

# SharkSSL



## Embedded SSL/TLS Server

SharkSSL is an extremely compact SSL/TLS stack designed from the ground up to enable completely secure communication and management of remote embedded devices and dedicated applications. A remote server can authenticate itself to an SSL enabled client -- that is, Internet Explorer, Firefox, etc..

## Why Use SSL in Embedded Devices

The ubiquitous internet is becoming the norm for connecting and controlling devices, but the internet is inherently insecure. It is, therefore, necessary to encrypt data that is interchanged with devices to prevent unauthorized users from intercepting and potentially gaining access to your devices.

## Why Use SharkSSL

Before developing our own Secure Sockets Layer, we searched for a small, SSL implementation. This proved very difficult to find. The companies providing such solutions were either providing too expensive solutions or solutions too big for practical use in embedded devices. For example, with the standard OpenSSL library over 1 MB in size, it is not suitable for embedded systems. We designed SharkSSL to be very small. Thus far, it is the smallest SSL/TLS Server on the market.

## Comparing SharkSSL to other SSL Stacks

SharkSSL is tested with the [Barracuda Embedded Web Server](#), which is much more than a web-server. Barracuda has helped us test SharkSSL with many types of non-browser clients such as C++ HTTP libraries, Python, Java, and [WebDAV](#). The WebDAV protocol is an excellent robustness test as it puts much more strain on the SSL stack than a typical web-browser.

Testing a SSL server with only browsers is a poor test as typically little data is sent from the client to the server. Make sure you ask our competitors for a list of non-browser client tests they have performed including WebDAV. You should also ask for a list of all the browsers they have tested the SSL stack with.



# SharkSSL



## Embedded SSL/TLS Server

### Features

- Object-oriented library in ANSI C (with C++ wrapper code)
- Supports all Freescale™ ColdFire™ hardware-acceleration encryption engines (including the upcoming DragonFire™)
- Code size is less than 24KB total footprint on ColdFire(R) 54xx supporting the onboard hardware-acceleration encryption engine.
- Includes crypto software library for processors without hardware encryption support or with partial hardware encryption acceleration (AES, DES, 3DES, ARC4, SHA1, MD5)
- Includes proprietary RSA crypto library that can be retargeted to dedicated DSP engines
- Configurable session caching
- Advanced embedded buffer management with no coding required to handle the SSL buffers. Custom memory allocators can be specified.
- Transport agnostic, works with any transport type, including TCP/IP.
- Multithreading support for optimal performance when used with mutlti tasking/process OS's.
- Easily portable to any RTOS and any hardware-acceleration encryption engine. Off the shelf support for: OSE™, MQX™, SMX™, ThreadX™, SuperTask™, EBSnet™, rtplatform, uCLinux™, Linux and Windows™
- The Barracuda™ [Embedded Web Server](#) can take advantage of the optimized memory management supported by SharkSSL for persistent HTTP 1.1 connections

### SSLv3 & TLS 1.0/1.1\* cipher supported (in decreasing order of "strength"):

- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_RSA\_WITH\_RC4\_128\_SHA
- TLS\_RSA\_WITH\_RC4\_128\_MD5
- TLS\_RSA\_WITH\_DES\_CBC\_SHA
- TLS\_RSA\_WITH\_NULL\_SHA
- TLS\_RSA\_WITH\_NULL\_MD5

