Report on Certificate Tools

Oliver Bodnár

August 2016

Contents

Part I

Overview

Chapter 1

Overview Tables

1.1 General

Table 1.1: Overwiew General

Tools	General			
	Type	View + info	License	Operating System
OpenSSL	PKCS12	Yes	Public	Both
Keygen	JKS	Yes	Public	Both
GnuPG				
Keystore Explorer				

1.1.1 Type

This section defines type of storage file in which the certificates or keys are saved. Most common are PKCS#12 (.pfx and .p12 extentions) and JKS (Java KeyStore).

1.1.2 View and information

This section shows whether it is possible to view certificates or keys and additional information. Yes means that at least viewing is supported while not necessary meaning possibility of viewing more information about certificate.

1.1.3 License

Type of license and possibility of using said tool for testing or production code. Public means that license is not requiered for use in production code but does not mean that it should be used as such. Definition if it is advised to use said tool in production code or only in testing enviroment will be talked about in the next chapter under each tool.

1.1.4 Operating Systems

In this part the Operating System where the tool is imported is defined. Both means that it is implemented under UNIX like systems and Windows systems. Otherwise the Operating System is defined (Windows — UNIX).

1.2 Generation and signing of certificates

1.2.1 Keys, certificates and basic constraints

Table 1.2: Generation of keys and certificates and basic constrints

Tool	Generate keys				Basic Constraints	
	+ self-signed certificate	+ CSR	Specify length	Specify algorithm	Specify Type	Specify path length
OpenSSL	Yes	Yes	Yes	Yes	Yes	Yes
Keytool	Yes	Yes	Yes	Yes	Yes	Yes
GnuPG						
Keystore explorer						

Generate keys

Self-signed certificate possibility of using 1 command to generate key pair and self-signed certificate

Certificate Signing Request possibility of using a command to generate key pair and certificate signing request to certificate authority.

Specify length possibility to specify the length of output key

Specify algorithm possibility to choose between different types of algorithms for key generation

Basic Constraints

Specify Type specify if generated certificate will belong to certificate authority or whether it will be end certificate

Specify path length specify the maximum length of certificate authority chain

1.2.2 Specifications

Table 1.3: Specifications

Tool	CSR signing	Privkey + signed chain	Specify certificate validity	SAN for end certificates	Support for CSP
OpenSSL	Yes	Yes	Yes	Yes	Yes
Keygen	Yes	Yes	Yes	Yes	Yes
GnuPG					
Keystore explorer					

Certificate Signing Request signing possibility of signing a certificate signing request with certificate authority's key

Create combination of private key and signed chain possibility of generating private key and chain signed by certificate authority that will be outputed to a single file

Specify certificate validity possibility of choosing how long will the certificate be valid. This should be done by certificate authority.

Setting Subject Alternative Name for end certificates possibility of choosing Subject Alternative Name for end certificates. That should be done by IP's or DNS addresses.

Support for Cryptographic Service Provider whether the use, choosing and changing of Cryptographic Service Provider is supported.

1.3 Conversions

Table 1.4: Conversions

Tools	Conversions			
	Exporting		Direct JKS and PKCS12	Import certificate and private key into a file
	Certificate/chain only from file	Private key only		
OpenSSL	Yes	Yes	No	Yes
Keygen	Yes	No	Yes	Yes
GnuPG				
Keystore explorer				

1.3.1 Exporting

Certificate or certificate chain from a file

Possibility of extracting certificate or certificate only from a file. Choice of Yes based on possibility of extracting either from a file.

Private key only

Possibility of extracting private key from tool's file storage type of choice.

1.3.2 Direct conversion between Java Keystore and PKCS#12 file

Possibility of direct conversion (by a command of tested tool) between Java KeyStore and PKCS#12 type file.

1.3.3 Importing of certificates and keys into storage files

Possibility of importing (additional?) certificates and keys into storage files of said tool. Yes if it is possible to import or add another certificate or key into storage.

Part II

Tools

Chapter 2

OpenSSL

2.1 General

2.1.1 Type

OpenSSL uses PKCS12 to store keys and or certificates. Certificates and keys made in OpenSSL however are being made into PEM or DER encoded file. Said keys/certificates can then be stored inside single .pfx file.

2.1.2 View and information

Viewing stored certificates

PEM encoded certificates (.pem|.cer|.crt):

```
openssl x509 -in sample_cert.extention -text -noout
```

DER encoded certificates (.der):

```
openssl x509 -in certificate.der -inform der -text -noout
```

Importing PEM or DER encoded keys or certificates into PKCS#12 file:

```
openssl pkcs12 -export -in file.pem \
        -inkey privateKeyFile.key -out file.p12 \
        -name "My Certificate" -certfile othercerts.pem
```

certfile option is used only if importing more certificates into a single PKCS#12 file is wanted.

2.1.3 License

OpenSSL is free to use commercialy, however creating CA is not advised in O'REILLY's Network Security with OpenSSL by J. Viega, M. Messier and P. Chandra on page 59,

> Since OpenSSL's command-line CA functionality was intended primarely as an example of how to use OpenSSL to build a CA, we don't recommend that you attempt to use it in a large production environment.

and also hinted in manual pages:

> The ca command is quirky and at times downright unfriendly.
>
> The ca utility was originally meant as an example of how to do things in a CA . It was not supposed to be used as a full blown CA itself: nevertheless some people are using it for this purpose.
>
> The ca command is effectively a single user command: no locking is done on the various files and attempts to run more than one ca command on the same database can have unpredictable results.

2.2 Generation and signing of certificates

2.2.1 Keys, certificates and basic constraints

Generate keys

Self-signed certificate It is possible to generate private key and self signed certificate with

```
openssl req -x509 -sha256 -nodes -days 365 \
-newkey rsa:[length] -keyout privateKey.key \
        -out certificate_name.crt
```

Certificate Signing Request It is possible to generate private key and CSR with

```
openssl req -out CSR.csr -new -newkey rsa:[length] \
        -nodes -keyout privateKey.key
```

Specify length Length is specified while generating key/certificate.

Specify algorithm Currently OpenSSL supports Public-key cryptography algorithms: RSA, DSA, Diffie-Hellman key exchange, Elliptic Curve

In the past also support for GOST R 34.10-2001 but as of January 2016 deprecated (https://mta.openssl.org/pipermail/commits/2016-January/003023.html)

Basic constraints

Basic constraints are defined in CA's openssl.cnf `[ v3_req ]` part.

```
[ v3_req ]

basicConstraints=critical,CA:<BOOL_VAL>, pathlen:<maxChainLengthInteger>
```

Specify Type To specify if generated certificate belongs to CA or is end certificate you choose `<BOOL_VAL>` true or false respectively.

Specify path length Specifying of path length is made by setting a `<maxCheinLengthInteger>` to desired value (e.g. if generated certificate should be used only for generating end certificates the value is 0)

2.2.2 Specifications

Certificate Signing Request signing

SR signing can be done by CA created in openSSL by command

```
openssl ca -config ca/openssl.cnf \
        -extensions server_cert -days <validity_time>  \
-notext -md [message digest alg] -in ca/csr/www.example.com.csr.pem \
        -out ca/certs/www.example.com.cert.pem
```

Create combination of private key and signed chain

First creation of the request is needed, after that request must be signed. To combine them together the creation of PKCS#12 file is needed with commands:

```
openssl pkcs12 -export -out outfilename.p12 \
        -in signed_certificate.crt -inkey privateKey.key \
        -chain -CAfile ca-all.crt -password pass:PASSWORD
```

Specify certificate validity

Time how long the certificate will be valid can be given in csr but only as a suggestion. Ultimately the real time of validity is always given by CA signing the CSR.

Setting Subject Alternative Name for end certificates

The Subject Alternative Name is also given in `[ v3_req ]` part of openssl.cnf that CA uses. It can be given as an IP address but also as a DNS name.

```
[ v3_req ]
subjectAltName = @alt_names

[alt_names]
DNS.1 = example1.com
DNS.2 = example2.com
DNS.<next_number> = dns_webaddress.com
```

Support for Cryptographic Service Provider

OpenSSL has native support for Cryptographic Service Provider since version 0.9.7 by creating PKCS#12 file using -CSP option to set the cspname parameter

2.3 Conversions

2.3.1 Exporting

Certificate or certificate chain from a file

To export certificate from a PKCS#12 file use:

```
openssl pkcs12 -in name.[pfx|p12] -nokeys -clcerts -out name.pem
```

To export certificate chain from a PKCS#12 file use:

```
openssl pkcs12 -in name.[pfx|p12] -nokeys -cacerts -out CAchain.pem
```

It should be mentioned that if there are multiple certificates in a chain they are all going to be in same output file.

Private key only

To export an encrypted private key use:

```
openssl pkcs12 -in name.[pfx|p12] -nocerts -out name.encrypted.priv.key
```

To export an unencrypted private key use:

```
openssl pkcs12 -in name.[pfx|p12] -nocerts -nodes -out name.unencrypted.priv.key
```

2.3.2 Direct conversion between Java Keystore and PKCS#12 file

Direct conversion between Java Keystore and PKCS#12 type file is not supported by OpenSSL

2.3.3 Importing certificates and keys into storage files

To import certificate and private key into a single PKCS#12 type file use:

```
openssl pkcs12 -export -out certificate.pfx \
        -inkey privateKey.key -in certificate.crt \
        -certfile CACert.crt
```

Chapter 3

Keytool

3.1 General

3.1.1 Type

Keytool has native support for both PKCS#12 and Java KeyStore type files

3.1.2 View and information

To view information about PEM or DER encoded certificate use:

```
keytool -printcert -file certificate.extention
```

To view information about Jave KeyStore or PKCS#12 encoded file use:

```
keytool -list -v -keystore store_file.[pfx|p12|jks]
```

Viewing Public key part of certificate is not availiable in keytool.

3.1.3 License

Keytool is free to use in production code and is licensed under Apache Licence Version 2.0. For more detailed description look at http://www.apache.org/licenses/LICENSE-2.0.txt

3.2 Generation and signing of certificates

3.2.1 Keys, certificates and basic constraints

Generate keys

Self-signed certificate To generate new key pair and self-signed certificate use:

```
keytool -genkey -keyalg [RSA|DSA] -alias [alisaname] \
        -keystore [keystore_file] -storepass [password] \
        -validity [time] -keysize [length]
```

Keytool natively supports only DSA of length in between 512 and 1024, and is considered to be dangerous to use. RSA with key length of at least 2048 is preffered.

Certificate Signing Request To generate a Certificate Signing Request generation of a key pair with alias is needed:

```
keytool -genkey -alias namealias -keyalg [RSA|DSA] \
        -keysize [length] -keystore [keystore_path | new_keystore_name]
```

To create a CSR you then need to do:

```
keytool -certreq -keyalg [keyalg_from_genkey] -alias [alias_from_genkey] \
        -file certreq.csr -keystore [jks_from_genkey]
```

Where you need to set -keyalg -alias and -keystore as chosen when generating key pair earlier.

Specify length Length is specified when generating keys, RSA support for up to 4096, DSA support from 512 up to 1024.

Specify algorithm Algorithm is specified while generating key pair, currently support for DSA, RSA. EC is needed to be implemented.

Basic constraints

Specify Type With keytool you can make a self-signed CA certificate by using `-ext BC=ca:[true|false]` option.

Specify path length Specifying of path length is possible while making a certificate by using `-ext BC=pathlen:[int path length]` option.

3.2.2 Specifications

Certificate Signing Request signing

Keytool can sign a CSR if a CA certificate and a keystore containing it are avaliable:

```
keytool -gencert -rfc -keystore [root_keystore.jks] \
        -alias [CA_cert_alias] -storepass [store_password] \
        -infile [CSR_file.csr] -validity <int_length> \
        -outfile [output_certificate.crt] -ext BC=ca:[true|false] \
        -ext SAN=ip:<IP_address>,dn:<DNS_address>
```

Create combination of private key and signed chain

You can import a key pair into a Java KeyStore file along with signed chain if the signed chain is provided and a CA certificate is imported into said Java Keystore.

Specify certificate validity

Specification of certificate validity is set when generating key pair in `-validity [time]` option.

Setting Subject Alternative Name for end certificates

To set a Subject Alternative Name for certificate use `-ext san=dns:www.dnsexample.com,ip:1.1.1.1` option which will add SAN to certificate.

Support for Cryptographic Service Provider

Support for Cryptographic Service Provider is native but there is need for a definition of path where the CSP is located.

```
-storetype [keystore_type_of_implementation]
-provider [provider_implementation_name]
-providerpath [provider_implementation_type]
```

3.3 Conversions

3.3.1 Exporting

Certificate or certificate chain from a file

To export certificate with alias `myalias` use:

```
keytool -export -alias myalias -file [output.crt] -keystore [source_keystore.jks]
```

Exported certificate will be DER encoded.

Exporting of certificate chain from a Java KeyStore is not availiable using keytool.

Private key only

Exporting of private key only is not availiable in keytool.

3.3.2 Direct conversion between Java Keystore and PKCS#12 file

Conversion between Java KeyStore and PKCS#12 file is availiable. To convert Java KeyStore file to PKCS#12 use:

```
keytool -importkeystore -srckeystore [existing_keystore.jks] \
        -destkeystore [destination_filename.p12|pfx] -srcstoretype JKS \
        -deststoretype PKCS12 -srcstorepass [.jks password] \
        -deststorepass [same as .jks password] -srcalias [source_alias] \
        -destalias [source_alias] -srckeypass [source_key_password] \
        -destkeypass [source_key_password] -noprompt
```

To convert PKCS#12 to JKS use:

```
keytool -importkeystore -srckeystore [PKCS12_file.pfx|.p12] \
        -srcstoretype pkcs12 -destkeystore [dest_keystore_name.jks] \
        -deststoretype JKS
```

3.3.3 Importing certificates and keys into storage files

To import certificate use:

```
keytool -import -alias [certificate_chosen_alias] \
        -file [certificate_file.ext] -keystore [target_keystore]
```

Importing of keys alone is not supported, you need to fold them into a Java KeyStore or PKCS#12 file first.

Chapter 4

GnuPG

4.1 General

4.1.1 Type

4.1.2 View and information

4.1.3 License

4.2 Generation and signing of certificates

4.2.1 Keys, certificates and basic constraints

Generate keys

Self-signed certificate

Certificate Signing Request

Specify length

Specify algorithm

Basic constraints

Specify Type

Specify path length

4.2.2 Specifications

Certificate Signing Request signing

Create combination of private key and signed chain

Specify certificate validity

Setting Subject Alternative Name for end certificates

Support for Cryptographic Service Provider

4.3 Conversions

4.3.1 Exporting

Certificate or certificate chain from a file

Private key only

4.3.2 Direct conversion between Java Keystore and PKCS#12 file

4.3.3 Importing certificates and keys into storage files

Chapter 5

Keystore Explorer

5.1 General

5.1.1 Type

5.1.2 View and information

5.1.3 License

5.2 Generation and signing of certificates

5.2.1 Keys, certificates and basic constraints

Generate keys

Self-signed certificate

Certificate Signing Request

Specify length

Specify algorithm

Basic constraints

Specify Type

Specify path length

5.2.2 Specifications

Certificate Signing Request signing

Create combination of private key and signed chain

Specify certificate validity

Setting Subject Alternative Name for end certificates

Support for Cryptographic Service Provider

5.3 Conversions

5.3.1 Exporting

Certificate or certificate chain from a file

Private key only

5.3.2 Direct conversion between Java Keystore and PKCS#12 file

5.3.3 Importing certificates and keys into storage files

Chapter 6

Tool Page Template

6.1 General

6.1.1 Type

6.1.2 View and information

6.1.3 License

6.2 Generation and signing of certificates

6.2.1 Keys, certificates and basic constraints

Generate keys

Self-signed certificate

Certificate Signing Request

Specify length

Specify algorithm

Basic constraints

Specify Type

Specify path length

6.2.2 Specifications

Certificate Signing Request signing

Create combination of private key and signed chain

Specify certificate validity

Setting Subject Alternative Name for end certificates

Support for Cryptographic Service Provider

6.3 Conversions

6.3.1 Exporting

Certificate or certificate chain from a file

Private key only

6.3.2 Direct conversion between Java Keystore and PKCS#12 file

6.3.3 Importing certificates and keys into storage files