MatrixSSL Diffie-Hellman Cipher Suites

TABLE OF CONTENTS

1 ENABLING DIFFIE-HELLMAN IN MATRIXSSL

Diffie-Hellman is a key exchange algorithm that may be optionally included in the SSL protocol.

1.1 Raw Algorithm

The define `USE_DH` must be enabled in the *matrixsslConfig.h* header file to compile in Diffie-Hellman support.

1.2 Cipher Suites

The user must also enable any of the DH cipher suites that are desired. These defines are also listed in the *matrixsslConfig.h* file and are disabled by default. The list of supported cipher suites is:

TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBS_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA

TLS_DHE_PSK_WITH_AES_256_CBC_SHA
TLS_DHE_PSK_WITH_AES_128_CBC_SHA

TLS_DH_anon_WITH_AES_128_CBC_SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA
SSL_DH_anon_WITH_RC4_128_MD5
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA

Cipher suites prefixed with SSL also function with the TLS protocol. The cipher suites prefixed with TLS should only be used with the TLS protocol.

The DHE identifier refers to the fact that the RSA/DH combo suites are ephemeral. That is, new keys are generated for each connection. The trade-off for this 'perfect forward secrecy' is more CPU time.

The enabling of these defines is identical for both client and server targets of the MatrixSSL library.

2 CLIENT SIDE DIFFIE-HELLMAN

Once the MatrixSSL library is compiled with DH support and the desired cipher suites, there are no extra steps needed by developers for client-side applications. However, if the client wishes to force a DH cipher suite it may specify that explicitly in the `matrixSslNewClientSession` API using the `cipherSpec` parameter.

3 SERVER SIDE DIFFIE-HELLMAN

For server-side applications there is a very small amount of integration work that must be done to support DH. The server must load DH parameters at initialization from a PEM formatted file or through a memory location.

3.1 matrixSslLoadDhParams

```
int32 matrixSslLoadDhParams(sslKeys_t *keys, char *paramFile);
```

Parameter	Input/Output	Description
keys	input	Structure pointer for storing the DH key material that was previously allocated using `matrixSslNewKeys`
paramFile	input	A PEM encoded DH parameters file

Return Value	Description
PS_SUCCESS	Success. A valid key pointer will be returned in the keys parameter for use in a subsequent call to `matrixSslNewServerSession`
PS_MEM_FAIL	Failure. Unable to allocate memory for the structure
PS_PARSE_FAIL	Failure. Unable to parse DH parameter file

This function loads in a DH parameter file that is used for key generation when DH cipher suites are negotiated. The paramFile is a PEM formatted file that should include the standard `-----BEGIN DH PARAMETERS-----` header and `-----END_DH PARAMETERS-----` footer. Supported key sizes are 192, 512, 1024, 2048, and 4096 bit. DH parameter files may be uniquely generated, but this is not necessary and it is safe to use the DH parameter files provided in the MatrixSSL distribution.

RSA/DH Combination Cipher Suites

This API should be called along with `matrixSslLoadRsaKeys` for cipher suites that require RSA authentication and DH key exchange. The same `keys` parameter will be used for both function calls.

The DH parameters are added to the existing keys structure that will be passed into `matrixSslNewServerSession`. It is still possible to use an anonymous cipher suite in this usage scenario.

Anonymous DH and PSK Cipher Suites

The other use case for this API is to create a server application that supports anonymous DH cipher suites or Pre-shared key (PSK) cipher suites. Anonymous suites should never be used if authentication is required and no other mechanism is being used to perform authentication, but some implementations may require it. In this use case, it is not necessary to call `matrixSslLoadRsaKeys` with the `keys` structure.

3.2 matrixSslLoadDhParamsMem

```
int32 matrixSslLoadDhParamsMem(sslKeys_t *keys, unsigned char *dhBin,
      int32 dhBinLen);
```

Parameter	Input/Output	Description
keys	input	Structure pointer for storing the DH key material that was previously allocated using `matrixSslNewKeys`
dhBin	input	A DER encoded DH parameter stream
dhBinLen	input	The byte length of the `dhBin` parameter

Return Value	**Description**
PS_SUCCESS	Success. A valid key pointer will be returned in the keys parameter for use in a subsequent call to `matrixSslNewServerSession`
PS_MEM_FAIL	Failure. Unable to allocate memory for the structure
PS_PARSE_FAIL	Failure. Unable to parse DH parameter file

This version supports platforms that do not have file system support. This is the functional equivalent of the `matrixLoadDhParams` documented above.