http://www.yassl.com
mailto:info@yassl.com
phone: +1 206 369 4800

Building CyaSSL

CyaSSL was written with portability in mind, and should generally be easy to build on most systems. If you have difficulty building CyaSSL, do not hesitate to seek support through our forums on SourceForge.

*nix

When building CyaSSL on Linux, *BSD, OS X, Solaris, or other *nix like systems, use the autconf system. To build CyaSSL you only need to run two commands:

```
./configure
make
```

To install CyaSSL run:

```
make install
```

You may need superuser privileges to install, in which case proceed the command with sudo:

```
sudo make install
```

To test the build run the testsuite program from the testsuite directory.

./configure Options

--enable-debug	Enable CyaSSL debugging support, disabled by default
--enable-small	Enable the smallest possible build, disabled by default
--enable-singleThreaded	Enable single threaded mode, no multi thread protections
--enable-dtls	Enable CyaSSL DTLS support, this is disabled by default
--enable-opensslExtra	Enable extra OpenSSL API compatibility, increases the size
--enable-ipv6	Enable testing of IPV6, CyaSSL proper is IP neutral
--enable-fastmath	Enable fast math for BigInts, default is disabled
--enable-fasthugemath	Enable fast math + huge code for BigInt, default is disabled
--enable-bigcache	Enable a big session cache, default is disabled
--enable-hugecace	Enabe a huge session cache, default is disabled
--disable-shared	Disable the building of a shared CyaSSL library
--disable-static	Disable the building of a static CyaSSL library
--with-libz	Optionally include libz for compression

./configure Notes

Debug, enabling debug support allows easier debugging by compiling with debug information and defining the constant **DEBUG_CYASSL** which outputs messages to **stderr**. To turn debug logging on at runtime call *CyaSSL_Debugging_ON()* . To turn debug logging off at runtime call *CyaSSL_Debugging_OFF()*.

Small, enabling the small build option will create the smallest possible CyaSSL library. This will also remove features that may be desired like TLS, HMAC, SHA-256, error strings, or others. Only use this if the default build is too big and you don't mind losing features.

Single Threaded, enabling single threaded mode turns off multi thread protection of the session cache. Only enable single threaded mode if you know your application is single threaded or your application is multi threaded and only one thread at a time will be accessing the library.

DTLS, enabling DTLS support allows users of the library to also run DTLS in addition to TLS and SSL. DTLS support is still experimental so please send us any comments/ questions/suggestions.

OpenSSL Extra, enabling openssl extra includes a larger set of OpenSSL compatibility functions. The basic build will enable enough functions for most TLS/SSL needs. But if you're porting an application that uses 10s or 100s of OpenSSL calls then enabling this will allow better support. Our OpenSSL compatibility layer is under active development, so if there is a function missing that you need, then please contact us and we'll try to help.

IPV6, enabling IPV6 changes the test applications to use IPV6 instead of IPV4. CyaSSL proper is IP neutral, either version can be used, but currently the test applications are IP dependent, IPV4 by default.

fastmath, enabling fastmath will speed up public key operations like RSA, DH, and DSA. This switches the big integer library to a faster one that uses assembly if possible. Assembly inclusion is dependent on compiler and processor combinations. Some combinations will need additional configure flags and some may not be possible. Help with optimizing fastmath with new assembly routines is available on a consulting basis.

On ia32, for example, all of the registers need to be available so high optimization and omitting the frame pointer needs to be taken care of. CyaSSL will add `"-O3 -fomit-frame-pointer"` to **GCC** for non debug builds. If you're using a different compiler you may need to add these manually to **CFLAGS** during configure.

OS X will also need `"-mdynamic-no-pic"` added to CFLAGS. In addition, if you're building in shared mode for ia32 on OS X you'll need to pass options to LDLAGS as well:

```
LDFLAGS="-Wl,-read_only_relocs,warning"
```

This gives warning for some symbols instead of errors.

fastmath also changes the way dynamic and stack memory is used. The normal math library uses dynamic memory for big integers. fastmath uses fixed size buffers that hold 4096 bit integers by default, allowing for 2048 bit by 2048 bit multiplications. If you need 4096 bit by 4096 bit multiplications then change **FP_MAX_SIZE** in tfm.h. A couple of functions in the library use several temporary big integers meaning the stack can get relatively large. This should only some into play on embedded systems or in threaded environments where the stack size is set to a low value. If stack corruption occurs with fastmath during public key operations in those environments increase the stack size to accommodate the stack usage.

fasthugemath, enabling fasthugemath includes support for the fastmath library and greatly increases the code size by unrolling loops for popular key sizes during public key operations. Try using the benchmark utility before and after using fasthughmath to see if the slight speedup is worth the increased code size.

bigcache, enabling the big session cache will increase the session cache from 33 sessions to 1055 sessions. The default session cache size of 33 is adequate for TLS clients and embedded servers. The big session cache is suitable for servers that aren't under heavy load, basically allowing 200 new sessions per minute or so.

hugecache, enabling the huge session cache will increase the session cache size to 65,791 sessions. This option is for servers that are under heavy load, over 13,000 new sessions per minute are possible or over 200 new sessions per second.

disable shared, disabling the shared library build will exclude a CyaSSL shared library from being built. By default both a shared and static library are built. During testing, integration, or on limited systems you can save time and space by disabling either library from the build process.

disable static, disabling the static library build will exclude a CyaSSL static library from being built.

libz, enabling libz will allow compression support in CyaSSL from the libz library. Think twice about including this option and using it by calling *CyaSSL_set_compression()* . While compressing data before sending decreases the actual size of the messages

being sent and received, the amount of data saved by compression usually takes longer in time to analyze then it does to send it raw on all but the slowest of networks.

Windows

MSVC 6. Because some developers still use MSVC6 CyaSSL includes project files and a workspace for it. Though it is now deprecated and no longer supported.

VS 2005 / VS 2008. Solutions are included for Visual Studio 2005/2008 in the root directory of the install.

To test each build choose Build All and then run the testsuite program.

Building in a non-standard environment

While not officially supported, we try to help people wishing to build CyaSSL in a non standard environment, particularly with embedded and cross-compilation systems. Below are some notes on getting started with this.

1) Place all of the .c files from src/ and ctaocrypt/src into the same directory.

2) Place all of the .h files from include/ and ctaocrypt/include into the same directory as above.

3) Create an openssl directory below the directory containing the files above and place all the .h files from include/openssl into the openssl directory.

4) Even though all of the CyaSSL headers are now in the same directory as the source files some build systems will still want to explicitly know where the header files are so you may need to specify that.

5) CyaSSL defaults to a little endian system unless the configure process detects big endian. Since you aren't using the configure process you'll need to define **BIG_ENDIAN_ORDER** if you are using a big endian system.

6) CyaSSL benefits speed wise from having a 64 bit type available. The configure process determines if `long` or `long long` is 64 bits and if so sets up a define. So if `sizeof(long)` is 8 bytes on your system define **SIZEOF_LONG 8**. If it isn't but `sizeof(long long)` is 8 bytes then define **SIZEOF_LONG_LONG 8**.

7) Try and build the library, and let us know if you run into any problems. If you need help, then contact us at *info@yassl.com*.

8) Some defines that can modify the build

CYASSL_CALLBACKS is an extension that allows debugging callbacks through the use of signals in an environment without a debugger, it is off by default. It can also be used to set up a timer with blocking sockets. Please see the document "CyaSSL Extensions Reference" for more information.

SINGLE_THREADED is a switch that turns off the use of mutexes. CyaSSL currently only uses one for the session cache, if your use of CyaSSL is always single threaded you can turn this on.

HAVE_LIBZ is an extension that can allow for compression of data over the connection. It is off by default and normally shouldn't be used, see the note above under congiure notes libz.

NO_* removes parts from the build, you can also remove the respective source file as well from the build but not the header file.

NO_RC4 removes the use of the ARC4 steam cipher form the build. ARC4 is built in by default because it is still popular and widely used.

NO_DES removes the use of DES3 encryptions. DES3 is built in by default because some older servers still use it and it's required by SSL 3.0.

NO_DH and **NO_AES** are the same as the two above, they are widely used.

NO_RABBIT and **NO_HC128** remove stream cipher extensions from the build.

NO_MD4 removes MD4 from the build, MD4 is broken and shouldn't be used.

NO_DSA removes DSA since it's being phased out of popular use.

NO_PSK turns off the use of the pre shared key extension. It is built in by default.

OPENSSL_EXTRA builds even more OpenSSL compatibility into the library. It is off by default.

NO_CYASSL_CLIENT removes calls specific to the client and is for a server only build. You should only use this if you want to remove a few calls for the sake of size.

NO_CYASSL_SERVER likewise removes calls specific to the server side.

NO_FILESYSTEM is used if stdio isn't available to load certificates and key files. This enables the use of buffer extensions to be used instead of the file ones.

NO_TLS turns off TLS which isn't recommended.

NO_INLINE disables the automatic inlining of small heavily used functions. Turning this on will slow down CyaSSL and actually make it bigger since these are small functions, usually much smaller then function call setup/return.

NO_MAIN_DRIVER is used in the normal build environment to determine whether a test application is called on its own or through the testsuite driver application. You'll only need to use it with the test files; test.c, client.c, server.c, echoclient.c, echoserver.c, and testsuite.c

DEBUG_CYASSL builds in the ability to debug to steer. It is off by default.

TEST_IPV6 turns on testing of IPV6 in the test applications. CyaSSL proper is IP neutral, but the testing applications use IPV4 by default.

CYASSL_DTLS turns on the use of DTLS or datagram TLS, this isn't widely supported or used so it is off by default.