since year 2000, and with new versions are enhanced with features. The Microsoft Certificate Service is used to issue and manage certificates for a Public Key Infrastructure (PKI). Certificate Services allows a computer running Windows Server (2000, 2003 and 2008 version) to receive requests for certificates from users and computers, verify the identity of a requestor, issue and revoke certificates, and publish a CRL.

Certificate Services is a Windows service that runs on a designated certificate server. Certificate servers can be configured as one of four types of certification authorities[8]:

- **Enterprise root CA** – is a root CA is the certificate server at the root of the hierarchy for a Windows domain. It is the most trusted CA in the enterprise and must have access to Active Directory service.
- **Enterprise subordinate CA** – is a certificate server that will be a member of an existing CA hierarchy. It can issue certificates but must obtain its own CA certificate from the enterprise root CA.
- **Stand-alone root CA** – is the certificate server at the root of a nonenterprise hierarchy. It is the most trusted CA in its hierarchy and doesn't need access to Active Directory service.
- **Stand-alone subordinate CA** - is a certificate server that will be a member of an existing nonenterprise hierarchy. It can issue certificates but must obtain its own CA certificate from the stand-alone root CA in its hierarchy.

Certificate servers don't have to be dedicated to Certificate Services and can be the same servers used for web publishing. It is a good idea to have dedicated CA server in corporate domain that will act as certificate server and to use these servers only for that purpose. The installed certificate services can be accessed through this <http://servername/certsrv> address.

Comparing open source vs. Microsoft

Even Microsoft has made significant improvements in its actual certificates services in Windows Server 2008. Through its "CAPolicy.inf" configuration file is possible to customize the certification authority.

The OpenSSL through its configuration file and batch file gives greater flexibility. Through batch files OpenSSL is very suitable for testing different configuration modes of encryption schemes. Furthermore in any time it is possible to modify the source code, i.e. to add the newest encryption algorithms to OpenSSL project. In fact changes to OpenSSL can be made without knowledge of encryption algorithms, a basic knowledge of structure of batch file is recommended.

Conclusions

With OpenSSL is very easy to create and manage certification authorities for use in governmental applications. OpenSSL is open source and free of charge. It can be audited deeply till to the "bit" details, which is not the case in other commercial software. Through configuration files it increases the flexibility and supports different encryption schemes.

References

- [1] A. Arsenault and S. Turner. Internet x.509 public key infrastructure: Roadmap. PKIX Working Group; Internet Draft: <http://www.ietf.org/internet-drafts/draft-ietf-pkix-roadmap-09.txt>, July 2002.
- [2] William Burr, Donna Dodson, Noel Nazario, and W. Timothy Polk. Minimum interoperability specification for pki components, version 1 (mipspec). Technical report, NIST, <http://csrc.nist.gov/pki/mispec/welcome.html> , September 1997.
- [3] Peter Gutmann - X.509 style guide. University of Auckland, <http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt>, visited August 2009
- [4] Mwelwa Chibesakunda - Digital certificates: Study on attribute certificates. University of Cape Town, <http://people.cs.uct.ac.za/~mchibesa/report.pdf>, 2002.
- [5] ITU - International Telecommunication Union - Telecommunication Standardization Sector ITU-T; The Directory: Public Key and Attribute Certificate Frameworks. <http://www.itu.int>, 2009
- [6] Juha Paarjarvi. Xml encoding of SPKI certificates. IETF, March 2000.
- [7] Jalal Feghhi, Jalil Feghi, and Peter Williams. Digital Certificates: Applied Internet Security. Addison Wesley, ISBN = 0-201-30980-7, 1999.
- [8] OpenSSL - About the openssl project. <http://www.openssl.org/about/>, August 2009.
- [9] Microsoft, <http://technet.microsoft.com/en-us/library/bb727098.aspx#EEAA> visited August 2009