# MatrixSSL Elliptic Curve Cipher Suites

# TABLE OF CONTENTS

# 1 ELLIPTIC CURVE CRYPTOGRAPHY

This document describes the use of Elliptic Curve Cryptography within the TLS protocol and how to utilize ECC cipher suites within MatrixSSL server and client applications.

ECC is a public key cryptographic algorithm that competes with RSA and Diffie-Hellman to perform key exchange and authentication. Key exchange is performed by ECC using a Diffie-Hellman variant and is abbreviated as ECDH. Authentication is performed by ECC using a Digital Signature Algorithm variant and is abbreviated as ECDSA. The appeal to some security implementers is that the mathematical properties of elliptic curves enable an equivalent strength security to RSA and DH while having smaller key sizes. This table can often be found in comparisons between the algorithms.

**Equivalent Strength of Key Sizes (bits)**

| ECC | RSA/DH |
|---|---|
| 163 | 1024 |
| 233 | 2048 |
| 283 | 3072 |
| 409 | 7680 |
| 571 | 15360 |

In authentication performance metrics head-to-head with RSA using the key sizes in the above table, the smaller key sizes of ECDSA translate into must faster key generation and slightly faster signature creation. However, RSA is much faster performing a signature validation operation.

In key exchange performance metrics head-to-head with DH, ECDH is always much higher performance. Guidelines on when to use ECC in TLS are given in the sections to follow.

The ECC algorithm must take into account the specific curve from which the keys were derived. MatrixSSL supports these NIST-recommended named prime field curves and Brainpool curves.

secp192r1
secp224r1
secp256r1
secp384r1
secp521r1
brainpool256r1
brainpool384r1
brainpool512r1

These curve restrictions apply to both the EC keys within the certificate material and for the ECDHE key generation.

# 2 ECC CIPHER SUITES

There are four different ECC cipher suite types available in the TLS protocol. They vary according to the cryptographic algorithms that will be used for key exchange and authentication (digital signature). Key exchange will always be ECDH but can either be ephemeral (ECDHE) or fixed DH. Ephemeral mode requires that new public key pairs be generated each connection, so they are typically much slower than fixed mode. Authentication can either be RSA or ECDSA. The choice of cipher suite will determine what types of certificate and keys must be loaded at application initialization. Each cipher suite type is outlined here.

## 2.1 Variations

### 2.1.1 ECDHE_ECDSA

Ephemeral ECDH key exchange with ECDSA signatures. These cipher suite types require ECC keys for the server as well as the signing Certificate Authority.

### 2.1.2 ECDHE_RSA

Ephemeral ECDH key exchange with RSA signatures. Because the key exchange is ephemeral these cipher suite types allow the user to use existing RSA keys and certificates on both the clients and servers.

### 2.1.3 ECDH_ECDSA

Fixed ECDH with ECDSA-signed certificates. These cipher suite types require ECC keys for the server as well as the signing Certificate Authority.

### 2.1.4 ECDH_RSA

Fixed ECDH with RSA-signed certificates require both types of public keys to be used. These cipher suite types require ECC keys for the server certificate but an existing RSA Certificate Authority will have used its RSA key to sign that certificate. The intent of this cipher suite is to enable existing trusted RSA CAs to remain in place while allowing servers to use the faster signature creation of ECC.

## 2.2 Support in MatrixSSL

The define `USE_ECC` must be enabled in the *cryptoConfig.h* header file to compile in Elliptic Curve cryptography support.

The user must also enable each of the ECC cipher suites that are desired. These defines are also listed in the *matrixsslConfig.h* file and are disabled by default. Below is a representative list of the available cipher suites. The full list can be found in the header file.

```
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
```

# 3 WHEN ECC CIPHER SUITES ARE A GOOD ALTERNATIVE

Assuming an equivalent key strength is used, here are some basic guidelines of what happens to performance when ECC ciphers suites are used instead of the traditional RSA suites.

1. If the application typically negotiates to an ephemeral DHE_RSA based suite without client authentication, both servers and clients will greatly benefit in switching to a ECDHE suite (either ECDHE_RSA or ECDHE_ECDSA) to take advantage of the much faster key generation.

2. If the application typically negotiates to a standard RSA suite without client authentication (RSA key exchange and authentication), servers will likely benefit by switching to an ECDH_ECDSA suite but the client will suffer decreased performance due to the slower signature validation.

3. If the application typically engages in client authentication handshakes a server will suffer greatly decreased performance due to the two signature validations during the CERTIFIATE and CERTIFICATE_VERIFY message parsing.

# 4 API

After enabling ECC cipher suites, the only API difference from the standard library is to use the EC specific certificate and key loading functions (`matrixSslLoadEcKeys` and `matrixSslLoadEcKeysMem`), documented in *MatrixSSL API* reference manual.