MatrixSSL Getting Started
Version 4.1.0

TABLE OF CONTENTS

1 OVERVIEW

This Getting Started Guide explains how to quickly compile and test the MatrixSSL package on supported reference platforms. This guide also contains instructions on building and running the client and server applications provided in the package.

1.1 Who is this Document For?

- Software developers working on a supported platform that want to create a development environment for integrating MatrixSSL security into a custom application
- Software developers who want to port MatrixSSL to a new platform
- Anyone wanting to learn more about MatrixSSL

inside
secure
DRIVING TRUST

2 COMPILING AND TESTING MATRIXSSL

2.1 POSIX Platforms and Testing MatrixSSL

The POSIX classification in MatrixSSL encompasses support for several operating system platforms including Mac OSX 10.10 and most UNIX/LINUX varieties. This is the default platform for the Makefile system that is provided in the package and should be the first build option if you are unsure of your platform configuration.

2.1.1 Preparation

The development platform must have the following tools installed:

- The *tar* utility for expanding the package (or other decompression utility supporting *.tgz* files)

- A C source code compiler and linker (*GCC* is the default in the provided Makefile system)

- The *make* tool

2.1.2 Building the Source

1. From the command prompt, unpack the zipped tar image.

   ```
   $ tar -xzvf matrixssl-3-9-1-open.tgz
   ```

2. Change directory to the root of the package and build the MatrixSSL library

   ```
   $ cd matrixssl-3-9-1-open
   $ make
   ```

3. Confirm there were no compile errors and that the MatrixSSL libraries have been built. A successful build will result in 3 binaries. There should be a shared library for each of the three modules of ./core/libcore_s.a, ./crypto/libcrypt_s.a, and ./matrixssl/libssl_s.a. Applications link and interface with these libraries through the MatrixSSL public API set, which is documented in the MatrixSSL API reference, included in the distribution.

2.1.3 SSL Self-Test Application

Source code for a self-test application to exercise the SSL handshake and data exchange functionality of the MatrixSSL library is provided with the package. The following optional steps will enable the developer to build and run the test application to confirm the SSL protocol is fully functional.

1. Having successfully built the static library from the Building the source steps above, change directories to the test folder where the sslTest.c source is located and compile the application

   ```
   $ cd matrixssl/test
   ```

   ```
   $ make
   ```

4

inside
secure
DRIVING TRUST

2. Run the `sslTest` application from the command line. This sample output shows a successful run of the test using the default configuration of the open source package.

```
Testing TLS_RSA_WITH_AES_256_CBC_SHA suite
      Standard handshake test
            PASSED: Standard handshake
      Re-handshake test (client-initiated)
            PASSED: Re-handshake
      Resumed handshake test (new connection)
            PASSED: Resumed handshake
      Re-handshake test (server initiated)
            PASSED: Re-handshake
      Resumed Re-handshake test (client initiated)
            PASSED: Resumed Re-handshake
      Resumed Re-handshake test (server initiated)
            PASSED: Resumed Re-handshake
      Change cert callback Re-handshake test
            PASSED: Upgrade cert callback Re-handshake
      Change keys Re-handshake test
            PASSED: Upgrade keys Re-handshake
      Change cipher suite Re-handshake test
            PASSED: Change cipher suite Re-handshake
Testing TLS_RSA_WITH_AES_128_CBC_SHA suite
      Standard handshake test
            PASSED: Standard handshake
      Re-handshake test (client-initiated)
            PASSED: Re-handshake
      Resumed handshake test (new connection)
            PASSED: Resumed handshake
      Re-handshake test (server initiated)
            PASSED: Re-handshake
      Resumed Re-handshake test (client initiated)
            PASSED: Resumed Re-handshake
      Resumed Re-handshake test (server initiated)
            PASSED: Resumed Re-handshake
      Change cert callback Re-handshake test
            PASSED: Upgrade cert callback Re-handshake
      Change keys Re-handshake test
            PASSED: Upgrade keys Re-handshake
      Change cipher suite Re-handshake test
            PASSED: Change cipher suite Re-handshake
```

2.1.4 Sockets-Based Client and Server Applications

Source code for TCP/IP sockets-based client and server applications are provided with the MatrixSSL package. The following optional steps will enable the developer to build and run the applications to confirm the development platform is configured for MatrixSSL integration.

1. Having successfully built the library from the **Building the Source** steps above, change directories to the *apps/ssl* folder where the *client.c* and *server.c* source files are located and compile the applications.

```
$ cd apps/ssl
$ make
```

5

inside
secure
DRIVING TRUST

2. Run the *server* application from the command line

```
$ ./server Listening on port 4433
```

3. In a second shell environment, run the client application using the Bash shell script and verify two connections were made to the running server. Client trace in the successful case:

```
$ ./runClient.sh
client https://127.0.0.1:4433/ new:1 resumed:1 cipher count:1
version:3.3
=== 1 new connections ===
Validated cert for: Sample Matrix RSA-1024 Certificate.
SEND: [GET / HTTP/1.0
User-Agent: MatrixSSL/3.9.0-COMM
Accept: */*
Content-Length: 0
]
```

```
RECV PARSED: [HTTP/1.0 200 OK]
RECV PARSED: [Server: MatrixSSL/3.9.0-COMM]
RECV PARSED: [Pragma: no-cache]
RECV PARSED: [Cache-Control: no-cache]
RECV PARSED: [Content-type: text/plain]
RECV PARSED: [Content-length: 9]
RECV COMPLETE HTTP MESSAGE
N
146 bytes received
0 msec (0 avg msec/conn SSL handshake overhead)
0 msec (0 avg msec/conn SSL data overhead)
=== 1 resumed connections ===
SEND: [GET / HTTP/1.0
User-Agent: MatrixSSL/3.9.0-COMM
Accept: */*
Content-Length: 0
]
RECV PARSED: [HTTP/1.0 200 OK]
RECV PARSED: [Server: MatrixSSL/3.9.0-COMM]
RECV PARSED: [Pragma: no-cache]
RECV PARSED: [Cache-Control: no-cache]
RECV PARSED: [Content-type: text/plain]
RECV PARSED: [Content-length: 9]
RECV COMPLETE HTTP MESSAGE
R
146 bytes received
0 msec (0 avg msec/conn SSL handshake overhead)
0 msec (0 avg msec/conn SSL data overhead) RECV PARSED:
[HTTP/1.0 200 OK]
RECV PARSED: [Server: MatrixSSL/3.9.0-COMM]
RECV PARSED: [Pragma: no-cache]
RECV PARSED: [Cache-Control: no-cache]
RECV PARSED: [Content-type: text/plain]
RECV PARSED: [Content-length: 9]
RECV COMPLETE HTTP MESSAGE
N
146 bytes received
0 msec (0 avg msec/conn SSL handshake overhead)
0 msec (0 avg msec/conn SSL data overhead)
=== 1 resumed connections ===
SEND: [GET / HTTP/1.0
User-Agent: MatrixSSL/3.9.0-COMM
```

inside
secure
DRIVING TRUST

```
     Accept: */*
     Content-Length: 0
     ]
     RECV PARSED: [HTTP/1.0 200 OK]
     RECV PARSED: [Server: MatrixSSL/3.9.0-COMM]
     RECV PARSED: [Pragma: no-cache]
     RECV PARSED: [Cache-Control: no-cache]
     RECV PARSED: [Content-type: text/plain]
     RECV PARSED: [Content-length: 9]
     RECV COMPLETE HTTP MESSAGE
     R
     146 bytes received
     0 msec (0 avg msec/conn SSL handshake overhead)
```
0 msec (0 avg msec/conn SSL data overhead)

2.1.5 Debug Builds vs. Release Builds

The default compiler options in the *Makefile* build system use the *–O3* optimization flag for desktop systems to create a release quality MatrixSSL library. If you wish to create a debug version of the library (and applications) edit the common.mk file at the top of the package directory and change the `BUILD` macro to `debug` rather than the `release` default. To optimize for size, change the flag to –Os

inside
secure
DRIVING TRUST