wolfSSL Products

www.wolfSSL.com

Table of contents

Table of contents

wolfSSL Embedded SSL/ TLS Library

wolfSSL Version 5.7.6
Release Date: 2024-12-31

Description

The wolfSSL library is a lightweight SSL/TLS library written in ANSI C and targeted for embedded, RTOS, and resource-constrained environments - primarily because of its small size, speed, and feature set. It is commonly used in standard operating environments as well because of its royalty-free pricing and excellent cross platform support. wolfSSL supports industry standards up to the current **TLS 1.3** and **DTLS 1.3** levels, is up to *20 times smaller* than OpenSSL, and offers progressive ciphers such as ChaCha20, Curve25519, Poly1305, ED25519, and SHA-3. User benchmarking and feedback reports dramatically better performance when using wolfSSL over OpenSSL.

wolfSSL is powered by the wolfCrypt library. wolfCrypt is **FIPS 140-3 Level 1 validated**, with certificates **#4718**.

For additional information, visit our FIPS FAQ page or contact fips@wolfssl.com.

wolfSSL is built for maximum portability, and is generally very easy to compile on new platforms. If your desired platform is not listed under the supported operating environments, please contact wolfSSL.

wolfSSL supports the C programming language as a primary interface. It also supports several other host languages, including Java, C#, Ada, and Python, as well as PHP and Perl (through a SWIG interface). If you have interest in using wolfSSL in another programming language that it does not currently support, please contact wolfSSL at **facts@wolfssl.com**.

Features

- **TLS 1.3 support** (client and server)
- DTLS 1.1, 1.2 and 1.3 support (client and server)
- Legacy support for TLS 1.0, SSL 3.0, DTLS 1.0 (disabled by default) Minimum footprint size of **20-100 KB**
- Runtime memory usage between **1-36 KB**
- FIPS Ready build for easily starting FIPS compliance preparations
- OpenSSL compatibility layer
- Simple API
- OCSP, OCSP Stapling, and CRL support
- Multiple Hashing Functions:
  - MD5, SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512), SHA-3, BLAKE2b, Poly1305
- Block, Stream, and Authenticated Ciphers:
  - AES (CBC, CTR, GCM, CCM, XTS, OFB, CFB, GMAC, CMAC), Camellia, 3DES, ChaCha20
- Public Key Options:
  - RSA, DH, DHE, ECDH-ECDSA, ECDHE-ECDSA, ECDH-RSA, ECDHE-RSA, Ed448, Ed25519, Curve448, Curve25519
- Post Quantum Algorithms (CNSA 2.0 compliant):
  - ML-KEM (Kyber), ML-DSA (Dilithium), LMS, XMSS
- Password-based Key Derivation:
  - HMAC, PBKDF2
- PCKS #1, 3, 5, 7–12 support
- Linux kernel module support
- ECC and RSA Key Generation
- X.509v3 RSA and ECC Signed Certificate Generation
- Mutual authentication support (client/server)
- PSK (Pre-Shared Keys) and PSK-DHE
- Persistent session and certificate cache
- PEM and DER certificate support
- Standalone Certificate Manager
- Hardware Crypto Support
  - Intel AES-NI, AVX1/2, RDRAND, RDSEED, SGX, Cavium NITROX, Intel QuickAssist, STM32, NXP (CAU, mmCAU, SEC, LTC, CAAM), Microchip PIC32MZ, ARMv8, Renesas (TSIP, FSP)
- SSL Sniffer (SSL Inspection) Support
- Portable Abstraction Layers/User Callbacks
- Open-Source Project Integrations
  - MySQL, OpenSSH, Apache, nginx, Open vSwitch, and more
- Embedded IDPS integration (wolfSentry)

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, Yocto Linux, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, NonStop, TRON/ITRON/µITRON, Cesium, Micrium µC/OS-III, FreeRTOS, SafeRTOS, NXP/Freescale MQX, Nucleus,TinyOS, HP/UX, AIX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PetaLinux, Apache Mynewt, PikeOS, Deos, Azure Sphere OS

Supported Chipmakers

wolfSSL has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip/Atmel, STMicro, Analog Devices, Texas Instruments, Xilinx SoCs/FPGAs, Renesas, Espressif, and more.

If you would like to use or test wolfSSL on another chipset or OS, let us know and we'll be happy to support you.

wolfssl.com
github.com/wolfssl

wolfCrypt Embedded Crypto Engine

Description

The wolfCrypt cryptography engine is a lightweight crypto library written in ANSI C and targeted for embedded and RTOS environments - primarily because of its small size, speed, and feature set. It is commonly used in standard operating environments as well because of its royalty-free pricing and excellent cross platform support.

wolfCrypt supports the most popular algorithms and ciphers as well as progressive ones such as ChaCha20, Curve25519, Poly1305, ED25519, and SHA-3. wolfCrypt is stable, production-ready,and backed by our excellent team of security experts. It is used in millions of applications and devices worldwide.

Highlights

• ECC, up to 521 bit
• Hash-based PRNG
• Progressive list of supported ciphers
• Post Quantum Cryptography support
• Lightweight - small footprint size, low runtime memory
• Portable - simple and clean API
• Modular design - Individual algorithms and ciphers are easily broken out of the wolfCrypt package to be used independently

wolfCrypt is built for maximum portability and is generally very easy to compile on new platforms. It supports the C programming language as a primary interface.

For more information, please contact wolfSSL at **facts@wolfssl.com**.

Features

- Multiple Hash Functions:
  - MD5, SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512), SHA3, Poly1305
- Block, Stream, and Authenticated Ciphers:
  - AES (CBC, CTR, GCM, CCM, XTS, OFB,CFB, GMAC, CMAC), DES, 3DES, ChaCha20
- Public Key Algorithms:
  - RSA, DH,DHE, ECDH-ECDSA, ECDHE-ECDSA, ECDH-RSA, ECDHE-RSA,
- Post Quantum Algorithms (CNSA 2.0 compliant):
  - ML-KEM (Kyber), ML-DSA (Dilithium), LMS, XMSS
- Password-based Key Derivation: HMAC, PBKDF2
- Curve25519 and Ed25519
- PEM and DER certificate support
- X.509 Encoding / Decoding
- RSA and ECC Key Generation
- x509 v3 Signed Certificate Generation
- PKCS support:
  - PKCS#1 (RSA Cryptography Standard)
  - PKCS#5 (Password-Based Encryption Standard)
  - PKCS#7 (Cryptographic Message Syntax - CMS)
  - PKCS#8 (Private-Key Information Syntax)
  - PKCS#10 (Certificate Signing Request - CSR)
  - PKCS#12 (Personal Information Exchange Syntax Standard)
- Assembly Optimizations
- Custom Memory Hooks
- Easily ties in to Hardware-based RNG solutions
- Hardware Cryptography Support: Intel AES-NI, AVX1/2, RDRAND, RDSEED, SGX, Cavium NITROX, Intel QuickAssist, STM32F2/F4, Freescale/NXP (CAU, mmCAU, SEC, LTC), Microchip PIC32MZ, ARMv8, and more
- OpenSSL compatibility layer

Supported Chipmakers

wolfCrypt has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip/Atmel, STMicroelectronics, Analog Devices, Texas Instruments, Xilinx SoCs/FPGAs, Renesas, Espressif and more.
If you would like to use or test wolfSSL on another chipset or environment, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, Yocto Linux, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, NonStop, TRON/ITRON/µITRON, Cesium, Micrium µC/OS-III, FreeRTOS, SafeRTOS, NXP/Freescale MQX, Nucleus, TinyOS, HP/UX, AIX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PetaLinux, Apache Mynewt, PikeOS, Deos, Azure Sphere OS

wolfssl.com
github.com/wolfssl

All the News on **wolfSSL FIPS**

FIPS 140-3 Validation, Certificate #4718

We are the first to release under the new FIPS 140-3 validation procedures!

FIPS Ready Now Available!

wolfSSL offers a wolfCrypt FIPS Ready version of the library, providing FIPS-enabled cryptography within the wolfSSL source tree. FIPS Ready does not grant a FIPS certificate, however it ensures compliance with FIPS best practices, includes a default entry point, power-on self-tests, conditional algorithm self-tests and is “ready” to be validated the moment a hard FIPS need arises. You can download wolfCrypt FIPS Ready from https://www.wolfssl.com/download/ and access setup details in the https://www.wolfssl.com/docs/fips-ready-user-guide/. For already validated modules just reach out to fips@wolfssl.com. FIPS Ready a budget friendly alternative to full FIPS validated modules, it is intended for prototyping or first-time FIPS users where a hard FIPS need is not yet realized but may become necessary in the future.

New certificate #4718, Three Big Differences

- Includes the latest and greatest **Algorithms**, so potential FIPS users can stay at the cutting edge of industry with confidence all algorithms can be validated if and when needed!
- **Validated Entropy Source:** Getting your entropy source properly validated is difficult and time consuming. We’ve done the work and make it easy for our consumers.
- **Extensible Hardware Encryption:** wolfSSL certificate(s) support hardware encryption in an extensible way. Currently supported AES-NI and ARMv7/8. The cert can be extended to additional hardware encryption methods when demand arises.

wolfEntropy (MEMUSE)

wolfEntropy is our new, drop-in,validated entropy source for use with any cryptographic engine on Intel or ARM and can be extended to additional silicone as needed.

OpenSSL Engine - wolfEngine

wolfCrypt FIPS can now operate as an OpenSSL engine for **drop in** FIPS replacement in your OpenSSL applications.

OpenSSL 3.0 Provider Solution with FIPS - wolfProvider

wolfSSL has developed an OpenSSL 3.0 provider, allowing you to use the latest version of OpenSSL backed by our FIPS-certified wolfCrypt library. Like wolfEngine, the wolfSSL provider for OpenSSL is an excellent pathway for users looking to get FIPS compliance fast while still using OpenSSL.

Learn More

For more information on wolfSSL TLS 1.3 features or to evaluate it, please contact us at facts@wolfssl.com. Please send any comments or feedback on wolfSSL TLS 1.3 support to support@wolfssl.com. Thanks!

wolfssl.com
github.com/wolfssl

wolfTPM Portable TPM 2.0 Library

wolfTPM Version: 3.8.0
Release Date: 2025-01-07

Overview

Trusted Platform Module (TPM) is a standardized secure processor that lives as a dedicated chip alongside the main processor (MCU, CPU or SoC). TPM can be found in almost any modern computer systems. Computer programs can use a TPM to authenticate hardware devices, since each TPM chip has a unique and secret RSA key burned in as it is produced. Applications can protect their keys and secrets using the TPM as a vault.

wolfTPM is a portable, open-source TPM 2.0 stack with backward API compatibility, designed for embedded use. It is highly portable, due to having been written in native C, having a single IO callback for hardware interface, no external dependencies, and its compact code with low resource usage. The wolfTPM library includes a stable wrapper API interface to simplify common use cases and includes many ready-to-use examples.

Description

A Trusted Platform Module (TPM) is a cryptographic module that includes key generation and storage capabilities. The wolfTPM project is designed for embedded use with the TPM 2.0 specification.

Highlights:

- Securely generate, store, and use RSA or ECC keys
- Secure Non-Volatile (NV) data storage and counters key hierarchy
- Three branches with different privileges to better separate key purposes
- Hardware random number generator (TRNG)
- Dictionary attack prevention mechanisms
    - Hardware tracks and enforces attacks
- Quick key loading
    - Can load previously created TPM private key material encrypted with a key only the TPM knows

For more information, please contact wolfSSL at facts@wolfssl.com.

Supported Chipmakers

wolfTPM supports all TPM 2.0 modules including:
Infineon SLB9670/SLB9672/SLB9673, STMicro ST33KTPM2X, Microchip ATTPM20, Nations Tech Z32H330TC/NS350 and Nuvoton NPCT650/NPCT750

Supported Operating Environments

- Platform support for Raspberry Pi, STM32 with CubeMX, Microchip/Atmel, Infineon TriCore, Xilinx and Barebox
- Support for Microsoft Windows, Linux and Bare-Metal
- Support for fTPM and Simulators

If you would like to use or test wolfTPM/wolfSSL on another chipset or environment, let us know and we'll be happy to support you.

Features

- Provides all TPM 2.0 API's in compliance with the spec
- Built-in TPM TIS (Transport Interface Specification) layer for direct SPI or I2C communication
- Supports parameter encryption between the Host and the TPM to protect from man-in-the-middle attacks
    - Authenticated sessions with HMAC
        - Bound to Key and/or Salt
    - AES-CFB or XOR
- Backward API compatibility
- Bare-Metal or RTOS application or firmware
- Portability to different platforms:
    - Native C code designed for embedded use
    - Single IO callback for hardware SPI interface
    - No external dependencies
    - Compact code size and minimal memory use
- Includes stable wrapper API for:
    - Key Gen/Import/Export, RSA encrypt/decrypt, ECC sign/verify, ECDH, NV, Hashing/HMAC, AES, Policy, Secure Boot, Firmware Update
- Includes example code for:
    - Attestation (activate/make credential, quote)
    - TLS Client and Server
    - Endorsement Key/Cert retrieval and validation
    - Sealing/Unsealing Secrets
    - Use of the TPM's Non-volatile memory
    - Policies
    - Benchmarking TPM algorithms
    - PKCS#7
    - Certificate Signing Request (CSR)
    - Generation of signed timestamp
- Secure Boot integration with wolfBoot and U-Boot
- PKCS11 support using wolfPKCS11
    - Exposes TPM as PKCS11 interface

Product page: https://wolfssl.com/products/wolftpm
GitHub location: https://github.com/wolfSSL/wolfTPM

wolfMQTT Client Library

wolfMQTT Version: 1.19.1
Release Date: 2024-11-06

Description

MQTT (Message Queuing Telemetry Transport) is a lightweight open messaging protocol that was developed for constrained environments such as M2M (Machine to Machine) and IoT (Internet of Things), where a small code footprint is required. MQTT is based on the Pub/Sub messaging principle of publishing messages and subscribing to topics. The protocol efficiently packs messages to keep the overhead very low. The MQTT specification recommends TLS as a transport option to secure the protocol using port 8883 (secure-mqtt). Constrained devices can benefit from using TLS session resumption to reduce the reconnection cost.

The wolfMQTT library is a client implementation of the MQTT written in C for embedded use. It supports SSL/TLS via the wolfSSL library. It was built from the ground up to be multi-platform, space conscience and extensible. It supports all Packet Types, all Quality of Service (QoS) levels 0-2 and supports SSL/TLS using the wolfSSL library. This implementation provides support for MQTT v5.0 and is based on MQTT v3.1.1. Additionally, there is also client support for MQTT-SN (Sensor Network).

wolfMQTT is built for maximum portability, and is generally very easy to compile on new platforms. If your desired platform is not listed under the supported operating environments, please contact wolfSSL at facts@wolfssl.com.

Features

- Built from scratch by wolfSSL engineers
- Supports MQTT specifications v3.1.1 and v5.0
- Support for MQTT-SN
- Supports all client side packet types and protocol options
- Lightweight (Compiled size is less than 10KB)
- QoS Levels 0-2 (guaranteed delivery)
- Message integrity, security are still available
- Supports plain TCP or TLS (via the wolfSSL library)
- Single and multithread support
- Written in native C89 with portability/compatibility in mind
- User manual with build instructions, example overview, and API documentation
- Example MQTT client implementations
- Network interface is abstracted via callbacks for extensibility
- Packet parsing encoding/decoding structured for custom use
- Minimal external dependencies (strlen, memcpy, memset)
- Detailed error checking/handling
- Doxygen style inline documentation
- Less than 1200 lines of well structured C code
- Tested on multiple variants of MQTT broker servers, QoS levels 0-2 with/without TLS
- Tested on Linux, Mac OS X, and Freescale Kinetis K64
- Inherits wolfSSL library features such as lightweight TLS, FIPS 140-3, small size and portability
- Open source (GPLv2) and commercial licensing
- FreeRTOS + TCP support
- Example Arduino IDE project
- Examples for AWS and Azure
- Example UART interface for wolfMQTT

GitHub location: https://github.com/wolfSSL/wolfMQTT

Supported Chipmakers

wolfSSL has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip (PIC32)/Atmel, STMicroelectronics (STM32F2/F4), Analog Devices, TI, and more.
If you would like to use or test wolfMQTT/wolfSSL on another chipset or environment, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, Yocto Linux, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, NonStop, TRON/ITRON/µITRON, Cesium, Micrium µC/OS-III, FreeRTOS, SafeRTOS, NXP/Freescale MQX, TinyOS, HP/UX, AIX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PetaLinux, Apache Mynewt, PikeOS

wolfssl.com
github.com/wolfssl

wolfSSH Lightweight SSH Library

Current Version: 1.4.19 Release
Date: 2024-11-01

Description

The wolfSSH library is a lightweight SSHv2 client and server library written in ANSI C and targeted for embedded, RTOS, resource-constrained, and IoT environments - primarily because of its small size, speed, and feature set. It is commonly used in standard operating environments as well because of its royalty-free pricing and excellent cross platform support.wolfSSL supports the industry standard **SSH v2**.

wolfSSH is powered by the wolfCrypt library. wolfCrypt is **FIPS 140-3 validated**, with certificate **#4718** and **FIPS 140-2 validated**, with certificate **#3389**. For additional information, visit our FIPS FAQ page or contact **fips@wolfssl.com**.

wolfSSH is built for maximum portability, and is generally very easy to compile on new platforms. If your desired platform is not listed under the supported operating environments, please contact wolfSSL Inc.

wolfSSH supports the C programming language as a primary interface. If you have interest in using wolfSSH in another programming language that it does not currently support, please contact wolfSSL Inc. at **facts@wolfssl.com**.

Features

- SSH v2.0 (client and server)
- Minimum footprint size of 33kB
- Runtime memory usage between 1.4 and 2kB, not including a configurable receive buffer
- Multiple Hashing Functions:
  - SHA-1, SHA-2 (SHA-256, SHA-384, SHA-512), BLAKE2b
- Block, Stream, and Authenticated Ciphers: AES (CBC, CTR, GCM)
- Public Key Options: RSA, DH, ECC
- Support for hybrid post quantum use with Kyber
- ECC Support (ECDSA with curves: NISTP256, NISTP384, NISTP521)
- Curve25519 and Ed25519
- Client authentication support (RSA key, password)
- SCP and SFTP support
- Port forwarding support
- PEM and DER certificate support
- Hardware Cryptography Support:
  - Intel AES-NI and AVX1/2, RDRAND, RDSEED, Cavium NITROX, STM32F2/F4 hardware crypto, Freescale CAU/ mmCAU / SEC, Microchip PIC32MZ, MPLAB Harmony on PIC32
- Echoserver functionality
- Includes a MS Visual Studio solution to simplify SSH usage on Windows
- Interop Tested Against:
  - OpenSSH, Tera term, PuTTY, Dropbear, Firezilla, BitVise
- **FIPS 140-2 & FIPS 140-3 validated cryptography library with wolfCrypt!**

Product page: https://wolfssl.com/products/wolfSSH

GitHub location: https://github.com/wolfSSL/wolfSSH

Supported Chipmakers

wolfSSH has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip (PIC32)/Atmel, STMicroelectronics (STM32F2/F4), Analog Devices, TI, and more.

If you would like to test wolfSSH on another environment, let us know and we'll be happy to support you.

Supported Operating Environments

TRON/ITRON/µITRON, Cesium, Micrium's µC/OS, FreeRTOS, SafeRTOS, FreescaleMQX, Nucleus, TinyOS, HP/ UX, ARCMQX, TI-RTOS, contiki, Riot-OS, ChibiOS, NuttX, CMSIS-OS, RTEMS, and more
If you would like to test wolfBoot on another environment, let us know and we'll be happy to support you..

wolfBoot Secure Bootloader

wolfSSL Version 5.7.6
Release Date: 2024-12-31
Current Version: 2.4.0
Release Date: 2025-01-07

Description

wolfBoot is a secure bootloader that leverages wolfSSL's underlying wolfCrypt module to provide signature authentication for the running firmware. wolfBoot is easily ported and integrated in existing embedded software projects. wolfBoot is designed to be a portable, OS -agnostic, secure bootloader solution for all embedded systems, relying on wolfCrypt for firmware authentication.

wolfBoot comes with an included key generation tool. This tool generates a key -pair up on building the wolfBoot library. The generated key-pair can then be used to sign the firmware that is being loaded on to the device, and to transform a bootable firmware image to comply with the firmware image format required by the bootloader.

Due to its minimalist design and the tiny Hardware Abstraction Layer (HAL) API, wolfBoot is completely independent of any OS or bare-metal application and can be easily ported and integrated into existing embedded software solutions.

Upon receiving and installing a verified update, wolfBoot keeps a backup copy of the newest firmware image that had been confirmed to work correctly. If the new version is not confirmed by the application itself, or whenever the image installed is damaged or corrupt, the bootloader will restore the state of the system before the most recent update.

wolfBoot is entirely written in C and ARM assembly language and **does not use any dynamic memory** allocation, making it usable in safety-critical environments. For more information, please contact wolfSSL Inc. at **facts@wolfssl.com**

Features

- Multi-slot partitioning of the flash device
- Integrity verification of the firmware image (s)
- Authenticity verification of the firmware image (s) using wolfCrypt 's public key cryptography algorithms:
  - RSA
  - ECC
  - ED25519, ED448
  - PQC (LMS, XMSS, ML-DSA)
- Highly reliable, transport-agnostic firmware update
- Minimalist Hardware Abstraction Layer (HAL) interface to facilitate portability cross different vendors/ MCUs
- In -place chain-loading of the firmware image in the primary slot
- Support for ARM TrustZone
- Support for bootloader updates (self-update)
- Support for incremental (delta) updates
- Support for external SPI flash memory
- HAL Support for 40+ different targets
- Protection against fault injection attacks
- Quantum-resistant authentication
- Hybrid mode (PQC + classic)
- Architectures supported:
  - ARM Cortex-M
  - ARM Cortex-A
  - ARM Cortex-R
  - Aurix TC3xx
  - 32-bit Risc-V
  - Intel x86_64
  - PowerPC (32 and 64 bits)
  - Renesas RXv3
- **FIPS 1 40-3** validated cryptography library with wolfCrypt!
- **DO-178C** validation up to **DAL-A**

Supported Chipmakers

wolfBoot support includes the RISC-V, ARM Cortex -M, ARM Cortex-R, PowerPC, Intel x86_64 and RXv3 boot mechanisms among others.
If you would like to use or test wolfBoot/ wolfSSL on another arcohitecture, let us know!

Supported Operating Environments

TRON/ITRON/µITRON, Cesium, Micrium's µC/OS, FreeRTOS, SafeRTOS, FreescaleMQX, Nucleus, TinyOS, HP/ UX, ARCMQX, TI-RTOS, contiki, Riot-OS, ChibiOS, NuttX, CMSIS-OS, RTEMS, Linux, Integrity OS and more
If you would like to test wolfBoot on another environment, let us know and we'll be happy to support you..

wolfssl.com
github.com/wolfssl

wolfProvider

Description

wolfProvider is an OpenSSL provider backed by wolfSSL's wolfCrypt cryptography library. wolfCrypt is FIPS-validated, so wolfProvider can be used to achieve FIPS compliance with OpenSSL, all without having to touch OpenSSL code itself. wolfProvider is structured as a standalone library which links against wolfSSL (libwolfssl) and OpenSSL.

wolfProvider implements and exposes an **OpenSSL provider implementation** which wraps the wolfCrypt native API internally. Algorithm support matches that as listed on the wolfCrypt is **FIPS 140-3 validated**, with certificate **#4718** and **FIPS 140-2 certificate #3389.**

Highlights

- Initialization Mode
  - Dynamic provider loading
  - Static entry point
  - OpenSSL configuration file
- OpenSSL test version
  - 3.0.0
  - 3.0.8

OpenSSL Version Support

Initialization modes can be used with any OpenSSL version that supports the provider framework. Older versions of OpenSSL use a similar concept called engines. wolfSSL also offers an engine backed by wolfCrypt. Please reach out to **facts@wolfssl.com** if you're interested in evaluating the wolfSSL engine.

Features

- Multiple Hash Functions:
  - MD5-1, SHA-1, SHA-2, (SHA-224, SHA-256, SHA-384, SHA-512), SHA3 (SHA3-224, SHA3-256, SHA3-384, SHA3-512)
- Block, Stream and Authenticated Cipher
  - AES (128, 192, 256,ECB, CBC, CTR, GCM, CCM), GMAC, CMAC
- Deterministic Random Bit Generator (DRBG)
- Public Key Algorithms
  - RSA, DH
  - ECC
  - ECDSA, ECDH, Curve (P-192, P-224, P-256, P-384, P-521)
- Key Derivation
  - HMAC, PBKDF2, HKDF
- RSA and ECC key generation
- TLS PRF

Supported Chipmakers

wolfProvider has support for chipsets including ARM, Intel, Motorola, NXP/Freescale, Microchip/Atmel, STMicroelectonics, Analog devices, Texas Instruments, Xilinx SoCs/FPGAs, Renesas, Espressif and more. If you would like to use or test wolfProvider on another chipset or environment, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, Yocto Linux, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, PetaLinux, Apache Mynewt, PikeOS, Azure Sphere OS

wolfssl.com
github.com/wolfssl

wolfSentry Embedded IDPS

Current Version: 1.6.2
Date: 2024-01-02

Description

wolfSentry is an embedded firewall and IDPS (intrusion detection and prevention system). At its core, it features an embedded, dynamic firewall engine, with fast and efficient lookups. wolfSentry is dynamically configurable, with test-commit se man tic s, and can easily associate user-defined events with user- defined actions, contextualized by both built-in and user-defined connection attributes, tracking the evolution of the network transaction profile. wolfSentry is fully integrated into the wolfSSL library, as well as wolfMQTT, and wolfSSH, with optional in-tree call-ins and callbacks that give application developers turnkey IDP S across all network-facing wolfSSL product s, with a viable zero-configuration option. These integrations will be available via simple --enable-wolfsentry configure options in wolfSSL sibling products.

The wolfSentry engine is dynamically configurable programmatically through an API, o r from textual inputs supplied to the engine. Callback and client-server implementations are also un de r development that will deliver advanced capabilities including remote logging through MQTT or syslog, and remote configuration and status queries, all cryptographically secured.

Notably, wolfSentry is designed from the ground up to function well in resource-constrained, bare-metal, and real time environments, with or without thread support, using deterministic algorithms that maximize availability and stay within rigidly designated maximum memory and scheduling footprints. Use cases include RTOS IDPS, and IDP S for ARM silicon and other common embedded CPUs and MCUs. wolfSentry with dynamic firewalling can add as little as 100k to the code footprint, and 32k to the volatile state footprint, and can fully leverage the existing logic and state of applications and sibling libraries.

If you have interest in using wolfSentry or any questions or comments, please contact wolfSSL at facts@wolfssl.com.

Features

- wolfSentry is designed to integrate directly with network-facing applications/libraries to block bad traffic, and it can optionally integrate with host firewall facilities, via plugins.
- It can run on bare metal, in which case the firewall functions can be directly integrated into the network stack of the application via patched-in call-ins, or call backs installed using host environment interfaces.
- Fully extensible
  - a dynamically configurable logic hub
  - user-defined rules link app-defined events with app-defined actions via plugins
    - plugins can be filters, decision logic, and/or orchestration logic
  - hub and plugins are mainly keyed on network attributes, and track current status
  - plugins can also track and use fully app-defined data for each network association
- Fully integrated into wolfSSL, wolfMQTT, and wolfSSH
  - zero-development IDPS across all network-facing wolfSSL products, using bundled COTS plugins
  - zero-configuration option
  - simple --enable-wolfsentry configure options in wolfSSL sibling products
- Dynamically configurable
  - programmatically through an API
  - textual human-readable configuration files, loadable/ reloadable at anytime
- Bundled plugins for remote logging, commands, and status queries, secured with TLS
  - MQTT
  - Syslog
  - SMTP
  - embedded web server with RESTful API

Supported Chipmakers

wolfSSL has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip/Atmel, STMicro, Analog Devices, Texas Instruments, Xilinx SoCs/FPGAs, Renesas, Espressif, and more.

If you would like to use or test wolfSSL on another chipset or OS, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, Net BSD, OpenBSD, e mbe dde d Linux, Yo c to Linu s, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone(iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, OpenCL, NonStop, TRON/ITRON/µITRON, Cesium, Micrium's µC/OS, FreeRTOS, SafeRTOS, Freescale MQX, Nucleus, TinyOS, HP/UX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, GreenHills INTEGRITY, Keil RTX, TOPPERS, PetaL in us, Apache Mynewt, PikeOS, Deos, Azure Sphere OS, FreeBSD

wolfssl.com
github.com/wolfssl

wolfEngine wolfCrypt FIPS engine for OpenSSL

wolfSSL Version 5.7.0
Release Date: 2024-03-20

Description

wolfEngine is an OpenSSL engine backed by wolfSSL's wolfCrypt cryptography library. wolfCrypt is FIPS-validated, so wolfEngine can be used to achieve FIPS compliance with OpenSSL, all without having to touch the OpenSSL code itself. wolfEngine is structured as a standalone library which links against wolfSSL (libwolfssl) and OpenSSL.

wolfEngine implements and exposes and **OpenSSL engine implementation** which wraps the wolfCrypt native API internally. Algorithm support matches that as listed on the wolfCrypt is **FIPS 140-3 validated**, with certificate **#4718** and FIPS 140-2 certificate #3389.

Highlights

- Initialization modes
  - Dynamic engine loading
  - Static entry point
  - OpenSSL configuration file
- OpenSSL tested versions
  - 1.0.2h
  - 1.1.1b

Applications that have been successfully compiled using wolfEngine include OpenSSL Unit Tests, NGINX, cURL, Stunnel, OpenSSH, and many more.

OpenSSL Version Support

Initialization modes can be used with any OpenSSL version that supports the engine framework. Engines are deprecated in OpenSSL 3.0.0. They're replaced with a similar concept called providers. wolfSSL also offers a provider backed by wolfCrypt. Please reach out to facts@wolfssl.com if you're interested in evaluating the wolfSSL provider.

Features

- Multiple Hash Functions:
  - SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512), SHA3 (SHA3-224, SHA3-256, SHA3-384, SHA3-512)
- Block, Stream, and Authenticated Ciphers:
  - AES (128, 192, 256, ECB, CBC, CTR, GCM, CCM), CMAC
- Deterministic Random Bit Generator (DRBG)
- Public Key Algorithms:
  - RSA, DH
- ECC:
  - ECDSA, ECDH, Curve (P-192, P-224, P-256, P-384, P-521)
- Key Derivation:
  - HMAC, PBKDF2, HKDF
- RSA and ECC Key Generation
- TLS PRF
- SHA-3 support available with OpenSSL versions 1.1.1+.
- EC_KEY_METHOD available with OpenSSL versions 1.1.1+.
- OpenSSL compatibility layer

Supported Chipmakers

wolfEngine has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip/Atmel, STMicroelectronics, Analog Devices, Texas Instruments, Xilinx SoCs/FPGAs, Renesas, Espressif and more.

If you would like to use or test wolfEngine on another chipset or environment, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, Yocto Linux, OpenEmbedded, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, NonStop, TRON/ITRON/µITRON, Cesium, Micrium µC/OS-III, FreeRTOS, SafeRTOS, NXP/Freescale MQX, Nucleus, TinyOS, HP/UX, AIX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PetaLinux, Apache Mynewt, PikeOS, Deos, Azure Sphere OS

wolfssl.com
github.com/wolfssl

wolfFE - File Encryption for support of Double Layer encryption (CSfC-listed and NIAP-certified)

Description

The File Encryption application adheres to the requirements of the software protection profile (SPP) and incorporates mod file encryption (MOD-FE) capabilities. wolfFE can be used as the second layer in a dual layer encryption requirement scenario. For example, if you have full drive encryption, and need to layer in an additional layer of software encryption, wolfFE is the perfect solution. The application utilizes the FIPS-validated wolfCrypt AES GCM encryption/ decryption algorithm to enhance file security with third-party hardware or software solutions. It provides two API interfaces:

Int **encrypt_file_AesGCM**(const char* in_file, const chr* out_file, const char* key_str, const char* ivv_str)
In **decrypt_file_AesGCM**(const char* in_file, const char* out_file, const char* key_str)

The application uses POSIX APIs, a 32-byte key size, and an IV length of 16 bytes. It is optimized to use a configurable buffer size to minimize file 1/0 overhead.

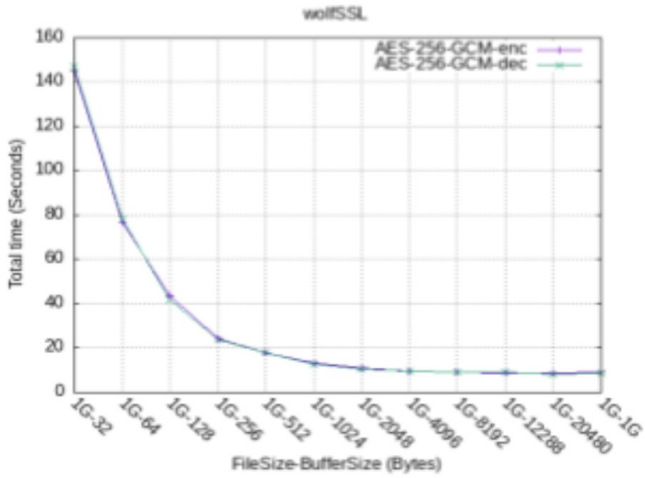
When using the encrypt_file_AesGCM() API, the given file will be encrypted and written to the specified cipher output file. The cipher output file will have a header at the beginning: WOLFSSL (7 Bytes) magic/ identifier | TAG (16 Bytes) | IV (16 Bytes) | cipher data (= size of the plain file) …

The decrypt_file_AesGCM() API extracts/ uses the header and decrypts the cipher data.

The wolfFE application is being evaluated for NIAP certifications, and once certified, it can be listed in the CSfC program. For additional information, visit our FAQ page ot contact facts@wolfssl.com.

When encrypting a file, data is read from the file and processed using read and write buffers before being written to another file. The size of these buffers is critical for the encryption speed and overall performance. It is important to consider factors such as memory fragmentation, MMU memory alignments, cache utilization, and minimizing OS overhead to optimize performance. The MAX_BUFFER_SIZE should be set to a value equal to or greater than the system's page size and smaller than the available system memory for optimal performance. However, in a bare metal system, you can utilize the maximum available memory for your buffers to achieve the best performance. The default buffer size values can be changed with CPPFLAGS="-DMAX_BUFFER_SIZE=8192-DMIN_BUFFER_SIZE=1024".

Here's a plot analyzing Buffer Size and Time Performance of



Supported Chipmakers

wolfFE supports chipsets such as Intel, ARM, and Mac.

If you would like to use or test wolfFE on another chipset, let us know and we'll be happy to support you.

Supported Operating Environments

Linux, VxWorks and can be extended to any other operating system.

If you would like to test wolfFE on another environment, let us know and we'll be happy to support you.

wolfssl.com
github.com/wolfssl

wolfEntropy SP 800-90B Software TRNG Source

wolfSSL Version 5.7.0

Description

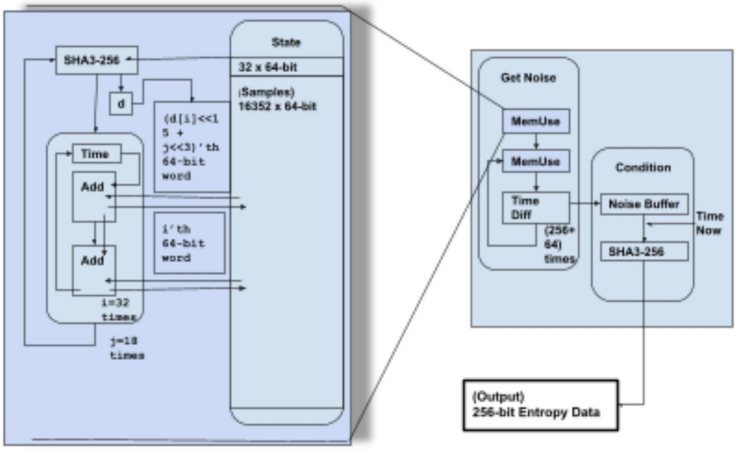
The wolfEntropy library is a software-based source of entropy that utilized timing jitter variations of memory accesses in different cache levels to derive entropy. wolfEntropy was built from scratch by the wolfSSL team. This source of entropy has been available since the release of wolfSSL v5.5.4. You no longer need to redesign your hardware to include hardware based entropy. A high performance software entropy source is available now!

It is designed to fully conform with SP800-90B. SP800-90B is a publication by the National Institute of Standards and Technology (NIST) that specifies the requirements for entropy sources including continuous testing and entropy estimation. In order to generate random numbers that are suitable for cryptographic use, a true random number generator is required.

The wolfEntropy library source is being tested to ensure that it fulfills the requirements of a FIPS 140-3 entropy source, as defined in the publication, and will soon go through **FIPS 140-3 Entropy Source Validation Testing (ESVT)**. For additional information, visit our FIPS FAQ page or contact fips@wolfssl.com.

Features

When wolfSSL is build with the configuration flag ./configure –enable-mem-use and the built-in random number generator (RNG) is used, the HASH_DRBG will make use of wc_Entropy_Get() to obtain a seed source. Here is a block diagram illustrating the operation of the entropy source.



Supported Chipmakers

wolfEntropy supports various chipsets including Intel, RISC-V and ARM, with high-resolution time counters. The wolfSSL team can add support for other chips on demand.

If you would like to use or test wolfEntropy on another chipset, let us know and we'll be happy to support you.

Supported Operating Environments

Linux or Bare Metal (Support for others available on demand)

If you would like to test wolfEntropy on another environment, let us know and we'll be happy to support you.

wolfssl.com
github.com/wolfssl

wolfSSL Adds Support for DO-178 DAL A

Enabling Secure Boot & Secure Firmware Update for Avionics

wolfSSL, provider of the most popular embedded cryptography with over 2 Billion devices secured, is adding support for complete RTCA DO-178C level A certification. wolfSSL will offer DO-178 wolfCrypt as a commercial off-the-shelf (COTS) solution for connected avionics applications. Adherence to DO-178C level A will be supported through the first wolfCrypt COTS DO-178C certification kit release that includes traceable artifacts for the following encryption algorithms:

- SHA-256 for message digest.
- AES for encryption and decryption.
- RSA to sign and verify a message.
- chacha20_poly1305 for authenticated encryption and decryption
- ECC to sign, verify and share secrets
- HMAC for keyed-hashing for message authentication
- A certificate revocation list (CRL)

The primary goal of this initial release is to provide the proper cryptographic underpinnings for **secure boot** and **secure firmware update** in commercial and military avionics. wolfSSL brings trusted, military-grade security to connected commercial and military aircraft. Avionics developers now have a flexible, compact, economical, high-Performance COTS solution for quickly delivering enhanced, secure communications that can be readily certified to DO-178. In addition, FIPS 140-3 and FIPS 140-2 validated crypto algorithms can be used in DO 178 mode for combined FIPS 140-3, FIPS 140-2/DO 178 consumption. The wolfCrypt cryptography library has been FIPS 140-3 validated, with certificate #4718**,** FIPS 140-2 validated (Certificate's #2425 and #3389). For additional information contact fips@wolfssl.com.

Optimization Support

We understand that securely rebooting avionic systems has rigorous performance requirements. As such, we're here to help with cryptographic performance optimizations through our services organization.

Release Plan

- Basic crypto for secure boot and secure firmware updates – Available Now!
- wolfBoot Secure Boot – **Q1, 2023**
- wolfDTLS – **Q3, 2024**
- More wolfCrypt algorithms on demand

Supported Chipmakers

- wolfCrypt has support for chipsets including ARM, Intel, Motorola, mbed, NXP/Freescale, Microchip/Atmel, STMicro, Analog Devices, Texas Instruments, and more.
- If you would like to use or test wolfSSL on another chipset, let us know and we'll be happy to support you.

Supported Operating Environments

Win32/64, Linux, macOS, Solaris, ThreadX, VxWorks, FreeBSD, NetBSD, OpenBSD, embedded Linux, WinCE, Haiku, OpenWRT, iPhone (iOS), Android, Nintendo Wii and Gamecube through DevKitPro, QNX, MontaVista, OpenCL, NonStop, TRON/ITRON/µITRON, Cesium, Micrium's µC/OS, FreeRTOS, SafeRTOS, Freescale MQX, Nucleus, TinyOS, HP/UX, ARC MQX, TI-RTOS, uTasker, embOS, INtime, Mbed, uT-Kernel, RIOT, CMSIS-RTOS, FROSTED, Green Hills INTEGRITY, Keil RTX, TOPPERS, PikeOS

wolfssl.com
github.com/wolfssl

cURL Command Line Tool & Library

Overview

cURL is an open-source command line tool and library written in C with over 5 billion installations worldwide. cURL is used in command lines or scripts to transfer data. It is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, set-top boxes, and media players, and is the internet transfer backbone for thousands of software applications affecting billions of humans every day. The cURL software project provides a library for transferring data using various protocols. These protocols include (but are not limited to) FTP, FTPS, HTTP, HTTPS and more.

Usage

cURL can be used in multiple different ways depending on the desired end result. curl is ideal for secure data transfer. It allows enterprise customers to securely transfer critical business information between users, locations and partners in compliance with data security regulations such as HIPAA, PCI DSS and the EU's GDPR. If the user wants to have a task that repetitively checks the status of a server's HTTP/SSH processes, it can be used to securely download .zip files through a proxy, it has the ability to do this as well.

tinycurl

For resource constrained and embedded users, wolfSSL has created tinycurl! tinycurl is a version of curl that is capable of performing HTTPS and fits within 100K (including the wolfSSL library) on a typical 32-bit architecture. It is approximately one-quarter of the size of the typical curl build on Debian-based Linux with an x86-64 architecture.

Subscription Packages

cURL subscription packages make it easier than ever to use cURL, libcurl, and tiny-curl in non-traditional, embedded, or challenging environments. Includes support for:

- Non-mainstream operating systems
- Embedded and RTOS environments
- Security and application integrations

Supported Protocols

MQTT, DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS,POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET, TFTP, and more.

For more information on using cURL with your project, please contact support@wolfssl.com.

Highlights

- Commercial support offered
- Can be built with wolfCrypt **FIPS 140-3 validated**, with certificate **#4718** and FIPS 140-2/140-3 and FIPS ready
- Optimization support and consulting offered
- Custom cURL builds available
- cURL subscription packages available
- cURL origional author and founder, Daniel Stenberg, part of the wolfSSL team
- Auditable file transfer
- Precision file transfer protocols
- Verbose output options
- Backward compatability
- Platform agnostic
- High performance data transfers
- Customizable feature set
- Extensive documentation
- Open source
- Foundation of support and consulting provided by origional
- cURL author and founder
- Available for dual-licensing

Support Page: www.wolfssl.com/products/support-packages/
Product Page: www.wolfssl.com/products/curl/
GitHub location: github.com/curl/curl

wolfssl.com
github.com/wolfssl

wolfSSL TLS 1.3 Sniffer

Description

The wolfSSL library includes a sniffer with TLS 1.3 support. The wolfSSL sniffer can be used to passively sniff SSL/TLS traffic including https traffic. wolfSSL supports industry standards up to the current **TLS 1.3**, for which we were the first commercial implementation. Our sniffer users benefit from this compatibility with the latest TLS protocol.

For TLS v1.3, all cipher suites use a new ephemeral key for each new session. In order to solve this, we added a “static ephemeral” feature, which allows setting a known key that is used for deriving a shared secret. The key can be rolled periodically and synchronized for internal or test environments. We also have created a Key Manager for secure distribution of ephemeral keys based on the ETSI TS 103 523-3 specification.

Sniffer Features

- Capture and decrypt live or recorded PCAP traces when at least one of the keys is known
- Allows Perfect Forward Secrecy (PFS) ciphers with TLS v1.3
- Includes a test application snifftest
- The wolfSSL sniffer can be integrated into any application using the existing sniffer API. Only five calls are required making it easy to integrate into any project.
- Assembly math enabled for optimized performance
- Ability to decrypt multiple packet streams on different threads

Advantages of TLS 1.3

wolfSSL supports **TLS1.3** on both client and server side. There are many benefits in changing to the newest version of the TLS specification, including:

- Quicker connection times (reduced round-trips during the handshake)
- Reduced latency
- Improved session resumption
- More secure crypto by default
- Check out the **TLS 1.3 documentation** on our website for more information

---

More info

The wolfSSL sniffer requires the wolfSSL library version 1.8.0 or later. The TLS v1.3 sniffer support was added in v4.6.0 or later. The latest release can be obtained from **http://www.wolfssl.com/download**. To evaluate our Key Manager go to **https://github.com/wolfSSL/wolfKeyMgr**.

wolfSSL is powered by the wolfCrypt library. wolfCrypt is **FIPS 140-3 validated**, with certificate **#4718** and **FIPS 140-2 Level 1 validated**, with certificate **#3389**. For additional information, visit our **FIPS FAQ** page or contact **fips@wolfssl.com**.

wolfSSL supports the C programming language as a primary interface, as well as several other host languages including Java (wolfSSL JNI), C# (wolfSSL C#), Python (wolfSSL Python), Ada and PHP and Perl (through a SWIG interface). If you have interest in using wolfSSL and our TLS 1.3 sniffer in another programming language that it does not currently support, please contact wolfSSL at **facts@wolfssl.com**.

For a list of supported operating systems and chipmakers, visit our **product page**.

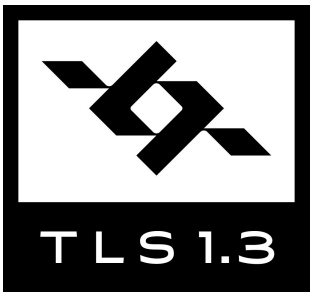
TLS 1.3 Now Available in wolfSSL

Lightweight SSL/TLS library supporting TLS 1.3

The wolfSSL lightweight SSL/TLS library supports **TLS 1.3** (RFC 8446) on both client and server side!

To compile wolfSSL with TLS 1.3 support, use the "**--enable-tls13**" configure option:

```
$ unzip wolfssl-X.X.X.zip
$ cd wolfssl-X.X.X
$ ./configure --enable-tls13
$ make
```



wolfSSL has two client and server side methods, which can be used to specify TLS1.3 during creation of a wolfSSL context (WOLFSSL_CTX):

```
WOLFSSL_METHOD* wolfTLSv1_3_server_method(void);
WOLFSSL_METHOD* wolfTLSv1_3_client_method(void);
```

The wolfSSL example client and server can be used to easily test TLS1.3 functionality. For example, to connect the wolfSSL example client and server to each other using TLS1.3 and the TLS1.3-AES128-GCM-SHA256 cipher suite, use the " -v" option with "4" to specify TLS1.3, and the "-l" option to specify the cipher suite:

```
$ ./examples/server/server -v 4 -l TLS13-AES128-GCM-SHA256
$ ./examples/client/client -v 4 -l TLS13-AES128-GCM-SHA256
```

Alternatively, the example client can be used to connect to an external server. For example, to connect to the wolfSSL website using TLS 1.3:

```
$ ./examples/client/client -v 4 -l TLS13-AES128-GCM-SHA256 \
      -h www.wolfssl.com -p 443 -g -A ./certs/wolfssl-website-ca.pem
```

In thi s command, "-h" specifies the host, "-p" the port, "-g" causes the client to send an HTTP GET request, and "-A" specifies the CA certificate used to authenticate the server.

wolfSSL currently supports the following TLS 1.3 cipher suites:

```
TLS13-AES128-GCM-SHA256
TLS13-AES256-GCM-SHA384
TLS13-CHACHA20-POLY1305-SHA256
TLS13-AES128-CCM-SHA256
TLS13-AES128-CCM-8-SHA256
```

Learn more

For more information on using wolfSSL with TLS 1.3, or to evaluate it, please contact us at facts@wolfssl.com. Please send any comments or feedback on wolfSSL TLS 1.3 support to support@wolfssl.com. Thank you!

wolfssl.com
github.com/wolfssl

Post-Quantum wolfSSL

The wolfSSL library is now safe against the “Harvest Now, Decrypt Later” post-quantum threat model with the addition of our new **TLS 1.3** post-quantum groups.

Hybrid Post Quantum Groups in TLS 1.3

Recently, we announced our own implementations and we have completed hybridization of our ML-KEM with NIST-standardized ECDSA components to continue future-proofing encrypted data streams. These hybridized algorithms continue to be **FIPS compliant** under the current NIST standards.

One approach we are taking involves hybridizing post-quantum algorithms with cryptographic algorithms that we already trust. ECC with NIST standardized curves seem like good candidates, especially since continued FIPS 140-3 compliance is a priority.

To achieve hybridization, we followed the following design:

- The client’s key share is the classical public key concatenated with the post-quantum public key.
- The server’s key share is the classical public key concatenated with the post-quantum ciphertext.
- The shared secret is the classical shared secret concatenated with the post-quantum shared secret.

Post Quantum cURL

wolfSSL is developing a test for **post-quantum cURL** to make cURL resistant to “harvest now; decrypt later” attacks from a future quantum-enabled adversary. This protection is important if you value confidentiality over the long term. This effort involves enabling the use of the new post- quantum groups for TLS 1.3 in cURL when built with wolfSSL.

Research Results from the pq-wolfssl Team

The pq-wolfssl development team has done an excellent experimental post-quantum integration, published in their paper “Mixed Certificate Chains for the Transition to Post-Quantum Authentication in TLS 1.3”.

In the paper, the team “**selected the open source TLS library wolfSSL** (v4.7.0) for our integrations of PQC, because it is suitable for embedded systems and supports TLS 1.3.”

Learn more

For more information on wolfSSL post-quantum projects, please contact us at facts@wolfssl.com.
Please send any comments or feedback to support@wolfssl.com. Thank you!

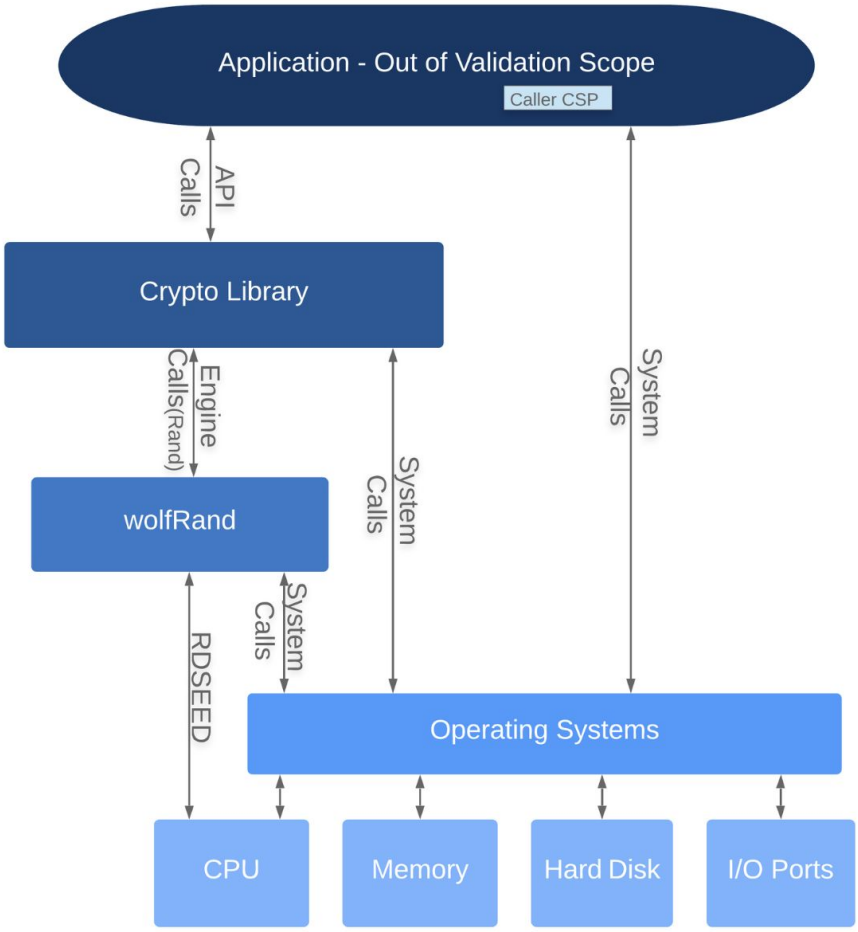
wolfssl.com
github.com/wolfssl

wolfRand Qualified Entropy Source

For Linux/Intel Operating Environments
Release Date January 2020

The Need – FIPS and FedRAMP

Entropy is crucial to information security. Regulators are fully aware that broken entropy means broken encryption and stolen information. As such, entropy assurance from cryptography regulators and labs is becoming more and more stringent, and rightfully so. Projects are now expected to take their entropy assurance to a higher level, and wolfRand is here to help.



wolfRand Description

- 256-bit security strength SP800-90B NDRBG
- FIPS 140-2 cryptographic module suitable for key generation
- Drop-in replacement for /dev/random or /dev/urandom to seed a secondary DRBG
- Conformant to SP 800-90B
- Uses native CPU hardware for strong entropy (available for Intel CPU families with RDSEED) but can be used in conjunction with "software only" validations.

  Also includes HMAC-SHA-256 and SHA-256 services

FAQ

Q: Why is regulatory assurance of sufficient entropy hard?

A: Careful, rigorous entropy assessment requires in depth knowledge of the entropy source design and its properties. The analysis must justify a claim for minimum entropy that is supported by design characteristics and by statistical analysis of the raw (unprocessed) output of the source of randomness. The skill set required to perform this analysis well is a rare commodity. Entropy source design information is hard to obtain. An entropy source must be instrumented to permit gathering raw output of samples for statistical testing. Together, these factors have led to a significant increase in effort when assurance of randomness (minimum entropy) is a regulatory goal.

Q: Why is regulatory assurance of sufficient entropy "about to get even harder"?

A: The introduction of FIPS 140-2 Implementation Guidance articles 7.14 and 7.15 put in place broad guidelines for validation laboratory assessment of entropy, significantly increasing the rigor of entropy assessment. The publication of SP 800-90B *Recommendation for the Entropy Sources Used for Random Bit Generation* has provided a more consistent framework for designers and evaluators of entropy sources to use, but this standardization push has again raised the bar for evaluation of entropy sources. The May 2019 introduction of FIPS 140-2 Implementation Guidance article 7.18 *Entropy Estimation and Compliance with SP 800-90B* mandates conformance to SP 800-90B by November 2020.

Q: What is a qualified entropy source?

A: wolfRand is a FIPS module that includes a hardware entropy source used for seeding material to a FIPS approved SP800-90A DRBG. The wolfRand DRBG output can be used for key generation directly or to create seeding material for another DRBG. The entropy rationale for wolfRand has been reviewed by the CMVP during the FIPS validation process.

Q: When is a qualified entropy source helpful?

A: Common Criteria evaluations require entropy rationale. FIPS 140-2 validations require entropy rationale when the source of entropy is within the cryptographic boundary. FedRAMP reviews are expected to evaluate entropy in greater detail as additional experience is gained by the auditors. wolfRand, as a qualified entropy source, eliminates or simplifies the entropy rationale effort. If you will be asked to provide an entropy rationale in the future, then a qualified entropy source will make your life better.

Q: Will the wolfRand FIPS 140-2 certificate have one of the following caveats?

•No assurance of the minimum strength of generated keys.

•The module generates cryptographic keys whose strengths are modified by available entropy.

A: Most FIPS software modules rely on an entropy source external to the module (for example, /dev/random or /dev/urandom). wolfRand includes the hardware entropy source as part of the module. This eliminates the problem of understanding the strength of external entropy sources. The wolfRand FIPS module will not require either of the caveats above.

•Implementations vary widely from release to release • Complex analysis

Q: What are the supported *Tested Configurations* for wolfRand?

A: wolfRand has been tested on Linux 4.4 (Ubuntu 16.04 LTS) with an Intel® Core™ i5-5300U CPU. Additional configurations may be added to the wolfRand FIPS certificate based on demand and the suitability of the hardware entropy source.

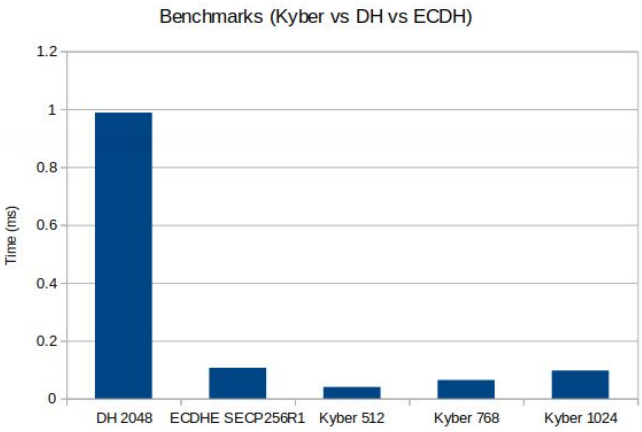
Q: When will the wolfRand FIPS certificate be posted on the NIST website?

A: The wolfRand FIPS certificate is expected in January 2020.

Kyber ML-KEM

wolfssl.com
github.com/wolfssl

wolfSSL Kyber ML-KEM implementation is now included for commercial customers. Future Proofing is here today. We are now including commercial Kyber/ML-KEM in our wolfSSL/wolfCrypt commercial packages.



Post-Quantum Kyber Benchmarks (MacOS)

Platform:
Apple MacBook Pro 18,3 with an Apple M1 Pro, 3.09 GHz processor

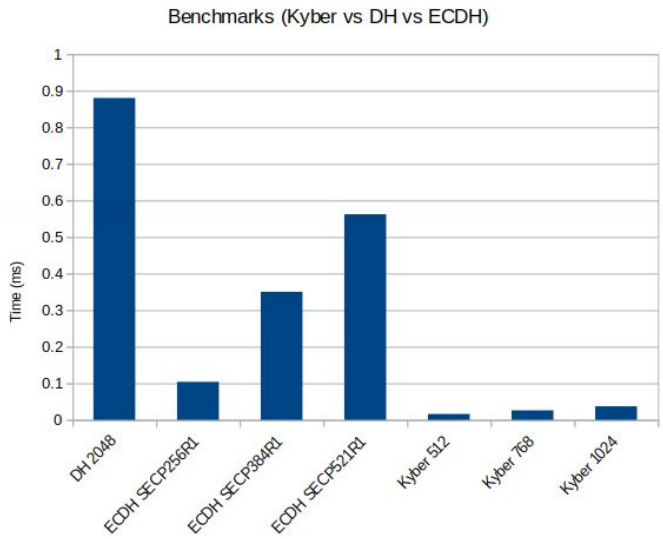
Notes:
- Only 1 core is used

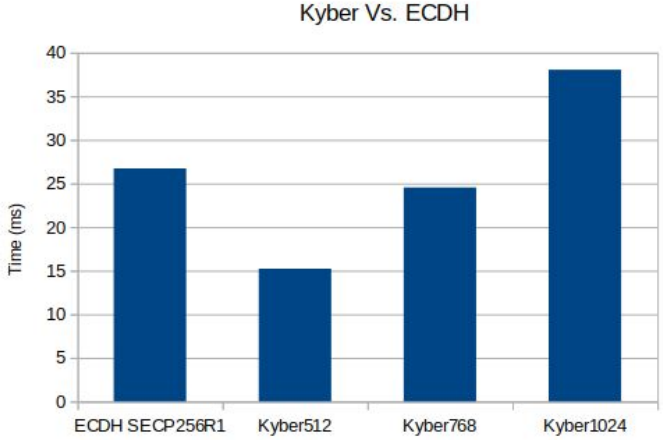
Post-Quantum Kyber Benchmarks (Linux)

Platform:
11th Gen Intel® Core™ i7-1185G7 @ 3.00GHz × 8

Notes:
- Only 1 core is used
- Conventional algorithms are present for comparison purposes
- The wolfSSL configuration used was:

```
./configure --disable-psk --disable-shared --enable-intelasm
 --enable-aesni --enable-sp --enable-sp-math
--enable-sp-asm
 --enable-kyber=wolfssl,all 'CFLAGS=-Os
-DECC_USER_CURVES -DHAVE_ECC256
 -DHAVE_ECC384'
```





Post-Quantum Kyber Benchmarks (ARM Cortex-M4)

Platform:
STM NUCLEO-F446ZE

Notes:
- The HCLK in the project was set to 168MHz
- Only 1 core used
- wolfSSL Math Configuration set to “Single Precision ASM Cortex-M3+ Math”
- Optimization flag: -Ofast
- Conventional algorithm are present for comparison purposes

wolfCrypt FIPS 140-3 Cryptographic Module

FIPS 140-3 Validated, Certificate #4718

FIPS 140-3 Overview

Federal departments and agencies using cryptographic-based security systems to protect sensitive (but unclassified) information are required to implement FIPS 140-3 cryptographic modules. The FIPS 140-3 standard specifies the security requirements for cryptographic modules.

Technology vendors are locked out of serving the federal space (or they place their customers at risk) if they do not have an associated FIPS 140-3 certificate for their cryptographic products.

Accelerate Your FIPS 140-3 Project

wolfCrypt is a cryptographic software API library. Your applications may rely on wolfCrypt to provide all of the cryptographic processing. Instead of performing your own FIPS validation, you may claim that you are using an embedded FIPS cryptographic module. This strategy reduces costs, reduces risks, and accelerates your time-to-market. This will make your Federal customers happy.

wolfCrypt is compliant with FIPS 140-3 Validated,Certificate #4718, Implementation Guidance 10.2.A. The library runs the power-on self-tests automatically with use of a default entry point. The FIPS OpenSSL module does not provide a default entry point.

The wolfSSL team has the FIPS expertise you need. We will form a FIPS strategy that is best for you. Before you search for a FIPS Consultant or begin calling several of the 23 FIPS Laboratories, contact us. We can save you time, money, and effort.

Algorithm	Description
AES	[FIPS 197, SP 800-38A, SP 800-38B, SP 800-38C, SP 800-38D SP 800-38E] Functions: Encryption, Decryption Modes: CBC, CTR, CCM, CMAC, GCM, OFB, ECB, XTS Key sizes: 128, 192, 256 bits
DRBG	[SP 800-90A] Functions: Hash DRBG Security Strength: 256 bits
ECDHE DHE	[FIPS 186-4, RFC7919] Functions: Shared Secret Generation (KAS-SSC-FFC, KAS-SSC-ECC) FFC groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 ECC curves: P-224, P-256, P-384, P-521
ECDSA	[FIPS 186-4] Functions: Key Generation, Signature Generation, Signature Verification SHA sizes: SHA-1 (verification only), SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Curves: P224, P256, P384, P521
RSA	[FIPS 186-4, PKCS #1 v2.1 (PKCS1.5), PSS] Functions: Key Generation, Signature Generation, Signature Verification SHA sizes: SHA-1 (verification only) SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Key sizes: 1024 (verification only), 2048, 3072, 4096
HMAC	[FIPS 198-1] Functions: Generation, Verification SHA sizes: SHA-1 (verification only), SHA2-224, SHA2-256, SHA2-384, SHA3-224, SHA3-256, SHA3-384, SHA3-512
SHA	[FIPS 180-4, FIPS 202] Functions: Message Digest, Secure Hash SHA sizes: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512
TLS KDF TLSv1.3 KDF SSHv2 KDF	[SP 800-135, RFC7627, RFC8446] Functions: Key-Derivation Functions SHA sizes: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512

wolfSSL FIPS Services

- Rebranding of the wolfCrypt FIPS certificate
- Expanded FIPS 140-3 validations (e.g., Level 2, 3, 4)
- Testing of new Operating Environments
- Do you have FIPS 140-3 questions? Give us a test drive by contacting fips@wolfSSL.com

Upcoming efforts:

New module to include SRTP-KDF, AES-GCM Streaming service, AES-CFB (1, 8, 128), AES-XTS (256, 512), AES-KW, PBKDF2, FIPS 186-5, EDDSA (448, 25519), SHAKE (128, 256)

FIPS Post Quantum for NSA 2.0 by 2028!

wolfHSM

Description

Automotive HSMs (Hardware Security Modules) dramatically improve the security of cryptographic keys and cryptographic processing by isolating signature verification and cryptographic execution, which are the core of security, into physically independent processors. Automotive HSMs are mandatory or strongly recommended for ECU's that require robust security. With this in mind, wolfSSL has ported our popular, well tested, and industry leading cryptographic library to run in popular Automotive HSMs like **Aurix Tricore TC3XX and SPC58NN**. However, this is not tied to any specific HSM hardware.

wolfHSM provides a portable and open-source abstraction to hardware cryptography, non-volatile memory, and isolated secure processing that maximizes security and performance for ECUs. By integrating the wolfCrypt software crypto engine on hardware HSM's like Infineon Aurix Tricore TC3XX, Chinese mandated government algorithms like SM2, SM3, SM4 are available. Additionally, Post Quantum Cryptography algos like Kyber, LMS, XMSS and others are easily made available to automotive users to meet customer requirements. At the same time, when hardware cryptographic processing is available on the HSM, we consume it to enhance performance.

Developers will also be able to run wolfHSM in a POSIX simulator for desktop prototyping without any hardware, thus providing true portability and rapid deployment on any hardware.

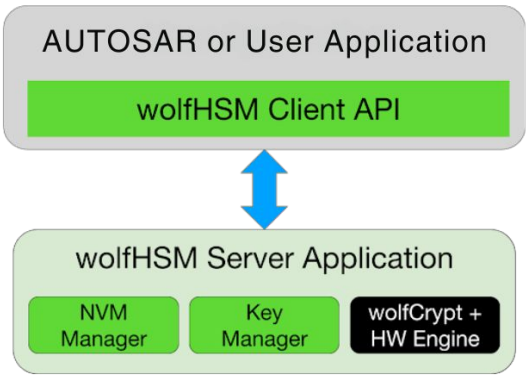
wolfBoot is a mature and portable secure bootloader solution designed for bare-metal bootloaders and equipped with failsafe NVM controls. It offers comprehensive firmware authentication and update mechanisms, leveraging a minimalistic design and a tiny HAL API, which makes it fully independent from any operating system or bare-metal application. wolfBoot manages the flash interface and pre-boot environment, accurately measures and authenticates applications, and utilizes low-level hardware cryptography as needed. wolfBoot can use the wolfHSM client to support HSM-assisted application core secure boot. Additionally, wolfBoot can run on the HSM core to ensure the HSM server is intact, offering a secondary layer protection. This setup ensures a secure boot sequence, aligning well with the booting processes of HSM cores that rely on NVM support.

Features

- Extensibility of cryptographic algorithms
- Consistency with security functions
- Integration with Autosar
- Integration with SHE+
- Direct usage of HSM from wolfCrypt's externalized API's
- PKCS11 interface available
- TPM 2.0 interface available
- Secure OnBoard Communication (SecOC) module integration available
- Certificate handling available
- Symmetric and Asymmetric keys and cryptography
- Customization available
- FIPS 140-3 and DO-178C available

Supported HSM's

- Infineon Aurix TC3xx
- Infineon Aurix TC4x (Coming soon)
- Infineon Traveo T2G (Coming soon)
- ST SPC58NN
- Renesas RH850 (Coming soon)
- Renesas RL78 (Coming soon)



Supported Operating Environments and Chipmakers

wolfHSM operates wherever wolfSSL is supported.

Learn more

For more information on wolfHSM, please contact us at facts@wolfssl.com.

wolfssl.com
github.com/wolfssl

wolfSSL Cybersecure and Compliant Satellite Communication with full FIPS 140-3 and CNSA 2.0 Support

wolfSSL empowers developers of satellite applications with a robust and secure cryptographic solution that meets the demanding requirements of spaceborne systems. Our products, wolfSSL and wolfCrypt, offer a unique combination of FIPS 140-3 validation and CNSA 2.0 support, ensuring the highest levels of cryptographic security for your critical satellite communication.

Unparalleled Security for Demanding Environments:

- **FIPS 140-3 Validation:** wolfCrypt holds the world's first SP800-140Br1 validated FIPS 140-3 certificate (#4718). This rigorous validation guarantees the cryptographic module security, crucial for protecting sensitive satellite data from unauthorized access or manipulation.
- **CNSA 2.0 Support:** wolfSSL prepares you for the future with support for CNSA 2.0's post-quantum cryptography (PQC) algorithms. These advanced algorithms safeguard against potential attacks from quantum computers, a growing threat to traditional cryptography.
- **Proven:** Used by top providers in all parts of the satellite industry; from space vehicles to on the ground systems including launch.
- **Extensive Environment Support:** Full FIPS and CNSA support for all of the hardware and OS's used in the satellite industry.

Advantages for Satellite Applications:

- **Enhanced Security for Sensitive Data:** wolfSSL ensures the confidentiality and integrity of critical satellite data, such as telemetry, command and control information, and scientific observations. This is vital for missions ranging from national security to environmental monitoring.
- **Reduced Risk of System Compromise:** Full FIPS-validated cryptography and PQC support significantly reduce the risk of unauthorized access to your satellite systems, protecting them from cyberattacks and potential disruptions.
- **Compliance with Evolving Security Standards:** wolfSSL helps you stay ahead of the curve by adhering to both FIPS 140-3 and CNSA 2.0, which are increasingly being mandated for government and defense satellite programs.
- **Lightweight and Efficient:** wolfSSL is known for its small footprint and efficient resource utilization. This is critical for resource-constrained satellite environments where power and processing power are limited.

Additional Benefits:

- **Simplified Development:** Achieve compliance with both FIPS 140-3 and CNSA 2.0 through a single, secure TLS 1.3 connection. This streamlines development and reduces integration time.
- **Long-Term Security:** PQC algorithms prepare your satellite systems for the potential challenges of quantum computing, ensuring continued secure communication well into the future.
- **Expert Support:** wolfSSL's team of security professionals offers guidance and support in achieving FIPS 140-3 and CNSA 2.0 compliance within your satellite applications.

Conclusion:

By choosing wolfSSL, you gain a powerful and secure cryptographic foundation for your satellite applications. With FIPS 140-3 validation and CNSA 2.0 support, wolfSSL empowers you to meet the most stringent security requirements while ensuring efficient resource utilization in the demanding space environment.

Contact wolfSSL Today:

For further inquiries or to explore how wolfSSL can elevate the security and compliance of your satellite communication, contact our team at facts@wolfSSL.com or +1 425 245 8247.