CycloneSSH

CycloneSSH is a SSHv2 library dedicated to 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.

<table>
<tr><td>SFTP</td><td>SCP</td><td>7 - Application</td></tr>
<tr><td>SSH Connection Protocol</td><td>SSH Authentication Protocol</td><td rowspan="2">6 - Presentation</td></tr>
<tr><td colspan="2">SSH Transport Protocol</td></tr>
<tr><td colspan="2">Socket</td><td>5 - Session</td></tr>
</table>

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
- Post-quantum hybrid key exchange (using ML-KEM-768, ML-KEM-1024 and Streamlined NTRU Prime 761 KEMs)
- 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

PQ-Hybrid Key Exchange Algorithms

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

Key Exchange Algorithms

- rsa1024-sha1(w)
- diffie-hellman-group1-sha1(w)
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellman-group18-sha512
- diffie-hellman-group-exchange-sha224@ssh.com
- diffie-hellman-group-exchange-sha384@ssh.com
- ecdh-sha2-nistp256
- ecdh-sha2-nistp521
- curve25519-sha256@libssh.org
- rsa2048-sha256
- diffie-hellman-group14-sha1(w)
- diffie-hellman-group15-sha512
- diffie-hellman-group17-sha512
- diffie-hellman-group-exchange-sha1(w)
- diffie-hellman-group-exchange-sha256
- diffie-hellman-group-exchange-sha512@ssh.com
- ecdh-sha2-nistp384
- curve25519-sha256
- 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-v01@openssh.com(w)
- ssh-rsa-cert-v01@openssh.com(w)
- rsa-sha2-256-cert-v01@openssh.com
- rsa-sha2-512-cert-v01@openssh.com
- ecdsa-sha2-nistp256-cert-v01@openssh.com
- ecdsa-sha2-nistp384-cert-v01@openssh.com
- ecdsa-sha2-nistp521-cert-v01@openssh.com
- ssh-ed25519-cert-v01@openssh.com

Encryption Algorithms

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

MAC Algorithms

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

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

Supported Processors

- ARM Cortex-M3
- ARM Cortex-M4
- ARM Cortex-M7
- ARM Cortex-M33
- ARM Cortex-M85
- ARM Cortex-R4
- ARM Cortex-A5
- ARM Cortex-A7
- ARM Cortex-A8
- ARM Cortex-A9
- Legacy ARM7TDMI / ARM926EJ-S
- 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
- Microsoft Azure RTOS (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

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: The chacha20-poly1305@openssh.com Authenticated Encryption Cipher
- RFC draft: Sending and Handling of Global Requests in Secure Shell (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

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