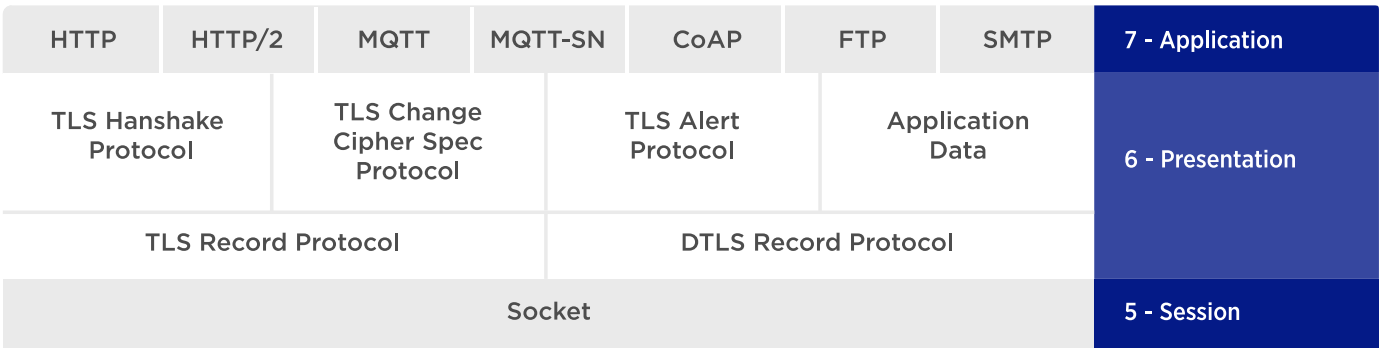
CycloneSSL

CycloneSSL is a lightweight TLS / DTLS implementation targeted for use by embedded application developers. It provides the ability to secure communications over the Internet (e.g. IoT protocols, electronic mail, web server, file transfer, VoIP).



Main Features

- Server and/or client operation
- Supports TLS 1.0, TLS 1.1, TLS 1.2 and TLS 1.3 protocols
- Supports DTLS 1.0 and DTLS 1.2 (Datagram Transport Layer Security)
- Supports QUIC transport layer
- Robust and efficient implementation
- Supports ECC (Elliptic Curve Cryptography)
- Rich set of TLS cipher suites (including Suite B and CNSA profiles)
- RSA, Diffie-Hellman and ECDH key exchange algorithms
- Post-quantum key agreement
- Compliant with BSD socket API
- Flexible memory footprint. Built-time configuration to embed only the necessary features
- Consistent application programming interface (API)
- Portable architecture (no processor dependencies)
- The library is distributed as a full ANSI C and highly maintainable source code

Supported Algorithms

- ECDH key exchange based on Curve25519 (X25519) and Curve448 (X448)
- FFDHE (Finite Field Diffie-Hellman Ephemeral)
- ML-KEM post-quantum key exchange algorithms (ML-KEM-512, ML-KEM-768 and ML-KEM-1024)
- PQ-hybrid key exchange algorithms (SecP256r1MLKEM768, SecP384r1MLKEM1024 and X25519MLKEM768)
- Supports PSK (Pre-Shared Key) cipher suites
- RSA signature schemes (RSASSA PKCS#1 v1.5 and RSASSA-PSS)
- DSA and ECDSA signature schemes
- EdDSA signature scheme (Ed25519 and Ed448 elliptic curves)
- Supports stream ciphers and CBC block ciphers
- Cipher Block Chaining-MAC (CCM) and Galois Counter Mode (GCM)
- ChaCha20Poly1305 Authenticated Encryption with Associated Data (AEAD)
- Supports AES, Camellia, SEED and ARIA encryption algorithms
- Legacy support for RC4, IDEA, DES and 3DES encryption algorithms
- Supports SHA-256, SHA-384 and SHA-512 hash algorithms
- Legacy support for MD5 and SHA-1 hash algorithms
- Supports ShangMi (SM) cipher suites for TLS 1.3
- Session resumption mechanism
- Session ticket mechanism (RFC 5077)
- Supports secure renegotiation (RFC 5746)
- TLS channel binding (RFC 5929)
- Fallback SCSV signaling cipher suite
- SNI extension (Server Name Indication)
- Raw Public Keys (RFC 7250)
- Maximum Fragment Length extension (RFC 6066)
- Record Size Limit extension (RFC 8449)
- Application-Layer Protocol Negotiation (ALPN) extension
- Encrypt-Then-MAC extension
- Extended Master Secret extension
- ClientHello Padding extension (RFC 7685)
- (EC)DHE and PSK key establishment (TLS 1.3)
- Middlebox compatibility mode (TLS 1.3)
- Key update mechanism (TLS 1.3)
- Early data (TLS 1.3 client)
- X.509 certificate parsing and PKIX path validation
- Parsing of public/private keys (PKCS #1 and PKCS #8 formats supported)
- Parsing of encrypted private keys (PKCS #1 and PKCS #8 formats supported)

Supported Extensions

- session_ticket (RFC 5077)
- renegotiation_info (RFC 5746)
- server_name (RFC 6066)
- max_fragment_length (RFC 6066)
- trusted_ca_keys (RFC 6066)
- client_certificate_type (RFC 7250)
- server_certificate_type (RFC 7250)
- application_layer_protocol_negotiation (RFC 7301)
- encrypt_then_mac (RFC 7366)
- extended_master_secret (RFC 7627)
- padding (RFC 7685)
- supported_groups (RFC 7919)
- ec_point_formats (RFC 8422)
- signature_algorithms (RFC 8446)
- pre_shared_key (RFC 8446)
- early_data (RFC 8446)
- supported_versions (RFC 8446)
- cookie (RFC 8446)
- psk_key_exchange_modes (RFC 8446)
- certificate_authorities (RFC 8446)
- signature_algorithms_cert (RFC 8446)
- key_share (RFC 8446)
- record_size_limit (RFC 8449)
- quic_transport_parameters (RFC 9001)

RFC

- RFC 2246: The TLS Protocol Version 1.0
- RFC 3268: Advanced Encryption Standard (AES) Cipher Suites for TLS
- RFC 4162: Addition of SEED Cipher Suites to Transport Layer Security (TLS)
- RFC 4279: Pre-Shared Key Cipher Suites for Transport Layer Security (TLS)
- RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1
- RFC 4347: Datagram Transport Layer Security (DTLS)
- RFC 4492: Elliptic Curve Cryptography (ECC) Cipher Suites for TLS
- RFC 5077: Transport Layer Security (TLS) Session Resumption without Server-Side State
- RFC 5116: An Interface and Algorithms for Authenticated Encryption
- RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2
- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for TLS
- RFC 5289: TLS ECC Cipher Suites with SHA-256/384 and AES Galois Counter Mode
- RFC 5469: DES and IDEA Cipher Suites for Transport Layer Security (TLS)
- RFC 5487: PSK Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode
- RFC 5489: ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)
- RFC 5746: TLS Renegotiation Indication Extension
- RFC 5929: Channel Bindings for TLS
- RFC 5932: Camellia Cipher Suites for TLS
- RFC 6066: Transport Layer Security (TLS) Extensions: Extension Definitions
- RFC 6101: The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC 6176: Prohibiting Secure Sockets Layer (SSL) Version 2.0
- RFC 6209: Addition of the ARIA Cipher Suites to Transport Layer Security (TLS)
- RFC 6347: Datagram Transport Layer Security Version 1.2
- RFC 6367: Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)
- RFC 6460: Suite B Profile for Transport Layer Security (TLS)
- RFC 6655: AES-CCM Cipher Suites for Transport Layer Security (TLS)
- RFC 7027: Elliptic Curve Cryptography (ECC) Brainpool Curves for TLS
- RFC 7250: Using Raw Public Keys in TLS and DTLS
- RFC 7251: AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS
- RFC 7301: TLS Application-Layer Protocol Negotiation Extension
- RFC 7366: Encrypt-then-MAC for TLS and DTLS
- RFC 7465: Prohibiting RC4 Cipher Suites
- RFC 7507: TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks
- RFC 7525: Recommendations for Secure Use of TLS and DTLS
- RFC 7539: ChaCha20 and Poly1305 for IETF Protocols
- RFC 7568: Deprecating Secure Sockets Layer Version 3.0
- RFC 7627: TLS Session Hash and Extended Master Secret Extension
- RFC 7685: A Transport Layer Security (TLS) ClientHello Padding Extension
- RFC 7905: ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)
- RFC 7919: Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS
- RFC 8422: ECC Cipher Suites for TLS Versions 1.2 and Earlier
- RFC 8442: ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2
- RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3
- RFC 8447: IANA Registry Updates for TLS and DTLS
- RFC 8449: Record Size Limit Extension for TLS
- RFC 8734: Elliptic Curve Cryptography (ECC) Brainpool Curves for TLS Version 1.3
- RFC 8996: Deprecating TLS 1.0 and TLS 1.1
- RFC 8998: ShangMi (SM) Cipher Suites for TLS 1.3
- RFC 9001: Using TLS to Secure QUIC
- RFC 9150: TLS 1.3 Authentication and Integrity-Only Cipher Suites
- RFC 9151: Commercial National Security Algorithm (CNSA) Suite Profile for TLS and DTLS 1.2 and 1.3
- RFC 9155: Deprecating MD5 and SHA-1 Signature Hashes in TLS 1.2 and DTLS 1.2
- RFC 9325: Recommendations for Secure Use of TLS and DTLS
- RFC draft: ML-KEM Post-Quantum Key Agreement for TLS 1.3
- RFC draft: Hybrid key exchange in TLS 1.3
- RFC draft: Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3
- RFC draft: Hybrid Post-quantum Key Exchange SM2-MLKEM for TLSv1.3
- RFC draft: The SSLKEYLOGFILE Format for TLS

NIST

- SP 800-52: Guidelines for the Selection and Use of TLS Implementations

Supported Processors

- ARM Cortex-M3
- ARM Cortex-M4
- ARM Cortex-M7
- ARM Cortex-M33
- ARM Cortex-M55
- ARM Cortex-M85
- ARM Cortex-R4
- ARM Cortex-A5
- ARM Cortex-A7
- ARM Cortex-A8
- ARM Cortex-A9
- ARM Cortex-A55
- RISC-V
- MIPS M4K
- MIPS microAptiv / M-Class
- Infineon TriCore AURIX
- PowerPC e200
- Coldfire V2
- RX600
- AVR32
- Xtensa LX6

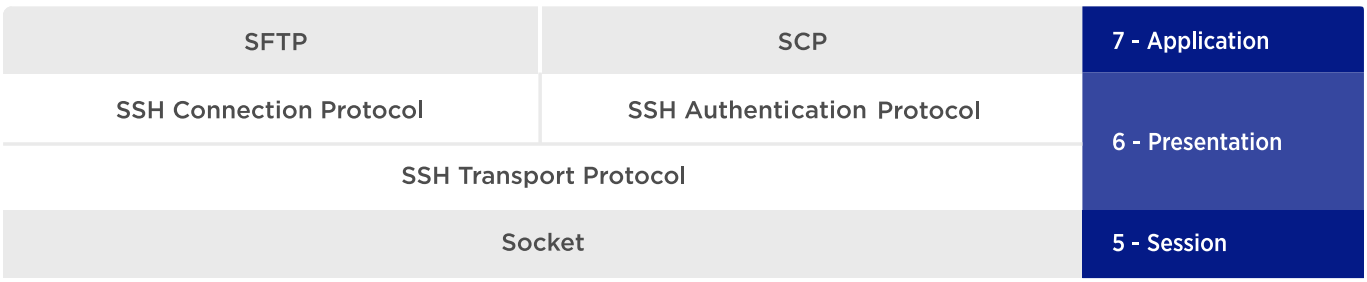
Supported Operating Systems

- Amazon FreeRTOS
- SafeRTOS
- ChibiOS/RT
- CMSIS-RTOS
- CMSIS-RTOS2
- CMX-RTX
- Keil RTXv4 and RTXv5
- Micrium µC/OS-II and µC/OS-III
- Eclipse ThreadX
- PX5 RTOS
- Segger embOS
- TI-RTOS (SYS/BIOS)
- Zephyr RTOS
- Bare Metal programming (without RTOS)

Supported Compilers / Toolchains

Toolchain / IDE	Compiler
Makefile	GCC
AC6 System Workbench for STM32 (SW4STM32)	GCC
Atollic TrueSTUDIO	GCC
Espressif ESP-IDF	GCC
HighTec Toolset for TriCore	GCC
IAR Embedded Workbench	EWARM, EWRX
Infineon DAVE	GCC
Keil MDK-ARM	ARM Compiler v5, ARM Compiler v6 (CLANG)
Microchip Studio (Atmel Studio)	GCC
Microchip MPLAB X	GCC, XC32
Microsoft Visual Studio	MSVC
NXP MCUXpresso	GCC
NXP S32 Design Studio (S32DS)	GCC
Renesas e2Studio	GCC, CC-RX
Segger Embedded Studio	GCC
ST STM32CubeIDE	GCC
Tasking VX-Toolset	VX-Toolset for TriCore

CycloneSSH is a SSHv2 library designed for embedded applications. It can be used to operate network services such as remote shell and file transfer over an unsecured network. The authentication layer of SSH uses public-key cryptography to authenticate the remote machine. The transport layer of SSH provides confidentiality and integrity of data exchanged between the client and server.



Main Features

- SSH version 2.0 implementation
- Client and server modes of operation
- Password and public key user authentication methods
- Password change mechanism supported at server side
- Supports certificate-based authentication (using OpenSSH certificate format)
- Secure shell client and server (for remote execution of commands)
- SCP client and server
- SFTP client and server
- Key exchange using RSA, Diffie-Hellman, DH GEX, ECDH, Curve25519 and Curve448 algorithms
- Pure post-quantum key exchange (using ML-KEM-512, ML-KEM-768 and ML-KEM-1024)
- Post-quantum hybrid key exchange (using ML-KEM-768, ML-KEM-1024 and SNTRUP761)
- Strict key exchange extension
- RSA, DSA, ECDSA, Ed25519 and Ed448 host key algorithms
- 3DES, AES, Twofish, Serpent, Camellia, SEED and ChaCha20Poly1305 encryption algorithms
- Legacy support for RC4, CAST-128, IDEA and Blowfish encryption algorithms
- CBC, CTR and GCM encryption modes
- HMAC using SHA-1, SHA-256 or SHA-512
- Legacy support for MD5 and RIPEMD-160 hash algorithms
- Supports Encrypt-then-MAC (EtM) construction
- Elliptic Curve Cryptography (ECC) supported
- Commercial National Security Algorithm (CNSA) suite cryptography
- SSH extension negotiation mechanism
- Support for "server-sig-algs" and "global-requests-ok" extensions
- Parsing and formatting of SSH public keys (SSH2 and OpenSSH formats supported)
- Parsing and formatting of SSH private keys (OpenSSH format supported)
- Parsing of encrypted OpenSSH private keys
- Flexible memory footprint. Built-time configuration to embed only the necessary features
- Portable architecture (no processor dependencies)
- The library is distributed as a full ANSI C and highly maintainable source code

Pure PQ Key Exchange Algorithms

- mlkem512-sha256
- mlkem768-sha256
- mlkem1024-sha384

PQ-Hybrid Key Exchange Algorithms

- mlkem768nistp256-sha256
- mlkem1024nistp384-sha384
- mlkem768x25519-sha256
- sntrup761x25519-sha512
- sntrup761x25519-sha512@openssh.com

Key Exchange Algorithms

- rsa1024-sha1(w)
- rsa2048-sha256
- diffie-hellman-group1-sha1(w)
- diffie-hellman-group14-sha1(w)
- diffie-hellman-group14-sha256
- diffie-hellman-group15-sha512
- diffie-hellman-group16-sha512
- diffie-hellman-group17-sha512
- diffie-hellman-group18-sha512
- diffie-hellman-group-exchange-sha1(w)
- diffie-hellman-group-exchange-sha224@ssh.com
- diffie-hellman-group-exchange-sha256
- diffie-hellman-group-exchange-sha384@ssh.com
- diffie-hellman-group-exchange-sha512@ssh.com
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521
- curve25519-sha256
- curve25519-sha256@libssh.org
- curve448-sha512

Host Key Algorithms

- ssh-dss(w)
- ssh-rsa(w)
- rsa-sha2-256
- rsa-sha2-512
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp521
- ssh-ed25519
- ssh-ed448
- ssh-dss-cert(w)
- ssh-dss-cert-v01@openssh.com(w)
- ssh-rsa-cert(w)
- ssh-rsa-cert-v01@openssh.com(w)
- rsa-sha2-256-cert
- rsa-sha2-256-cert-v01@openssh.com
- rsa-sha2-512-cert
- rsa-sha2-512-cert-v01@openssh.com
- ecdsa-sha2-nistp256-cert
- ecdsa-sha2-nistp256-cert-v01@openssh.com
- ecdsa-sha2-nistp384-cert
- ecdsa-sha2-nistp384-cert-v01@openssh.com
- ecdsa-sha2-nistp521-cert
- ecdsa-sha2-nistp521-cert-v01@openssh.com
- ssh-ed25519-cert
- ssh-ed25519-cert-v01@openssh.com
- ssh-ed448-cert

(†) denotes insecure algorithms
(w) denotes weak algorithms

Pure PQ Key Exchange Algorithms

- mlkem512-sha256
- mlkem768-sha256
- mlkem1024-sha384

Encryption Algorithms

- arcfour(†)
- arcfour128(†)
- arcfour256(†)
- cast128-cbc(†)
- cast128-ctr(†)
- idea-cbc(†)
- idea-ctr(†)
- blowfish-cbc(†)
- blowfish-ctr(†)
- 3des-cbc(w)
- 3des-ctr(w)
- aes128-cbc
- aes128-ctr
- aes192-cbc
- aes192-ctr
- aes256-cbc
- aes256-ctr
- camellia128-cbc
- camellia128-ctr
- camellia192-cbc
- camellia192-ctr
- camellia256-cbc
- camellia256-ctr
- serpent128-cbc
- serpent128-ctr
- serpent192-cbc
- serpent192-ctr
- serpent256-cbc
- serpent256-ctr
- twofish128-cbc
- twofish128-ctr
- twofish192-cbc
- twofish192-ctr
- twofish256-cbc
- twofish256-ctr
- twofish-cbc
- seed-cbc@ssh.com
- aes128-gcm
- aes128-gcm@openssh.com
- aes256-gcm
- aes256-gcm@openssh.com
- AEAD_AES_128_GCM
- AEAD_AES_256_GCM
- AEAD_CAMELLIA_128_GCM
- AEAD_CAMELLIA_256_GCM
- chacha20-poly1305
- chacha20-poly1305@openssh.com

MAC Algorithms

- hmac-md5(†)
- hmac-md5-etm@openssh.com(†)
- hmac-md5-96(†)
- hmac-md5-96-etm@openssh.com(†)
- hmac-ripemd160@openssh.com(w)
- hmac-ripemd160-etm@openssh.com(w)
- hmac-sha1(w)
- hmac-sha1-etm@openssh.com(w)
- hmac-sha1-96(†)
- hmac-sha1-96-etm@openssh.com(†)
- hmac-sha2-256
- hmac-sha2-256-etm@openssh.com
- hmac-sha2-512
- hmac-sha2-512-etm@openssh.com

(†) denotes insecure algorithms

(w) denotes weak algorithms

SSH Core

- RFC 4250: The Secure Shell (SSH) Protocol Assigned Numbers
- RFC 4251: The Secure Shell (SSH) Protocol Architecture
- RFC 4252: The Secure Shell (SSH) Authentication Protocol
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254: The Secure Shell (SSH) Connection Protocol

SSH Extensions

- RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
- RFC 4335: The Secure Shell (SSH) Session Channel Break Extension
- RFC 4344: The Secure Shell (SSH) Transport Layer Encryption Modes
- RFC 4345: Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4419: Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4432: RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4716: The Secure Shell (SSH) Public Key File Format
- RFC 5647: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol
- RFC 5656: Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer
- RFC 6668: SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol
- RFC 8268: More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for SSH
- RFC 8270: Increase the Secure Shell Minimum Recommended Diffie-Hellman Modulus Size to 2048 Bits
- RFC 8308: Extension Negotiation in the Secure Shell (SSH) Protocol
- RFC 8332: Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol
- RFC 8709: Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol
- RFC 8731: Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448
- RFC 8758: Deprecating RC4 in Secure Shell (SSH)
- RFC 9142: Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)
- RFC 9212: Commercial National Security Algorithm (CNSA) Suite Cryptography for Secure Shell (SSH)
- RFC 9519: Update to the IANA SSH Protocol Parameters Registry Requirements
- RFC draft: Camellia cipher for the Secure Shell Transport Layer Protocol
- RFC draft: Fixed AES-GCM modes for the SSH protocol
- RFC draft: Secure Shell (SSH) authenticated encryption cipher: chacha20-poly1305
- RFC draft: Module-Lattice Key Exchange in SSH
- RFC draft: PQ/T Hybrid Key Exchange in SSH
- RFC draft: SSH KEX Method Using Hybrid Streamlined NTRU Prime sntrup761 and X25519 with SHA-512
- RFC draft: SSH Certificate Format
- RFC draft: SSH Strict KEX extension
- RFC draft: Sending and Handling of Global Requests in Secure Shell (SSH)

SFTP

- RFC draft: SSH File Transfer Protocol (version 3)

Vendor Extensions (OpenSSH)

- PROTOCOL: Encrypt-then-MAC MAC Algorithms
- PROTOCOL.key: OpenSSH Private Key Format
- PROTOCOL.certkeys: Simple Public-Key Certificate Authentication System
- PROTOCOL.chacha20poly1305: chacha20-poly1305@openssh.com Authenticated Encryption Cipher
- PROTOCOL: Strict key exchange extension

Supported Processors

- ARM Cortex-M3
- ARM Cortex-M4
- ARM Cortex-M7
- ARM Cortex-M33
- ARM Cortex-M55
- ARM Cortex-M85
- ARM Cortex-R4
- ARM Cortex-A5
- ARM Cortex-A7
- ARM Cortex-A8
- ARM Cortex-A9
- ARM Cortex-A55
- RISC-V
- MIPS M4K
- MIPS microAptiv / M-Class
- Infineon TriCore AURIX
- PowerPC e200
- Coldfire V2
- RX600
- AVR32
- Xtensa LX6

Supported Operating Systems

- Amazon FreeRTOS
- SafeRTOS
- ChibiOS/RT
- CMSIS-RTOS
- CMSIS-RTOS2
- CMX-RTX
- Keil RTXv4 and RTXv5
- Micrium µC/OS-II and µC/OS-III
- Eclipse ThreadX
- PX5 RTOS
- Segger embOS
- TI-RTOS (SYS/BIOS)
- Zephyr RTOS
- Bare Metal programming (without RTOS)

Supported Compilers / Toolchains

Toolchain / IDE	Compiler
Makefile	GCC
AC6 System Workbench for STM32 (SW4STM32)	GCC
Atollic TrueSTUDIO	GCC
Espressif ESP-IDF	GCC
HighTec Toolset for TriCore	GCC
IAR Embedded Workbench	EWARM, EWRX
Infineon DAVE	GCC
Keil MDK-ARM	ARM Compiler v5, ARM Compiler v6 (CLANG)
Microchip Studio (Atmel Studio)	GCC
Microchip MPLAB X	GCC, XC32
Microsoft Visual Studio	MSVC
NXP MCUXpresso	GCC
NXP S32 Design Studio (S32DS)	GCC
Renesas e2Studio	GCC, CC-RX
Segger Embedded Studio	GCC
ST STM32CubeIDE	GCC
Tasking VX-Toolset	VX-Toolset for TriCore

CycloneIPSEC

CycloneIPSEC is an IPsec / IKEv2 library designed for embedded applications. IPsec is a suite of protocols used to implement secure communication between two sites over the Internet. IPsec operates at the network layer of the OSI model. The main protocols comprising IPsec are AH (Authentication Header), ESP (Encapsulating Security Payload) and IKEv2 (Internet Key Exchange version 2). AH provides data integrity protection while ESP provides both confidentiality and data integrity protection. IKEv2 is the protocol used to manage security associations between two entities.

Main Features

- AH (Authentication Header) implementation
- ESP (Encapsulating Security Payload) implementation
- IKEv2 (Internet Key Exchange version 2) implementation
- Supports Transport mode over IPv4 (Tunnel mode is not supported)
- Pre-shared key and certificate authentication methods
- Key exchange using Diffie-Hellman, ECDH, Curve25519 and Curve448 algorithms
- RSA, RSA-PSS, DSA, ECDSA, Ed25519 and Ed448 signature algorithms
- AES, Camellia and ChaCha20Poly1305 encryption algorithms
- Legacy support for IDEA, DES and 3DES encryption algorithms
- CBC, CTR, CCM and GCM encryption modes
- SHA-256, SHA-384 and SHA-512 hash algorithms
- Legacy support for MD5, SHA-1 and Tiger hash algorithms
- Commercial National Security Algorithm (CNSA) suite cryptography
- Anti-replay mechanism with configurable sliding window size (64 by default)
- HMAC, CMAC and XCBC-MAC integrity algorithms
- Supportes ESNs (Extended Sequence Numbers)
- Cookie generation and verification
- Supports Digital Signature method
- Supports SIGNATURE_HASH_ALGORITHMS and INITIAL_CONTACT notifications
- Supports DPD (Dead Peer Detection) mechanism
- Flexible memory footprint. Built-time configuration to embed only the necessary features
- Portable architecture (no processor dependencies)
- The library is distributed as a full ANSI C and highly maintainable source code

Key Exchange Methods

- 768-bit MODP Group(†)
- 1024-bit MODP Group(w)
- 1536-bit MODP Group(w)
- 2048-bit MODP Group
- 3072-bit MODP Group
- 4096-bit MODP Group
- 6144-bit MODP Group
- 8192-bit MODP Group
- 192-bit random ECP Group
- 224-bit random ECP Group
- 256-bit random ECP Group
- 384-bit random ECP Group
- 521-bit random ECP Group
- 224-bit Brainpool ECP Group
- 256-bit Brainpool ECP Group
- 384-bit Brainpool ECP Group
- 512-bit Brainpool ECP Group
- Curve25519
- Curve448

Authentication Methods

- Shared Key Message Integrity Code
- RSA Digital Signature
- DSS Digital Signature
- ECDSA with SHA-256 on P-256 Curve
- ECDSA with SHA-384 on P-384 Curve
- ECDSA with SHA-512 on P-521 Curve
- Digital Signature

Pseudorandom Functions

- PRF_HMAC_MD5(†)
- PRF_HMAC_SHA1(w)
- PRF_HMAC_TIGER(w)
- PRF_HMAC_SHA2_256
- PRF_HMAC_SHA2_384
- PRF_HMAC_SHA2_512
- PRF_AES128_CMAC
- PRF_AES128_XCBC

Encryption Algorithms

- ENCR_IDEA(†)
- ENCR_DES(†)
- ENCR_3DES(w)
- ENCR_AES_CBC
- ENCR_AES_CTR
- ENCR_AES_CCM_8
- ENCR_AES_CCM_12
- ENCR_AES_CCM_16
- ENCR_AES_GCM_8
- ENCR_AES_GCM_12
- ENCR_AES_GCM_16
- ENCR_CAMELLIA_CBC
- ENCR_CAMELLIA_CTR
- ENCR_CAMELLIA_CCM_8
- ENCR_CAMELLIA_CCM_12
- ENCR_CAMELLIA_CCM_16
- ENCR_CHACHA20_POLY1305

Integrity Algorithms

- AUTH_HMAC_MD5_96(†)
- AUTH_HMAC_SHA1_96(w)
- AUTH_HMAC_SHA2_256_128
- AUTH_HMAC_SHA2_384_192
- AUTH_HMAC_SHA2_512_256
- AUTH_AES_CMAC_96
- AUTH_AES_XCBC_96

(†) denotes insecure algorithms

(w) denotes weak algorithms

IPsec Core

- RFC 4301: Security Architecture for the Internet Protocol
- RFC 4302: IP Authentication Header
- RFC 4303: IP Encapsulating Security Payload (ESP)

IPsec Extensions

- RFC 2451: The ESP CBC-Mode Cipher Algorithms
- RFC 3566: The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec
- RFC 3602: The AES-CBC Cipher Algorithm and Its Use with IPsec
- RFC 3686: Using Advanced Encryption Standard (AES) Counter Mode With IPsec ESP
- RFC 4106: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
- RFC 4308: Cryptographic Suites for IPsec
- RFC 4309: Using Advanced Encryption Standard (AES) CCM Mode with IPsec ESP
- RFC 4494: The AES-CMAC-96 Algorithm and Its Use with IPsec
- RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec
- RFC 5529: Modes of Operation for Camellia for Use with IPsec
- RFC 6379: Suite B Cryptographic Suites for IPsec
- RFC 6380: Suite B Profile for Internet Protocol Security (IPsec)
- RFC 7634: ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec
- RFC 8221: Cryptographic Algorithm Implementation Requirements and Usage Guidance for ESP and AH
- RFC 9206: Commercial National Security Algorithm (CNSA) Suite Cryptography for IPsec

IKE Core

- RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)

IKE Extensions

- RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
- RFC 4307: Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)
- RFC 4434: The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)
- RFC 4615: The AES-CMAC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)
- RFC 4718: IKEv2 Clarifications and Implementation Guidelines
- RFC 4753: ECP Groups For IKE and IKEv2
- RFC 4754: IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)
- RFC 4945: The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX
- RFC 5282: Using Authenticated Encryption Algorithms with the Encrypted Payload of the IKEv2 Protocol
- RFC 5903: Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2
- RFC 5930: Using Advanced Encryption Standard Counter Mode (AES-CTR) with the IKEv2 Protocol
- RFC 6932: Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry
- RFC 6954: Using the Elliptic Curve Cryptography (ECC) Brainpool Curves for the IKEv2
- RFC 6989: Additional Diffie-Hellman Tests for the IKEv2
- RFC 7427: Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)
- RFC 7815: Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation
- RFC 8031: Curve25519 and Curve448 for the IKEv2 Key Agreement
- RFC 8247: Algorithm Implementation Requirements and Usage Guidance for the IKEv2
- RFC 8420: Using the Edwards-Curve Digital Signature Algorithm (EdDSA) in the IKEv2
- RFC 9395: Deprecation of the IKEv1 Protocol and Obsoleted Algorithms
- RFC 9827: Renaming the Extended Sequence Numbers (ESN) Transform Type in the IKEv2

Supported Processors

- ARM Cortex-M3
- ARM Cortex-M4
- ARM Cortex-M7
- ARM Cortex-M33
- ARM Cortex-M55
- ARM Cortex-M85
- ARM Cortex-R4
- ARM Cortex-A5
- ARM Cortex-A7
- ARM Cortex-A8
- ARM Cortex-A9
- ARM Cortex-A55
- RISC-V
- MIPS M4K
- MIPS microAptiv / M-Class
- Infineon TriCore AURIX
- PowerPC e200
- Coldfire V2
- RX600
- AVR32
- Xtensa LX6

Supported Operating Systems

- Amazon FreeRTOS
- SafeRTOS
- ChibiOS/RT
- CMSIS-RTOS
- CMSIS-RTOS2
- CMX-RTX
- Keil RTXv4 and RTXv5
- Micrium µC/OS-II and µC/OS-III
- Eclipse ThreadX
- PX5 RTOS
- Segger embOS
- TI-RTOS (SYS/BIOS)
- Zephyr RTOS
- Bare Metal programming (without RTOS)

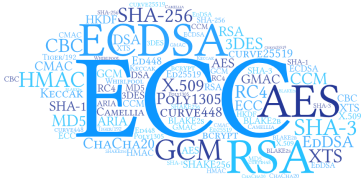
Supported Compilers / Toolchains

Toolchain / IDE	Compiler
Makefile	GCC
AC6 System Workbench for STM32 (SW4STM32)	GCC
Atollic TrueSTUDIO	GCC
Espressif ESP-IDF	GCC
HighTec Toolset for TriCore	GCC
IAR Embedded Workbench	EWARM, EWRX
Infineon DAVE	GCC
Keil MDK-ARM	ARM Compiler v5, ARM Compiler v6 (CLANG)
Microchip Studio (Atmel Studio)	GCC
Microchip MPLAB X	GCC, XC32
Microsoft Visual Studio	MSVC
NXP MCUXpresso	GCC
NXP S32 Design Studio (S32DS)	GCC
Renesas e2Studio	GCC, CC-RX
Segger Embedded Studio	GCC
ST STM32CubeIDE	GCC
Tasking VX-Toolset	VX-Toolset for TriCore

CycloneCRYPTO

CycloneCRYPTO is a cryptographic toolkit designed for use in embedded systems. It provides a comprehensive set of cryptographic primitives (hash functions, stream and block ciphers, public key cryptography) that can be used to add security features to your embedded application.

Main Features



- Base64 encoding
- MD2, MD4 and MD5 hash functions
- RIPEMD-128 and RIPEMD-160 hash functions
- SHA-1 hash function
- SHA-2 family hash functions (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256)
- SHA-3 family hash functions (SHA3-224, SHA3-256, SHA3-384 and SHA3-512)
- BLAKE2b family hash functions (BLAKE2b160, BLAKE2b256, BLAKE2b384, BLAKE2b512)
- BLAKE2s family hash functions (BLAKE2s128, BLAKE2s160, BLAKE2s224, BLAKE2s256)
- Tiger/192 hash function
- Whirlpool hash function
- SHAKE128, SHAKE256 and cSHAKE extendable-output functions (XOF)
- Keccak sponge function
- HMAC, CMAC, GMAC, KMAC, XCBC-MAC and Poly1305 message-authentication code
- Supports ChaCha, Salsa20, Trivium and ZUC stream ciphers
- Legacy support for RC4 stream ciphers
- Supports 128-bit block ciphers (RC6, CAST-256, AES, Twofish, MARS, Serpent, Camellia, ARIA, SEED)
- Legacy support for 64-bit block ciphers (RC2, CAST-128, IDEA, DES, 3DES, Blowfish, PRESENT, TEA, XTEA)
- Supports ECB, CBC, CFB, OFB, CTR and XTS operation modes for all symmetric block ciphers
- Cipher Block Chaining-MAC (CCM) and Galois Counter Mode (GCM)
- Synthetic Initialization Vector (SIV) authenticated encryption
- ChaCha20Poly1305 Authenticated Encryption with Associated Data (AEAD)
- Ascon-Based Lightweight Cryptography (Ascon-AEAD128, Ascon-Hash256, Ascon-XOF128, Ascon-CXOF128)
- RSA public key cryptography (PKCS #1 v1.5 and v2.2)
- Digital Signature Algorithm (DSA)
- Diffie-Hellman key exchange (PKCS #3)
- Password-Based Cryptography Standard (PKCS #5)
- Cryptographic Message Syntax (PKCS #7)
- Elliptic Curve Cryptography (ECC)
- Elliptic Curve Diffie-Hellman (ECDH)
- ECDH over Curve25519 and Curve448 elliptic curves (X25519 and X448)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Deterministic ECDSA signature generation
- EdDSA signature scheme (Ed25519 and Ed448 elliptic curves)
- Supports elliptic curves defined over prime fields (NIST-P and Brainpool)
- ShangMi (SM) cryptographic algorithms SM2, SM3 an SM4
- Multiple precision arithmetic library with optimized assembly code (for ARM and MIPS-based MCUs)
- X.509 certificate, CRL and CSR parsing functions
- X.509 certification and CSR generation
- OCSP client (Online Certificate Status Protocol)
- Parsing and formatting of public/private keys (PKCS #1 and PKCS #8 formats supported)
- Parsing of encrypted private keys (PKCS #1 and PKCS #8 formats supported)
- PBKDF1, PBKDF2, HKDF and Concat KDF key derivation functions
- bcrypt, scrypt, MD5-crypt and SHA-crypt password hashing functions
- Hash_DRBG, HMAC_DRBG, CTR_DRBG and XDRBG pseudorandom number generators
- Flexible memory footprint. Built-time configuration to embed only the necessary features
- Portable architecture (supports little-endian and big-endian architectures)
- Extensive test suite available on request (for commercial licenses)

RFC

- RFC 1319: The MD2 Message-Digest Algorithm
- RFC 1321: The MD5 Message-Digest Algorithm
- RFC 1423: Privacy Enhancement for Internet Electronic Mail Part III: Algorithms, Modes, and Identifiers
- RFC 2104: HMAC: Keyed-Hashing for Message Authentication
- RFC 2144: The CAST-128 Encryption Algorithm
- RFC 2268: A Description of the RC2 Encryption Algorithm
- RFC 2313: PKCS #1: RSA Encryption Version 1.5
- RFC 2315: PKCS #7: Cryptographic Message Syntax Version 1.5
- RFC 2612: The CAST-256 Encryption Algorithm
- RFC 2631: Diffie-Hellman Key Agreement Method
- RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0
- RFC 2985: PKCS #9: Selected Object Classes and Attribute Types Version 2.0
- RFC 2986: PKCS #10: Certification Request Syntax Specification Version 1.7
- RFC 3174: US Secure Hash Algorithm 1 (SHA1)
- RFC 3447: PKCS #1: RSA Cryptography Specifications Version 2.1
- RFC 4269: The SEED Encryption Algorithm
- RFC 4493: The AES-CMAC Algorithm
- RFC 4648: The Base16, Base32, and Base64 Data Encodings
- RFC 5208: PKCS #8: Private-Key Information Syntax Specification Version 1.2
- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- RFC 5297: Synthetic Initialization Vector (SIV) Authenticated Encryption Using the AES
- RFC 5639: ECC Brainpool Standard Curves and Curve Generation
- RFC 5794: A Description of the ARIA Encryption Algorithm
- RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF)
- RFC 5915: Elliptic Curve Private Key Structure
- RFC 5958: Asymmetric Key Packages
- RFC 6090: Fundamental Elliptic Curve Cryptography Algorithms
- RFC 6229: Test Vectors for the Stream Cipher RC4
- RFC 6234: US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)
- RFC 6979: Deterministic Usage of the DSA and ECDSA
- RFC 6818: Updates to the Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- RFC 6960: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP)
- RFC 7468: Textual Encodings of PKIX, PKCS, and CMS Structures
- RFC 7539: ChaCha20 and Poly1305 for IETF Protocols
- RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)
- RFC 7748: Elliptic Curves for Security (Curve25519 and Curve448)
- RFC 7914: The scrypt Password-Based Key Derivation Function
- RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2
- RFC 8018: PKCS #5: Password-Based Cryptography Specification Version 2.1
- RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)
- RFC 8410: Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the PKIX
- RFC 8603: Commercial National Security Algorithm (CNSA) Suite Certificate and CRL Profile
- RFC 8954: Online Certificate Status Protocol (OCSP) Nonce Extension
- RFC 9295: Clarifications for Ed25519, Ed448, X25519, and X448 Algorithm Identifiers
- RFC draft: SM2 Digital Signature Algorithm
- RFC draft: The SM3 Cryptographic Hash Function
- RFC draft: The SM4 Block Cipher Algorithm And Its Modes Of Operations

IEEE

- IEEE Std 1363-2000: Standard Specifications for Public-Key Cryptography

Certicom Research

- SEC 1: Elliptic Curve Cryptography
- SEC 2: Recommended Elliptic Curve Domain Parameters

NIST

- FIPS 46-3: Data Encryption Standard
- FIPS 180-4: Secure Hash Standard (SHS)
- FIPS 186-5: Digital Signature Standard (DSS)
- FIPS 197: Advanced Encryption Standard
- FIPS 198-1: The Keyed-Hash Message Authentication Code (HMAC)
- FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable Output Functions
- SP 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques
- SP 800-38C: The CCM Mode for Authentication and Confidentiality
- SP 800-38D: Galois/Counter Mode (GCM) and GMAC
- SP 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
- SP 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators
- SP 800-232: Ascon-Based Lightweight Cryptography Standards for Constrained Devices

RSA Laboratories

- PKCS #1: RSA Cryptography Standard
- PKCS #3: Diffie-Hellman Key Agreement Standard
- PKCS #5: Password-Based Cryptography Standard
- PKCS #7: Cryptographic Message Syntax Standard
- PKCS #8: Private-Key Information Syntax Standard
- PKCS #9: Selected Attribute Types
- PKCS #10: Certification Request Standard

Supported Processors

- ARM Cortex-M3
- ARM Cortex-M4
- ARM Cortex-M7
- ARM Cortex-M33
- ARM Cortex-M55
- ARM Cortex-M85
- ARM Cortex-R4
- ARM Cortex-A5
- ARM Cortex-A7
- ARM Cortex-A8
- ARM Cortex-A9
- ARM Cortex-A55
- RISC-V
- MIPS M4K
- MIPS microAptiv / M-Class
- Infineon TriCore AURIX
- PowerPC e200
- Coldfire V2
- RX600
- AVR32
- Xtensa LX6

Supported Compilers / Toolchains

Toolchain / IDE	Compiler
Makefile	GCC
AC6 System Workbench for STM32 (SW4STM32)	GCC
Atollic TrueSTUDIO	GCC
Espressif ESP-IDF	GCC
HighTec Toolset for TriCore	GCC
IAR Embedded Workbench	EWARM, EWRX
Infineon DAVE	GCC
Keil MDK-ARM	ARM Compiler v5, ARM Compiler v6 (CLANG)
Microchip Studio (Atmel Studio)	GCC
Microchip MPLAB X	GCC, XC32
Microsoft Visual Studio	MSVC
NXP MCUXpresso	GCC
NXP S32 Design Studio (S32DS)	GCC
Renesas e2Studio	GCC, CC-RX
Segger Embedded Studio	GCC
ST STM32CubeIDE	GCC
Tasking VX-Toolset	VX-Toolset for TriCore