Matrix Key and Certificate Generation Utilities

Table of Contents

1 Certificate and Key Overview

This document explains how to use the Matrix command line utilities to easily generate and format private key and X.509 certificate files suitable for use where public-key cryptography is required. All files generated with the utilities output standards compliant formats.

1.1 Certificate Authorities, Certificates and Private Keys

This document does not discuss the complicated topics of Public Key Infrastructure (PKI) in relation to security protocols. However, a basic overview of public keys and certificates is essential to understanding how authentication is performed in these environments.

The hierarchy of the trust chain begins with the creation of a new private key and a self-signed Certificate Authority (CA). A self-signed CA can then "issue" certificates. Any certificate that has the ability to issue other certificates is called a CA. It is not uncommon for a self-signed CA to issue other CAs and create a certificate chain.

To issue a certificate, the CA must have a certificate request. A certificate request is created by generating a new private key and then using that private key to create the request. The certificate request contains information about the requesting entity. The private key will not be shared with the CA when creating a new issued certificate. However, the CA will use its own private key during the issue process. This is the only time the private key of the CA is used. The self-signed CA private key should never be loaded into a MatrixSSL application.

1.2 Buy or Generate?

The most common way to obtain certificates is to buy them from a commercial certificate authority. This will result in a public key that has been digitally signed by a trusted third party so that client applications receiving the certificate can be very sure they are communicating with the entity they intended to communicate with. One benefit of obtaining certificates from a popular commercial certificate authority is that the issuing CA file will be pre-installed in web browsers or platform key chains to enable universal SSL clients to communicate with your servers without additional configuration.

If generation is an option, these utilities are used to do that.

1.3 Matrix Certificate Generation Features

The X.509 standard is large and constantly growing. The Matrix certificate generation utility implements the minimum required subset of extensions and a couple of the most common. If support for specific extensions is needed, please contact Rambus.

1.3.1 Distinguished Name Attributes

The following table lists the Distinguished Name Attributes supported by MatrixSSL, together with the compilation options needed to enable them.

Compilation option	Distinguished Name Attributes
always enabled	Attributes listed as "MUST support" in RFC 5280: `country, organization, organizationalUnit, dnQualifier, serialNumber, state, commonName, domainComponent`

`#define USE_EXTRA_DN_ATTRIBUTES_RFC5280_SHOULD`	Attributes listed as "SHOULD support" in RFC 5280: `locality, title, surname, givenName, initials, pseudonym, generationQualifier`
`#define USE_EXTRA_DN_ATTRIBUTES`	Attributes not mentioned in RFC 5280: `streetAddress, postalAddress, telephoneNumber, uid, name, email`

1.3.2 Certificate Extensions

The following table lists the certificate extensions supported by the MatrixSSL tools.

Supported Extension	Description
basicConstraints	Identifies a cert as a CA and how long a certificate chain it will allow
subjectAltName	Alternative names that associate specific DNS, IP Address, and other identification with the certificate
issuerAltName	Analogous to subjectAltName, but contains issuer information
subjectKeyId	Fingerprint of the subject's public key (automatically included)
authorityKeyId	Fingerprint of the issuer's public key (automatically included)
keyUsage	Supports keyAgreement and keyCertSign usage
extendedKeyUsage	Supports TLS Server Authentication and TLS Client Authentication
authorityInfoAccess	This extension is used to specify ways for accessing information about a CA. The most common use is to provide the URI of the CA's OCSP responder in this extension
certificatePolicies	Policies indicating the terms under which the certificate should be used
policyConstraints	The requireExplicitPolicy and inhibitPolicyMappings constraints are supported
policyMappings	This extension is used in CA certificates to specify pairs of policies that the CA considers to be equivalent

2 GETTING STARTED

2.1 Compiling the Matrix crypto library

Ensure the `USE_CERT_GEN` is defined in *./crypto/cryptoConfig.h* in order to build a *libcrypt_s* library with the needed functionality.

2.2 Compiling the Utilities

These command line utilities have been written for use on POSIX platforms and will compile using *./apps/crypto/Makefile*

From the root of the product directory structure, change directory to the ./apps/crypto directory and build.

```
$ cd apps/crypto
```

Compile using the provided *Makefile*

```
$ make
```

2.3 Configuration Files

The utilities have been written to prompt the user for any required configuration information that was not provided through the command line. However, it is strongly recommended the user provide configuration files to the utilities for two reasons.

1. All extensions are supported through configuration file entries
2. To preserve the information that went into creating the certificate requests and certificate files for later reference.

The configuration file format uses a simple attribute=value format that must adhere to the following rules.

- One entry per line
- There must be no spaces between the attribute the equals sign and the value
- The value must be double quoted (the value may contain spaces)
- The line must terminate with a semicolon

For example:

```
ca="1";
serialNum=”756”;
pathLen="3";
validDays="365";
algorithm="sha1";
country="US";
organization="Acme Inc";
commonName="www.sampleacmesite.com";
```

```
organizationalUnit="Test Department";
stateOrProvince="WA";
locality="Seattle";
```

2.3.1 Distinguished Name Attributes

The Distinguished Name (DN) field is used to describe the identity of the certificate subject. The DN field consist of attributes. The following example configuration file adds a value for every DN attribute supported by MatrixSSL. Note that `serialNum` refers to the serial number of the certificate, while `serialNumber` corresponds to the serialNumber field of the Distinguished Name.

```
serialNum="756";
pathLen="3";
validDays="365";
algorithm="sha1";
commonName="Common Name";
country="US";
stateOrProvince="Test State or Province";
locality="Test Locality";
organization="Organization Name";
organizationalUnit="Organizational Unit Name";
serialNumber="012bf123aa";
name="Givenname Surname";
givenName="Givenname";
surname="Surname";
initials="GS";
pseudonym="Pseudonym";
streetAddress="My StreetAddress 99";
telephoneNumber="1234-5678-9012";
title="Dr.";
postalAddress="12345";
generationQualifier="III";
dnQualifier="123456789";
domainComponent="rambus.com";
uid="1234";
email="gsurname@rambus.com";
```

2.3.2 SubjectAltName Configuration Entry

The X.509 Subject Alternative Name extension is a widely used identifier that is supported in certificate request and certificate creation. The extension has several different supported types so the configuration entry is a bit different from the normal.

To add a subjectAltName entry to a certificate request or to a certificate generation operation begin the configuration line with `subjectAltName-` followed by one of the supported types and then the equals sign. These are the supported types and sample entries:

```
        subjectAltName-dNSName="127.0.1.1";

        subjectAltName-rfc822Name="jim@jimbo.gov";

        subjectAltName-directoryName="/root";

        subjectAltName-iPAddress="127.1.1.1";

        subjectAltName-uniformResourceIdentifier="1.2.3.4";

        subjectAltName-otherName="2ab00f:some other identifier";
```

The value for the `otherName` type requires an `<OID>:<string>` format. The OID must be in a hex format for the desired OID (the dot notation is not supported here).

2.3.3 IssuerAltName Configuration Entry

The issuerAltName extension is analogous to subjectAltName. Also the configuration entry for issuerAltName is identical, except that the "issuerAltName" prefix should be used instead of "subjectAltName".

2.3.4 KeyUsage Configuration Entry

The keyUsage entry supports only "keyAgreement" and "keyCertSign" usages. If both are desired, simply append them in a single entry:

```
keyUsage="keyAgreement keyCertSign";
```

2.3.5 ExtendedKeyUsage Configuration Entry

The extendedKeyUsage entry supports only "serverAuth" and "clientAuth" usages. If both are desired, simply append both in a single entry:

```
extendedKeyUsage="serverAuth clientAuth";
```

2.3.6 Authority Information Access

The authorityInfoAccess extension can be used to specify the location of the OCSP responder or a location from which the issuer certificates can be downloaded. The configuration entries for this extension must begin with "authorityInfoAccess-", followed by either "ocsp" or "caIssuers". Multiple ocsp and caIssuers fields are supported. The example below has one caIssuers entry, but two OCSP entries:

```
authorityInfoAccess-ocsp="http://ocsp.rambus.com";
authorityInfoAccess-caIssuers="http://ca.rambus.com/cacerts.der";
authorityInfoAccess-ocsp2="http://ocsp2.rambus.com";
```

2.3.7 Certificate Policies

The Certificate Policies extension has a somewhat complex hierarchical structure. The extension consists of multiple PolicyInformation values, which include a policy OID and optionally multiple PolicyQualifierInfos.

A PolicyQualifierInfo contains either a Certificate Practice Statement (CPS) location or UserNotice. The most common way to encode a CPS location is to specify an URI; MatrixSSL only supports this method. A UserNotice is a piece of user-readable text describing the policy. The following example adds two policies, each with 3 qualifiers (2 CPS entries and 1 UserNotice). Note that the certificatePolicy OID must be specified as a hex string representing the DER-encoded OID. For more information on DER encoding of OIDs, consult the ASN.1 literature, such as the standard tutorial A *Layman's Guide to a Subset of ASN.1, BER and DER*, for example.

```
certPolicy1-id="67810C010201";
certPolicy1-cps="http://www.rambus.com/policy1/cps1";
certPolicy1-cps="http://www.rambus.com/policy1/cps2";
certPolicy1-unotice1-organization="Rambus Inc.";
certPolicy1-unotice1-noticeNumbers="1,2,3";
certPolicy2-id="67810C010202";
certPolicy2-cps="http://www.rambus.com/policy2/cps1";
certPolicy2-cps="http://www.rambus.com/policy2/cps2";
certPolicy2-unotice1-organization="Rambus Global Inc., Finnish branch";
certPolicy2-unotice1-noticeNumbers="4,5,6";
```

2.3.8 Policy Mappings

A policy mapping consists of a pair of policy OIDs. As in the Certificate Policies extension, the policy OIDs must be DER encoded and given as hex strings. For example:

```
policyMappings="67810C010201:67810C010202";
policyMappings="67810C010203:67810C010204";
```

2.3.9 Policy Constraints

MatrixSSL supports the following two policy constraints, which are defined in RFC 5280: requireExplicitPolicy and inhibitPolicyMappings. The following example configuration entries specify that an explicit policy is required for the 5 next certificates down the certificate chain, and disallows policy mappings after the next 2 certificates down the chain.

```
policyConstraints-requireExplicitPolicy="5";
policyConstraints-inhibitPolicyMapping="2";
```

3 RSA KEY GENERATION

The `matrixRSAkeygen` utility is used to generate PKCS#1 private key files. Private keys are the foundation for public-key cryptography and **must never be shared**. It is recommended the `-pass` option be used in the command line to encrypt the private key.

No configuration files are used for this utility.

3.1 Usage

```
matrixRSAkeygen -out filename [-outform type] [-keysize size] [-pass password]
```

out	Required. Identifies the file name of the new private key file. If omitted, the user will be prompted for a file name.
outform	Optional. Either `PEM` (default) or `DER`. It is not possible to password protect DER formatted files in PKCS#1 so this format is not recommended.
keysize	Optional. Allows the user to set the RSA modulus key byte size (key strength). The possible values for this option are: `1024`, `2048`, or `4096`. If no key size is given, a key strength of 1024 will be used. The tradeoff for the more secure larger keys is a performance slowdown when cryptographic operations are used.
pass	Optional. Allows the user to generate an encrypted file using the PKCS#5 standard. Pass the plaintext password to this option. Password protecting private key files is strongly recommended.

3.2 Examples

Generate a 1024 bit unprotected private key file:

```
matrixRSAkeygen -out privkey.pem
```

Generate a 1024 bit password protected private key file:

```
matrixRSAkeygen -out privkey.pem -pass password
```

Generate a 2048 bit password protected private key file:

```
matrixRSAkeygen -out privkey.pem -keysize 2048 -pass password
```

4 EC Key Generation

The `matrixECkeygen` utility is used to generate Elliptic Curve private key files. Private keys are the foundation for public-key cryptography and **must never be shared**. It is recommended the `-pass` option be used in the command line to encrypt the private key.

The following prime r1 NIST curves are supported and may be used as the "curve" parameter:

secp192r1

secp224r1

secp256r1

secp384r1

secp521r1

No configuration files are used for this utility.

4.1 Usage

```
matrixECkeygen -out filename -curve <curve> [-outform type] [-pass password]
```

out	Required. Identifies the file name of the new private key file. If omitted, the user will be prompted for a file name.
curve	Required. The curve on which the private key will be created. Valid values are listed above
outform	Optional. Either `PEM` (default) or `DER`. It is not possible to password protect DER formatted files in PKCS#1 so this format is not recommended.
pass	Optional. Allows the user to generate an encrypted file using the PKCS#5 standard. Pass the plaintext password to this option. Password protecting private key files is strongly recommended.

4.2 Examples

```
matrixECkeygen -out privkey.pem -curve secp384r1
```

Generate a password protected private key file:

```
matrixRSAkeygen -out privkey.pem -curve secp256r1 -pass password
```

5 CERTIFICATE REQUESTS

The `matrixCertReq` utility is used to generate PKCS#10 certificate request files.

The typical way in which a certificate is created is for an entity to present a certificate request file to a Certificate Authority (CA). The certificate request is a standard format that contains only the public portion of the key (remember never to share a private key) along with some identification information about the requesting entity. The CA then verifies this information and issues a signed certificate to the requesting entity.

5.1 Usage

```
matrixCertReq -out filename -key privKey [-pass password] [-reqconf confFile]
```

out	Required. Identifies the file name of the new request file. If omitted, the user will be prompted for a file name.
key	Required. The existing private key file for the requesting entity. Only the public portion of the key will be included in the output as this request file will be submitted to a CA.
pass	Optional. If the privKey is password protected this option must be used to supply the password to the utility
reqconf	Optional. Identifies an existing configuration file that contains the Distinguished Name information and certificate extension information that will be used in the certificate request. If omitted, the user will be prompted for the basic information on the command line. Information on the configuration file format and contents can be found in the Configuration File section below.

5.2 Configuration File

The `reqconf` option will identify a configuration file that contains the Distinguished Name of the requesting entity as well as the hash strength for the signature algorithm and optional subjectAltName and other extensions. All the configuration entries mentioned in section 2.3 are supported. A basic configuration file would look like this:

```
commonName=”<string>”;
country=”<string>”;
organization=”<string>”;
organizationalUnit=”<string>”;
stateOrProvince=”<string>”;
locality=”<string>”;
algorithm=”<hashAlg>”;
subjectAltName-dNSName=”<string>”;
subjectAltName-otherName=”<hex oid>:<string>”
keyUsage=”keyCertSign”;
extendedKeyUsage=”serverAuth clientAuth”;
```

commonName	This should be set to the domain or IP address of the entity that will be using the certificate. The CA must verify the requesting entity has ownership of this domain when issuing a certificate.
organization	Organization name
country	Normally the two letter country code abbreviation (ie US)
organizationalUnit	Further classification of the department within the organization

stateOrProvince	State or Province
locality	Typically the city name
algorithm	Value of `sha1`, `sha256`, or `sha384`. SHA-1 is the default if not provided
subjectAltName	See section 2.2.1
keyUsage	See section 2.2.2
extendedKeyUsage	See section 2.2.3

5.3 Examples

Create a *req.pem* certificate request file with Distinguished Name information provided through the *reqConfig.txt* file:

```
matrixCertReq -out req.pem -key privkey.pem -pass asdf
        -reqconf reqConfig.txt
```

Create a *req.pem* certificate request file with Distinguished Name information provided through user inputs prompted on the console standard input:

```
matrixCertReq -out req.pem -key privkey.pem -pass asdf
```

6 CERTIFICATE GENERATION

The matrixCertGen utility is used to generate X.509 version 3 certificates. These certificates represent the public key portion of the key pair and also contain information about which CA the certificate was issued by, who the certificate was issued to, and other details about how the certificate is to be used.

The two types of certificates generated are **Self-Signed** certificates and **CA-Issued** certificates.

6.1 Self-Signed Certificate Authority Usage

A self-signed certificate is generated when a root CA certificate is needed. This type of certificate is at the top of a certificate hierarchy and has the authority to issue certificates.

```
matrixCertGen -out filename -certconf configFile -key privkeyfile [-pass pass]
```

out	Required. Identifies the file name of the newly generated certificate. If omitted, the user will be prompted for the file name on the command line.
key	Required. Specifies the private key file to be used for self-signing the generated cert.
pass	Required if the private key file has been password protected. Specify the password with this option.
certconf	Required. Specifies the file name of the self-signed certificate configuration file. If omitted, the user will be prompted for the necessary information on the command line. Information on the configuration file format and contents can be found in the section below.

6.2 Self-Signed Certificate Authority Configuration File

The `certconf` option will identify a configuration file that contains the Distinguished Name attributes as well as the certificate creation parameters and optional subjectAltName. The contents of the configuration file for this self-signed case must include all the following attributes:

```
ca=”1”;
serialNum=”<integer>”;
pathLen=”<integer>”;
validDays=”<integer>”;
algorithm=”<hashAlg>”;
country=”<string>”;
organization=”<string>”;
commonName=”<string>”;
organizationalUnit=”<string>”;
stateOrProvince=”<string>”;
locality=”<string>”;
keyUsage=”keyCertSign”;
subjectAltName-dNSName=”<string>”;
subjectAltName-otherName=”<hex oid>:<string>”
```

ca	A ‘0’ or ‘1’ value to indicate whether the certificate being created will be allowed to issue certificates itself. As this is a self-signed CA file, the value must be ‘1’
serialNum	The serial number to be given to this certificate. A database of serial numbers should be kept by the CA to aid in future
pathLen	An integer value that is only meaningful if the ‘ca’ attribute is set to ‘1’. The pathLen attribute specifies how long a certificate chain may be that originates with this certificate.
validDays	An integer value that specifies the number of days the certificate being generated will be valid for.
algorithm	Value of `sha1`, `sha256`, or `sha384`. SHA-1 is the default if not provided
commonName	This should be set to the domain or IP address of the entity that will be distributing the certificate, or an email address if the certificate is used for email. The CA must verify the requesting entity has ownership of this domain when issuing a certificate.
organization	Organization name
country	Normally the two letter country code abbreviation (ie US)
organizationalUnit	Further classification of the department within the organization
stateOrProvince	State or Province
locality	Typically the city name
keyUsage	MUST be “keyCertSign”
subjectAltName	See section 2.2.1

6.3 Self-Signed Certificate Authority Examples

Generate a *ssCA.pem* self-signed certificate file with Distinguished Name and certificate parameters provided by the *ssCA.conf* configuration file:

```
matrixCertGen -out ssCA.pem -certconf ssCA.conf -key privkey.pem
-pass asdf
```

Generate a *ssCA.pem* self-signed certificate file with Distinguished Name and certificate parameters provided through user inputs prompted on the console standard input:

```
matrixCertGen -out ssCA.pem -key privkey.pem -pass asdf
```

6.4 CA-Issued Certificate Usage

A CA-Issued certificate is the usual manner in which a certificate is generated. In this case, a certificate request file is passed on the command line and the private key information is from the CA itself.

```
matrixCertGen -out filename -req requestFile -certconf caConfFile
-CAcert CAcert -CAkey CAkey [-CApass password]
```

out	Required. Identifies the file name of the newly generated certificate. If omitted, the user will be prompted for the file name on the command line.
req	Required. Identifies the certificate request file that was presented by the requesting entity
CAcert	Required. Specifies a Certificate Authority file that will be used to issue this new certificate. If omitted, the user will be prompted for the location of CA file on the command line.
CAkey	Required. Specifies the CA private key file to be used for signing the generated cert.
CApass	Required if the CA private key file has been password protected. Specify the password with this option.
certconf	Required. Specifies the file name of the CA-Issued certificate configuration file. If omitted, the user will be prompted for the necessary information on the command line. Information on the configuration file format and contents can be found in the Configuration Files section below.

6.5 CA-Issued Certificate Configuration File

The contents of the configuration file for the CA issued case will only include the attributes related to the certificate parameters. The Distinguished Name information is being provided through the Certificate Request file. So, the configuration file will only contain only the following required attributes:

```
serialNum=”<integer>”;
pathLen=”<integer>”;
validDays=”<integer>”;
algorithm=”<hashAlg>”;
ca=”1”;
keyUsage=”keyCertSign”;
subjectAltName-dNSName=”<string>”;
subjectAltName-otherName=”<hex oid>:<string>”;
```

ca	A ‘0’ or ‘1’ value to indicate whether the certificate being created will be allowed to issue certificates itself. As this is a self-signed CA file, the value must be ‘1’
keyUsage	If ca is ‘1’ then this MUST be set to ‘keyCertSign’
serialNum	The serial number to be given to this certificate. A database of serial numbers should be kept by the CA to aid in future
pathLen	An integer value that is only meaningful if the ‘ca’ attribute is set to ‘1’. The pathLen attribute specifies how long a certificate chain may be that originates with this certificate.
validDays	An integer value that specifies the number of days the certificate being generated will be valid for.
algorithm	Value of `sha1`, `sha256`, or `sha384`. SHA-1 is the default if not provided
subjectAltName	See section 2.2.1. NOTE: If the certificate request has a valid subjectAltName extension the CA-issuer configuration file may not use a subjectAltName as well.

Certifcate requests may contain the additional extensions that it wishes to have in the final issued certificate. If so, the CA must choose to allow all the extensions that appear. If the CA configuration file contains extensions that are already present in the certificate request, the certificate request will take precedence.

6.6 CA-Issued Certificate Example

```
matrixCertGen -out cert.pem -req req.pem -certconf caConf.txt -CAcert ssCA.pem
      -CAkey CAprivkey.pem -CApass asdf
```

7 PEM FILE TO HEADER FILE CONVERSION

The matrixPem2Mem utility is used to convert private key and certificate files into a C language header file for source level inclusion of private key or certificate information. These in-memory versions of the key material can be used in platforms that do not include file system support.

7.1 Private Key Usage

```
matrixPem2Mem -key privkey.pem [-pass password]
```

key	Required. Identifies the file name of the private key file.
pass	Optional. The password if the private key file is password protected.

7.2 Certificate File Usage

```
matrixPem2Mem -cert cert.pem
```

cert	Required. Identifies the file name of the certificate to convert