How to Upgrade Legacy Systems with Elliptic Curve Cryptography

Use Case: Using a Hybrid Public Key Infrastructure to Migrate Web Applications to Suite B

A Certicom Technology Brief
February 2008

Table of Contents

certicom™

Executive Summary

The National Security Agency's (NSA) decision to augment the Advanced Encryption Standard with Suite B Cryptography propelled Elliptic Curve Cryptography (ECC) for digital signatures to the forefront of methods used to protect classified and unclassified national security systems. NSA announced Suite B in 2005, after it licensed the core Intellectual Property from Certicom. Since then, leading government agencies and global companies have started to adopt this stronger level of security.

Many standards for ECC algorithms originated in the decade prior to the NSA's announcement, including IEEE 1363[1] in 1996 and ANSI X9.62-1998.[2] In 1998 the SECG[3] established standards for digital signature algorithms and elliptic curve parameters. Implementations of those algorithms began to materialize following the publication of these draft standards. While these standards made it theoretically possible to create a next generation ECC-based Public Key Infrastructure (PKI), the complete set of solution elements required by an ECC PKI became available only recently.

The critical purpose of the PKI is to verify and authenticate the validity of each party involved in a communication using a trusted third-party, called a Certificate Authority (CA). Also known as a trust hierarchy, the PKI is a system of digital certificates and the CAs that issue them. The concept itself is not unique to ECC as the PKI was also a vital component of earlier public key cryptography techniques using RSA.[4] In fact, the X.509[5] standard governing certificate formats dates back 20 years. The legacy PKIs that support existing applications are predominantly RSA-based, as is the massive installed base of end systems that rely on them.

Upgrading existing systems to Suite B requires the introduction of a next generation ECC PKI. As is often the case when upgrading the core of a communications infrastructure, legacy issues present their hurdles. And clearly a smooth migration path to the next generation greatly affects the success of the new technology.

Through a secure web application example, this paper outlines a "hybrid" migration strategy in which both legacy RSA PKIs and next generation ECC PKIs can interoperate in the same network, allowing for a manageable, gradual transition to Suite B.

[1] Institute of Electrical and Electronics Engineers (IEEE) 1363 *Standard Specifications For Public-Key Cryptography*
[2] American National Standards Institute (ANSI) X9.62-1998 *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*
[3] Standards for Efficient Cryptography Group (SECG) *SEC 1: Elliptic Curve Cryptography, and SEC 2: Recommended Elliptic Curve Domain Parameters*
[4] A public-key encryption technology developed by RSA Data Security, Inc. RSA stands for Rivest, Shamir, and Adelman, the inventors.
[5] X.509 (Version 1) was first issued in 1988 as a part of the ITU X.500 Directory Services standard.

certicom™

Available Products Supporting Suite B

Upgrading existing systems to use Suite B Cryptography requires the convergence of a number of ECC-based system elements. These include: a CA, the applications capable of creating certificate requests, and end-user application support for ECC. Until recently several key components were missing that made it extremely difficult, if not impossible, to deploy ECC-based PKIs.

Fortunately, 2007 was a pivotal year that saw the introduction of commercial applications and open systems supporting ECC. These applications place developers and IT managers within reach of an upgrade path to Suite B. The table below shows a selection of system elements that are currently available with support for Suite B Cryptography. These products will be further explored in later sections of this paper.

CERTIFICATE AUTHORITIES	APPLICATIONS CAPABLE OF CREATING AND REQUESTING CERTIFICATES	WEB SERVERS	END-USER APPLICATIONS CAPABLE OF VALIDATING CERTIFICATES
Certicom Device CA	Certicom ECC Certificate Utility	Apache HTTP Server	Certicom Suite B Browser
EJBCA	Microsoft Windows Server 2008	Certicom Suite B Web Server	Firefox Browser
Microsoft Windows Server 2008[6]	Netscape Network Security Services (NSS)	Microsoft Windows Server 2008	Microsoft Internet Explorer
	OpenSSL Certificate Tool		

Table 1. Suite B-enabled system elements

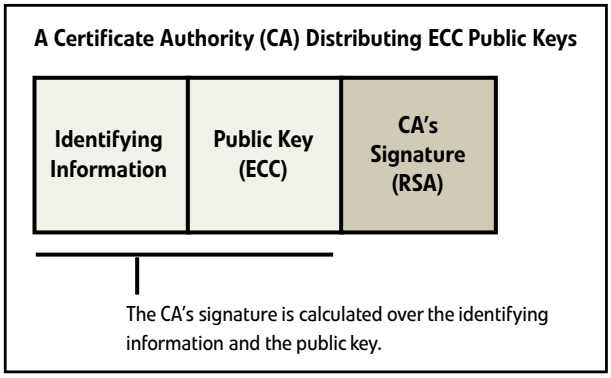
Managing Legacy and Next Generation PKIs

The usual method of managing public keys in networks of any scale is a CA - a trusted central server with a well-known public key that issues digital certificates. The role of the CA in this process is to guarantee that the individual granted the unique certificate is, in fact, the individual or entity it claims to be. Existing CAs (such as VeriSign) typically use legacy RSA keys to sign the certificate. In next generation PKIs, ECC keys are used instead.

Eventually all CAs will need to embrace ECC in order to support evolving ANSI, IETF, IEEE, and SECG standards. Since CAs are critical components of an organization's security infrastructure - and since the distribution of the CA's public key is a vital to making that CA useful, merely rolling out a new CA with an ECC key pair provides significant logistical hurdles. Therefore instead of wholesale replacement, what's needed is a smooth migration path from legacy PKIs to the next generation.

[6] Codename Longhorn

Fortunately, a bridge exists that allows legacy CAs with non-ECC public keys to distribute ECC public keys. This bridging technology is the *hybrid certificate*. A hybrid certificate is one that binds an ECC public key to an end entity, but which is itself signed using the CA's private key and non-ECC signature algorithms (i.e., RSA).



Hybrid certificates can be verified by any end entity which already has secure knowledge of the CA's existing public key - say a browser using certificates to verify the identities of web servers - without the need to distribute a new public key for the CA, and ECC public keys can be associated with any communicating party - without having to replace the CA.

In addition to enabling ECC security in the short term by leveraging the legacy PKI, the hybrid certificate can also provide additional benefit down the road when the time comes to upgrade the legacy CA to the next generation PKI. For an interim period, the upgraded CA could simultaneously support legacy and ECC-enabled clients in one of two ways. It could be a dual CA capable of issuing certificates signed by either RSA or ECC, or it could sign all certificates with ECC, including those issued to legacy clients. In the latter case, those certificates would be another form of hybrid. . Eventually, as the legacy clients are either updated to next generation PKI CA certificates or retired, the RSA certificate will expire and the CA need only issue ECC-signed certificates thereafter. Thus the use of hybrid certificates provides a smooth and gradual migration path to ECC both in the short and long term.

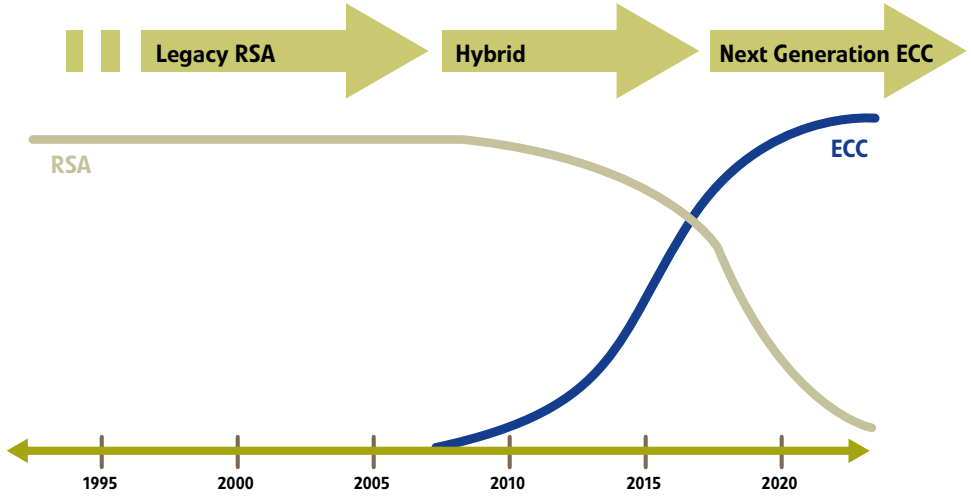


Figure 2. Migration from legacy RSA to next generation ECC using hybrid certificates

The process for obtaining and generating hybrid certificates is explained on page 6.

Application Use Case – Web Security

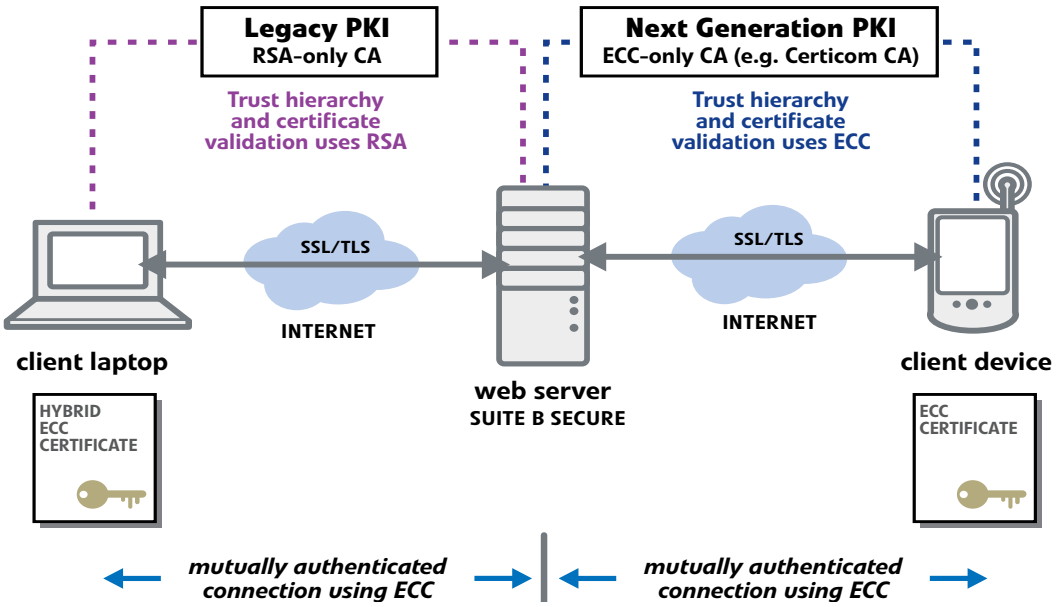


Figure 3. Interoperability of legacy and next generation PKIs

Illustrated above is an example of the interoperability of a legacy PKI with a next generation PKI. The wireless network on the right uses a next generation PKI. Size- and power-constrained handheld devices particularly benefit from ECC and the rapid pace of new gadget introductions quickly displaces older units, making this fertile ground for widespread ECC adoption. Contrasted at the left is a PC-server network categorized by a much larger installed base of legacy systems. Even today that includes Windows 98 clients! Thus the legacy PKI that's needed to support this base will remain in place for some time. Fortunately legacy systems, i.e. servers and client PCs, can be upgraded to ECC by leveraging the legacy PKI – with no need to wait for the next generation PKI.

The previous section discusses how hybrid certificates provide a means of integrating ECC within a legacy PKI. In the above illustration, we show the presence of the hybrid certificate on the laptop client, so what's needed to complete the picture is an ECC upgrade path for the existing client and server elements. The table below lists available web server and browser software that is Suite B enabled. By upgrading existing servers and clients with one of these products, and deploying hybrid certificates, Suite B security for existing web applications becomes a reality today and a migration path is established for the eventual upgrade to the next generation PKI in the future.

SERVER PRODUCT	OVERVIEW
Apache HTTP Server Source: Apache Software Foundation	Delivers ECC support when combined with mod_ssl and various patches.
Certicom Suite B Web Server Source: Certicom Corp.	Ceritcom delivers a plug-and-play module that makes Apache servers completely Suite B and FIPS 140-2 compliant. Certicom's Suite B Web Security Power Bundle also includes optimized tools that make communications over key protocols, i.e. TLS, Suite B compliant.
Microsoft Windows Server 2008 Source: Microsoft Corp.	Microsoft's latest server package includes Suite B enabled crypto and PKI platforms that support the ECC and the SHA-2 hashing algorithm family that can enroll and validate certificates.

BROWSER PRODUCT	OVERVIEW
Certicom Suite B Browser Source: Certicom Corp.	Certicom provides a plug-and-play module that makes the Firefox browser completely Suite B and FIPS 140-2 compliant. The browser module is another component of the Certicom Suite B Web Security Power Bundle mentioned above.
Firefox Browser Source: Mozilla	v2.0 supports Suite B as a standard feature.
Microsoft Internet Explorer Source: Microsoft Corp.	IE v7.0 supports Suite B as a standard feature.

Table 2. Suite B enabled web servers and browsers.

There is a small caveat here. Until the TLS 1.2 specification is finalized, all Suite B TLS connections cannot operate in Suite B mode. Over time product updates will bring various products into compliance.

How to Obtain a Hybrid Certificate

The process for creating and deploying a hybrid certificate is described below. Note that this process actually applies to any type of certificate request. A hybrid certificate is generated when the request includes an ECC key pair and the CA signs it with a non-ECC algorithm.

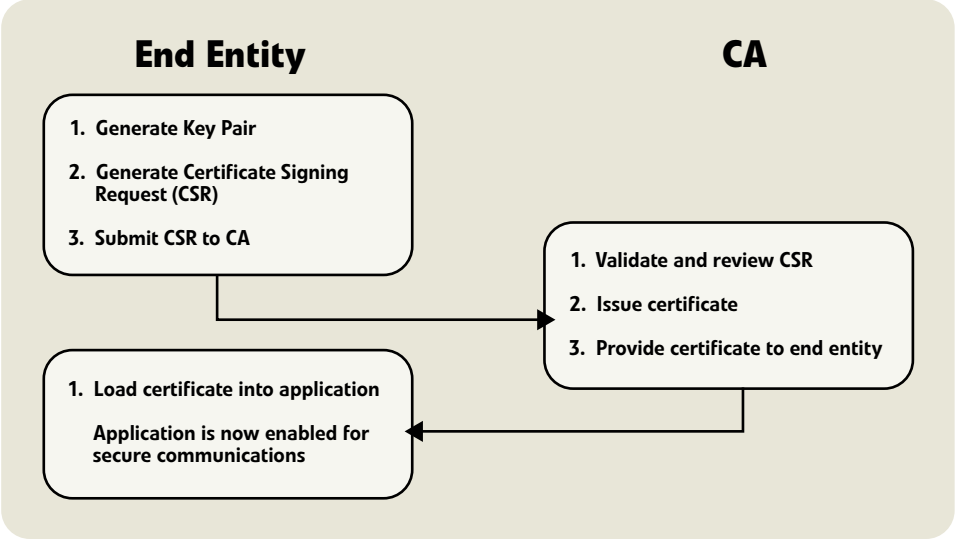


Figure 4. Process for obtaining a certificate from a CA

End Entity Submits Certificate Signing Request (CSR) to CA

1. Generate a key pair

The key pair consists of a public and private key. Both of these keys will be used in the creation of a CSR that will be submitted to the CA. The requestor maintains the secrecy of the private key. If the key pair is ECC and the CA is legacy, the end result is a hybrid certificate.

2. Generate a CSR

The CSR is a file containing the end-entity's public key, a Distinguished Name, plus additional attributes required by the CA. The Distinguished Name is that by which the end-entity is known and may comprise several different parts. It could be a person's common name (e.g., John Doe) or the fully qualified domain name of a server (e.g., www.certicom.com). The remaining set of required CSR attributes may include items like company name, organizational unit, city, province and country. These are spelled out in the CA's Certificate Practice Statement (CPS), which is the CA's published description of its policies and process for issuing certificates.

The CSR must be signed using the end entity's private key. This signature along with the public key enables the CA to determine that the request has not been tampered with and that the requesting entity is in possession of the private key. Possession of the private key is proven whenever the signature can be validated using the requesting entity's public key.

3. Submit CSR to CA

The manner in which the end entity submits the CSR varies by CA. In some cases the CSR is pasted into a form on a web site.

CA Issues Certificate

1. Validate and review the CSR

In accordance with its CPS, the CA performs some form of validation on the CSR, then reviews it and determines whether it will fulfill the request. In some cases, a certificate may simply be issued once the request is received and validated. Some requests may require verbal confirmation of the location and name of the end entity, and in yet other cases backup legal documentation may need to be provided.

2. Issue the end entity certificate

Issuing the certificate binds the identity of the end entity to the public key contained in the certificate. This binding occurs through the use of the CA's private key and self-signed certificate. A self-signed certificate is used because it sits at the root of the trust hierarchy created by the CA. The CA's private key is used to sign the certificate and the certificate can be validated using the CA's public key.

3. Provide certificate to end entity

The certificate is in the form of a PEM or PFX file containing one or more X.509 certificates. Its format is defined in PKCS #12 (Personal Information Exchange Syntax Standard), for the secure transport of personally identity information including certificates and private keys. This file is provided to the end entity by some means such as making it available for download from a web server.

End Entity Secures Application with Certificate

1. Load the end entity certificate into the application

For the end entity certificate to be useful it must be loaded into the browser, email client, or web server application. Both the certificate and its associated private key are loaded. The manner in which this is done can vary considerably depending on the application. For example, both could be pasted into a dialog box presented by the application or they could be loaded separately by the application.

For example, the Firefox Browser supports the import of the end entity certificate and private key as a single file using the command:

Edit > Preferences > Advanced > View Certificates > Import.

Once the certificate is loaded, the application can use it to authenticate and encrypt communications between itself and a remote peer.

certicom™

Tools For Generating Certificates

The following table lists available tools that can be utilized to create hybrid certificates. Actual step-by-step examples of each tool's usage are detailed in the appendix.

PRODUCT	OVERVIEW
Certicom ECC Certificate Utility Source: Certicom Corp.	Certicom's solution makes generating ECC certificates easy and enables developers to quickly install certificates on clients and servers. Users can create an ECC root and upgrade to an ECC-centric infrastructure or easily create hybrid certificates that combine the ECC public key algorithm with the RSA signature. For maximum flexibility, The Certicom ECC Certificate Utility supports multiple X.509 certificate types, encoding formats and key types. Identities, files, and certificate chains can be converted between PFX, Base-64, and binary formats.
OpenSSL Certificate Tool Source: The OpenSSL Project	The OpenSSL Project is a collaborative effort to develop an open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a general purpose cryptography library.
Microsoft Windows Server 2008 (codename Longhorn) Source: Microsoft Corp.	Microsoft's latest server package includes Suite B enabled crypto and PKI platforms that support the ECC and the SHA-2 hashing algorithm family that can enroll and validate certificates.
Netscape Network Security Services (NSS) Source: Mozilla	Network Security Services (NSS) is a set of libraries designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSL v2 and v3, TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, X.509 v3 certificates, and other security standards
Certicom Device CA[7] Source: Certicom Corp.	Certicom Device Certificate Authority (CA) is a managed service offering that fulfills customer requirements for large quantities of certificates. Operated from a secure Certicom site, the service provides subscribers with a complete system for managing, creating, issuing, publishing and revoking public key certificates. It supports X.509 RSA and ECC certificates as well as hybrid certificates. Primarily serving customers with high volume device and/or software manufacturing and distribution applications, it is an offline, batch-oriented system capable of delivering one million X.509 ECDSA certificates per day up to and including 384-bit curves (benchmarked using the secp384r1 curve).

[7] Certicom Device CA is not shown in the step-by-step usage examples since it is an offline service and there is no user-visible GUI.

certicom™

Conclusion

Existing systems can be upgraded to Suite B Cryptography today without the disruption of a wholesale replacement of the legacy RSA-based CA with a next generation ECC-based CA. This migration can be accomplished by upgrading the systems to ECC and by leveraging the legacy PKI through the use of hybrid certificates. Using hybrid certificates can provide the additional benefit of a smooth future transition to the next generation PKI when the legacy PKI is ultimately phased out.

A number of ECC-enabled CAs, the tools for creating hybrid certificates, and Suite B- enabled web server and browser products are recently available from Certicom and others. The combination of these solutions is now enabling developers and IT managers to upgrade existing systems to Suite B level security in a straightforward manner.

certicom™

Appendixes – Using the Tools to Create CAs and Generate Certificates

Appendix A – Certicom ECC Certificate Utility

Here are the steps to generate RSA and ECC CAs, and RSA, ECC, and hybrid entity certificates using Certicom's ECC Certificate Utility (v1.0.0).

Certicom's ECC Certificate Utility is available from **http://www.certicom.com**. (license fee required).

Create an RSA certificate authority

1. Generate the CA keypair

```
# eccutility rsa -kl 1024 -p password -privo rsaca_priv.pem -pubo rsaca_pub.pem

rsa - generating 1024 bit RSA key pair...
rsa - encryption private key with password: 'password'
rsa - writing private key to file: rsaca_priv.pem
rsa - writing public key to file: rsaca_pub.pem
```

2. Generate a certificate request

```
# eccutility pkcs10 -pubi rsaca_pub.pem -privi rsaca_priv.pem -p password -dn "CN=Sample RSA Certificate Authority"
-reqo rsaca_req.pem

pkcs10 - using password: password
pkcs10 - reading private key from file: rsaca_priv.pem
pkcs10 - reading public key from file: rsaca_pub.pem
pkcs10 - using subject name: CN=Sample RSA Certificate Authority
pkcs10 - writing certificate request to file: rsaca_req.pem
```

3. Generate the CA certificate

```
# eccutility cert -privi rsaca_priv.pem -p password -reqi rsaca_req.pem -ct:ca 0 -certo rsaca_cert.pem

cert - no issuer name (-certi or -in): self-signed
cert - using password: "password"
cert - reading private key from file: rsaca_priv.pem
cert - reading PKCS#10 request from file: rsaca_req.pem
cert - using issuer name:
  CN=Sample RSA Certificate Authority
cert - using subject name:
  CN=Sample RSA Certificate Authority
cert - writing certificate to: rsaca_cert.pem
```

Create an ECC certificate authority

1. Generate the CA keypair

```
# eccutility ecc -c secp256r1 -p password -privo eccca_priv.pem -pubo eccca_pub.pem -uncomp

ecc - generating key pair on curve secp256r1
ecc - using password 'password'
ecc - writing private key to file 'eccca_priv.pem'
ecc - writing public key to file 'eccca_pub.pem'
```

2. Generate a certificate request

```
# eccutility pkcs10 -pubi eccca_pub.pem -privi eccca_priv.pem -p password -dn "CN=Sample ECC
Certificate Authority" -reqo eccca_req.pem -uncomp

pkcs10 - using password: password
pkcs10 - reading private key from file: eccca_priv.pem
pkcs10 - reading public key from file: eccca_pub.pem
pkcs10 - using subject name: CN=Sample ECC Certificate Authority
pkcs10 - writing certificate request to file: eccca_req.pem
```

3. Generate the CA certificate

```
# eccutility cert -privi eccca_priv.pem -p password -reqi eccca_req.pem -ct:ca 0 -certo eccca_cert.pem
-uncomp

cert - no issuer name (-certi or -in): self-signed
cert - using password: "password"
cert - reading private key from file: eccca_priv.pem
cert - reading PKCS#10 request from file: eccca_req.pem
cert - using issuer name:
  CN=Sample ECC Certificate Authority
cert - using subject name:
  CN=Sample ECC Certificate Authority
cert - writing certificate to: eccca_cert.pem
```

certicom™

Create a pure RSA entity certificate

1. Generate the entity keypair

```
# eccutility rsa -kl 1024 -p password -privo rsaentity_priv.pem -pubo rsaentity_pub.pem

rsa - generating 1024 bit RSA key pair...
rsa - encryption private key with password: 'password'
rsa - writing private key to file: rsaentity_priv.pem
rsa - writing public key to file: rsaentity_pub.pem
```

2. Generate a certificate request

```
# eccutility pkcs10 -pubi rsaentity_pub.pem -privi rsaentity_priv.pem -p password -dn "CN=Sample RSA Entity" -reqo
rsaentity_req.pem

pkcs10 - using password: password
pkcs10 - reading private key from file: rsaentity_priv.pem
pkcs10 - reading public key from file: rsaentity_pub.pem
pkcs10 - using subject name: CN=Sample RSA Entity
pkcs10 - writing certificate request to file: rsaentity_req.pem
```

3. Generate the entity certificate

```
# eccutility cert -reqi rsaentity_req.pem -certi rsaca_cert.pem -privi rsaca_priv.pem -p password -ct:tls www.example.com
-certo rsaentity_cert.pem

cert - reading signer certificate from file: rsaca_cert.pem
cert - using password: "password"
cert - reading private key from file: rsaca_priv.pem
cert - reading PKCS#10 request from file: rsaentity_req.pem
cert - using issuer name:
  CN=Sample RSA Certificate Authority
cert - using subject name:
  CN=Sample RSA Entity
cert - writing certificate to: rsaentity_cert.pem
```

Create a pure ECC entity certificate

1. Generate the entity keypair

```
# eccutility ecc -c secp256r1 -p password -privo eccentity_priv.pem -pubo eccentity_pub.pem -uncomp

ecc - generating key pair on curve secp256r1
ecc - using password 'password'
ecc - writing private key to file 'eccentity_priv.pem'
ecc - writing public key to file 'eccentity_pub.pem'
```

2. Generate a certificate request

```
# eccutility pkcs10 -pubi eccentity_pub.pem -privi eccentity_priv.pem -p password -dn "CN=Sample ECC Entity" -reqo
eccentity_req.pem -uncomp

pkcs10 - using password: password
pkcs10 - reading private key from file: eccentity_priv.pem
pkcs10 - reading public key from file: eccentity_pub.pem
pkcs10 - using subject name: CN=Sample ECC Entity
pkcs10 - writing certificate request to file: eccentity_req.pem
```

3. Generate the entity certificate

```
# eccutility cert -reqi eccentity_req.pem -certi eccca_cert.pem -privi eccca_priv.pem -p password -ku ds -ct:tls www.
example.com -certo eccentity_cert.pem -uncomp

cert - reading signer certificate from file: eccca_cert.pem
cert - using password: "password"
cert - reading private key from file: eccca_priv.pem
cert - reading PKCS#10 request from file: eccentity_req.pem
cert - using issuer name:
  CN=Sample ECC Certificate Authority
cert - using subject name:
  CN=Sample ECC Entity
cert - writing certificate to: eccentity_cert.pem
```

certicom™

Create a hybrid ECC/RSA entity certificate

1. Generate the entity keypair

```
# eccutility ecc -c secp256r1 -p password -privo hybridentity_priv.pem -pubo hybridentity_pub.pem -uncomp

ecc - generating key pair on curve secp256r1
ecc - using password 'password'
ecc - writing private key to file 'hybridentity_priv.pem'
ecc - writing public key to file 'hybridentity_pub.pem'
```

2. Generate a certificate request

```
# eccutility pkcs10 -pubi hybridentity_pub.pem -privi hybridentity_priv.pem -p password -dn "CN=Sample ECC/RSA Hybrid Entity" -reqo hybridentity_req.pem -uncomp

pkcs10 - using password: password
pkcs10 - reading private key from file: hybridentity_priv.pem
pkcs10 - reading public key from file: hybridentity_pub.pem
pkcs10 - using subject name: CN=Sample ECC/RSA Hybrid Entity
pkcs10 - writing certificate request to file: hybridentity_req.pem
```

3. Generate the entity certificate

```
# eccutility cert -reqi hybridentity_req.pem -certi rsaca_cert.pem -privi rsaca_priv.pem -p password -ku ds -ct:tls www.example.com -certo hybridentity_cert.pem -uncomp

cert - reading signer certificate from file: rsaca_cert.pem
cert - using password: "password"
cert - reading private key from file: rsaca_priv.pem
cert - reading PKCS#10 request from file: hybridentity_req.pem
cert - using issuer name:
  CN=Sample RSA Certificate Authority
cert - using subject name:
  CN=Sample ECC/RSA Hybrid Entity
cert - writing certificate to: hybridentity_cert.pem
```

Check the validity of the certificates

```
# eccutility validate -certi rsaentity_cert.pem -trusti rsaca_cert.pem
# eccutility validate -certi eccentity_cert.pem -trusti eccca_cert.pem
# eccutility validate -certi hybridentity_cert.pem -trusti rsaca_cert.pem
```

The only warning displayed should be CRL_NOT_FOUND. If the certificate is not valid you will see other warnings, such as NO_TRUSTED.

You can rename these CA and entity certificate extensions from .pem to .cer extension and Microsoft Windows will happily recognize them.

Appendix B – OpenSSL Certificate Tool

Here are the steps to generate RSA certificate authorities, and hybrid certificates using an OpenSSL Snapshot (openssl-0.9.8-stable-SNAP-20070810), a successor of OpenSSL 0.9.8e.

The snapshot was compiled following the Quick Start instructions in the INSTALL file provided with the snapshot.

OpenSSL and OpenSSL snapshots are available from http://www.openssl.org/source/ and ftp://ftp.openssl.org/snapshot/. Snapshots are updated daily and may not always compile.

Create an RSA certificate authority

1. Generate the CA keypair

```
# openssl genrsa -des3 -out ca.key 2048

Generating RSA private key, 2048 bit long modulus
.............+++
....+++
e is 65537 (0x10001)
Enter pass phrase for ca.key:
Verifying - Enter pass phrase for ca.key:
```

2. Generate the CA certificate

```
# openssl req -new -x509 -days 365 -key ca.key -out ca.crt -config openssl.cnf

Enter pass phrase for ca.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) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:My CA Internet Widgits CA
Email Address []:
```

certicom™

Create a hybrid certificate

1. Generate the entity keypair

```
# openssl ecparam -out eckey.pem -name sect163k1 -genkey
```

2. Generate a certificate request

```
# openssl req -new -key eckey.pem -out myreq.csr -config openssl.cnf

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) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:me@certicom.internetwidgits.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Generate the entity certificate

```
# openssl ca -config openssl.cnf -keyfile ca.key -cert ca.crt -in myreq.csr -out mycert.crt

Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
      Serial Number: 287 (0x11f)
      Validity
         Not Before: Aug 10 14:04:58 2007 GMT
         Not After : Aug  9 14:04:58 2008 GMT
      Subject:
         countryName               = AU
         stateOrProvinceName       = Some-State
```

```
        organizationName        = Internet Widgits Pty Ltd
        commonName              = me@internetwidgits.com
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            8E:70:24:99:42:6D:2B:A6:0C:DC:C5:84:CA:5E:FC:4B:58:F9:25:C3
        X509v3 Authority Key Identifier:
            keyid:31:B9:58:D2:3F:79:49:CD:AB:5F:1C:12:34:21:43:87:F1:1C:C6:2C

Certificate is to be certified until Aug  9 14:04:58 2008 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

4. Verify the end entity certificate

In the first case, ECC support is enabled in the application performing the validation.

```
# openssl verify -CAfile ca.crt mycert.crt
mycert.crt: OK
```

Repeat the same test for the second case where ECC support is disabled.

```
# ./openssl verify -CAfile ca.crt mycert.crt
mycert.crt: OK
```

The verification succeeds in both cases because only RSA is needed to verify the trust path.

This command validates the statement that no ECC support exists in the
second version of OpenSSL.

```
# ./openssl ecparamopenssl:Error: 'ecparams' is an invalid command.
```

Appendix C – Microsoft Windows Server 2008

Microsoft's Windows Server 2008 (codename Longhorn) CA (Longhorn Evaluation Copy, Build 6001) provides the ability to generate ECC and hybrid certificates. To enable the CA the Longhorn server must have the Active Directory Certificate Services enabled as a standalone CA. This build of Longhorn does not permit the enrollment of end entities relying upon ECC or hybrid certificates through the Web Enrollment Interface (requires IIS 7.0 with SSL bindings enabled).

The use of a standalone CA removes the requirement for Active Directory Domain Services (ADDS). ADDS stores directory data and manages communication between users and domains, including user logon processes, authentication, and directory searches. Refer to the Microsoft documentation describing this service to determine whether its presence or absence is suitable in the environment that the CA is being deployed in.

To use Longhorn with ECC or hybrid certificates, PKCS 7 requests must be created using other certification requests tools. Suitable tools include Certicom's ECC Certificate Utility and OpenSSL's Certificate Tool.

To create an ECC CA, generate a Root CA to create a new private key. Select the EC cryptographic service provider. Choices are ECDSA_P256, ECDSA_P384, and ECDSA_P521 (all Suite B curves). Select an appropriate secure hash algorithm. Choices are SHA-1, SHA-256, SHA-384, and SHA-256. The test CA created here used ECDSA_P256 and SHA-256.

Once the CA is created, certification requests can be accepted. Using OpenSSL, a certification request can be performed as follows:

Create a hybrid certificate

Using any machine:

1. Generate the entity keypair

```
# openssl ecparam -out eckey.pem -name sect163k1 -genkey
```

2. Generate a certificate request

```
# openssl req -new -key eckey.pem -out myreq.csr -config openssl.cnf

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) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
```

```
Common Name (eg, YOUR name) []:me@certicom.internetwidgits.com
Email Address []:

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Generate the entity certificate

One the Longhorn CA:

3a. Using the command tool (Start->Run "cmd")

```
# certreq myreq.csr
```

3b. Using the Certificate Authority Tool (Start->Administrative Tools->Certification Authority)

Review the pending requests queue and issue the certificate (via the Actions menu).

Appendix D – Netscape Network Security Services (NSS)

Here are the steps to generate RSA and ECC CAs and RSA, ECC,
and hybrid entity certificates using Netscape's Network Security Services (NSS) (v3.11.7). This version of NSS was patched as described here:

http://www.mail-archive.com/dev-tech-crypto@lists.mozilla.org/msg01269.html

Without this patch you cannot generate an ECC CA cert. Do not build
with NSS_ECC_MORE_THAN_SUITE_B. Use the following compiler flags:

```
# gmake XCFLAGS="-DNSS_ENABLE_ECC" NSS_ENABLE_ECC=1 BUILD_OPT=1 nss_build_all
```

Network Security Services are available from
http://www.mozilla.org/projects/security/pki/nss/.

Create the Necessary Data Bases

Step 1. Create directory for databases

```
$ mkdir test
```

Step 2. Create security database, but not cert database.

```
$ modutil -create -dbdir test -nocertdb
```

Step 3 create cert database

```
$ certutil -d test -N
```

Create an ECC certificate authority

Step 4: Create self-signed ECC CA cert using SHA-256 here but SHA-1 can be used as well. Use a noise file to avoid entering entropy manually.

```
$ certutil -S -k ec  -q nistp256 -n mynistp256CA -s "CN=Self CA,OU=certicom.com,C=CA" -x -d test -t "CT,CT,CT" -Z SHA256
-m 101 -v 99 -5 -z noise

    Generating key.  This may take a few moments...

                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
```

```
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
5
                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
9
Is this a critical extension [y/N]?
y
```

Create an RSA certificate authority

Step 5: create self-signed RSA CA

```
$ certutil -S -k rsa  -n myrsaCA -s "CN=Self RSA CA,OU=certicom.com,C=CA" -x -d test -t "CT,CT,CT" -m 101 -v 99 -5 -z noise
```

Create a pure RSA entity certificate

Step 6: Create an RSA server cert

```
$ certutil -S -s "CN=Test RSA Server, O=Netscape, L=Mountain View, ST=California, C=US" -p "650-555-8888" -d test -c
myrsaCA -n myrsaserv -t "u,u,u" -m 102 -v 99 -5 -z noise

    Generating key.  This may take a few moments...

                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
1
                0 - SSL Client
                1 - SSL Server
```

```
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
9
Is this a critical extension [y/N]?
y
```

Create a hybrid certificate

Step 7: Create an EC RSA hybrid cert

```
$ certutil -S -s "CN=Test EC RSA Hybrid Server, O=Netscape, L=Mountain View, ST=California, C=US" -p "650-555-8888" -d
test -c myrsaCA -n myecrsaserv -t "u,u,u" -m 103 -v 99 -5 -z noise -k ec  -q nistp256

    Generating key.  This may take a few moments...

                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
1
                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
9
Is this a critical extension [y/N]?
y
```

Create a pure ECC entity certificate

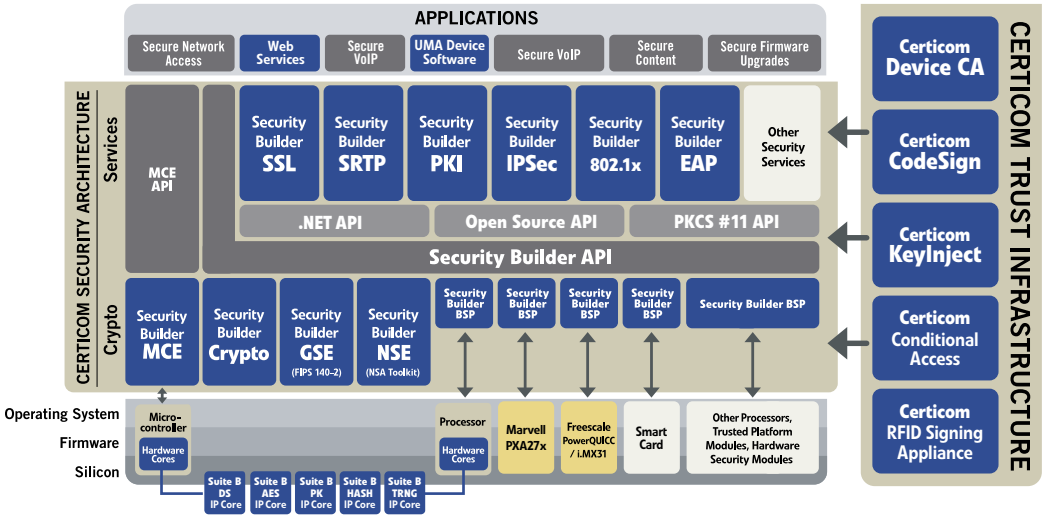
Step 8: Create an ECC server cert

```
$ certutil -S -s "CN=Test EC Server, O=Netscape, L=Mountain View, ST=California, C=US" -p
"650-555-8888" -d test -c mynistp256CA -n myecsrvr -t "u,u,u" -m 104 -v 99 -5 -z noise -k ec  -q nistp256

    Generating key.  This may take a few moments...

                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
    1
                0 - SSL Client
                1 - SSL Server
                2 - S/MIME
                3 - Object Signing
                4 - Reserved for future use
                5 - SSL CA
                6 - S/MIME CA
                7 - Object Signing CA
                Other to finish
    9
    Is this a critical extension [y/N]?
    y
```

About Certicom

Certicom protects the value of content, applications and devices with government-approved security. Adopted by the National Security Agency (NSA) for government communications, Elliptic Curve Cryptography (ECC) provides the most security per bit of any known public-key scheme. As the global leader in ECC, Certicom security offerings are currently licensed to more than 300 customers including General Dynamics, Motorola, Oracle, Research In Motion and Unisys. Founded in 1985, Certicom's corporate offices are in Mississauga, Ontario, Canada with worldwide sales and marketing headquarters in Reston, Virginia and offices in the U.S., Canada, Europe and China.

Visit www.certicom.com.



certicom™

Contact Certicom

Corporate Headquarters

5520 Explorer Drive, 4th Floor
Mississauga, Ontario
L4W 5L1
Tel: +1-905-507-4220
Fax: +1-905-507-4230
E-mail: info@certicom.com

Sales Offices

Worldwide Sales and Marketing Headquarters
1800 Alexander Bell Dr., Suite 400
Reston, Virginia 20190
Tel: 703-234-2357
Fax: 703-234-2356
E-mail: sales@certicom.com

U.S. Western Regional Office
393 Vintage Park Drive, Suite 260
Foster City, CA 94404
Tel: 650-655-3950
Fax: 650-655-3951
E-mail: sales@certicom.com

Europe
Golden Cross House
8 Duncannon Street
London WC2N 4JF UK
Tel: +44 20 7484 5025
Fax: +44 (0)870 7606778

Engelska Huset
Trappv 9
13242 Saltsjo-Boo
SWEDEN
Tel: +46 8 747 17 41
Mobile: +46 70 712 41 61
Fax: +46 708 74 41 61

www.certicom.com

Certicom White Papers

To read Certicom white papers, visit www.certicom.com/whitepapers.

Sum Total: Determining the True Cost of Security
Sourcing Security: Five Arguments in Favour of Commercial Security Solutions

Government

Making the Grade: Meeting Government Security Requirements (Suite B)
Meeting Government Security Requirements: The Difference Between Selling to the Government and Not
FAQ: The National Security Agency's ECC License Agreement with Certicom Corp.

Mobility

The Inside Story
Many Happy Returns: The ROI of Embedded Security
Welcome to the Real World: Embedded Security in Action

Sensor Networks

Securing Sensor Networks

DRM & Conditional Access

Injecting Trust to Protect Revenue and Reputation: A Key Injection System for Anti-Cloning, Conditional Access and DRM Schemes
Achieving DRM Robustness: Securing the Device from the Silicon Up to the Application(PDF)

Enterprise Software

Using Digital Signatures to cut down on Bank Fraud Loss

ECC

An Elliptic Curve Cryptography Primer
ECC in Action: Real World Applications of Elliptic Curve Cryptography
Using ECC for Enhanced Embedded Security (PDF)