WOLFSSL · TECHNICAL BENCHMARK

wolfSSL vs MbedTLS

An apples-to-apples benchmark across Intel, ARM (Cortex-A and Cortex-M), and RISC-V targets.

Scope: the full wolfCrypt algorithm suite vs MbedTLS, measured the same way on four platforms (Intel x86_64, a Raspberry Pi 5 (ARMv8-A Cortex-A76), a bare-metal STM32H563 Cortex-M33, and a Microchip PolarFire SoC RISC-V U54), plus the post-quantum and extended-algorithm coverage MbedTLS does not have. wolfSSL v5.9.1, MbedTLS 3.6.6, June 2026.

Method: identical sources built from source on each target. Every wolfSSL build is tuned for maximum speed (architecture assembly plus single-precision math); The MbedTLS benchmarking runs in its fastest stock configuration on each platform. `wolfcrypt/benchmark` vs `programs/test/benchmark`; on the MCU, the same operations are timed with the on-chip DWT cycle counter. Block size 1024 B, about 1 s per algorithm.

wolfSSL and MbedTLS look interchangeable on paper: two small, portable C libraries for embedded TLS and cryptography. When measured, they are not close. wolfCrypt carries hand-written assembly down almost every hot path on x86, ARM, and RISC-V, so the operations that gate secure boot, attestation, and TLS run many times faster than MbedTLS's portable C. wolfSSL also covers far more ground, including a complete post-quantum CNSA 2.0 suite (ML-KEM, ML-DSA, LMS, XMSS) plus SLH-DSA that MbedTLS does not include. The numbers below are measured, not quoted: one identical operation per algorithm, on four real platforms, with every library built from source with optimization for speed and size based on the scenario.

Intel i9-11950H (x86_64)

wolfCrypt built with `--enable-intelasm --enable-sp=yes,asm` (AES-NI, AVX2, SHA extensions, single-precision x86_64 assembly). MbedTLS stock Release (AES-NI plus `MBEDTLS_HAVE_ASM`).

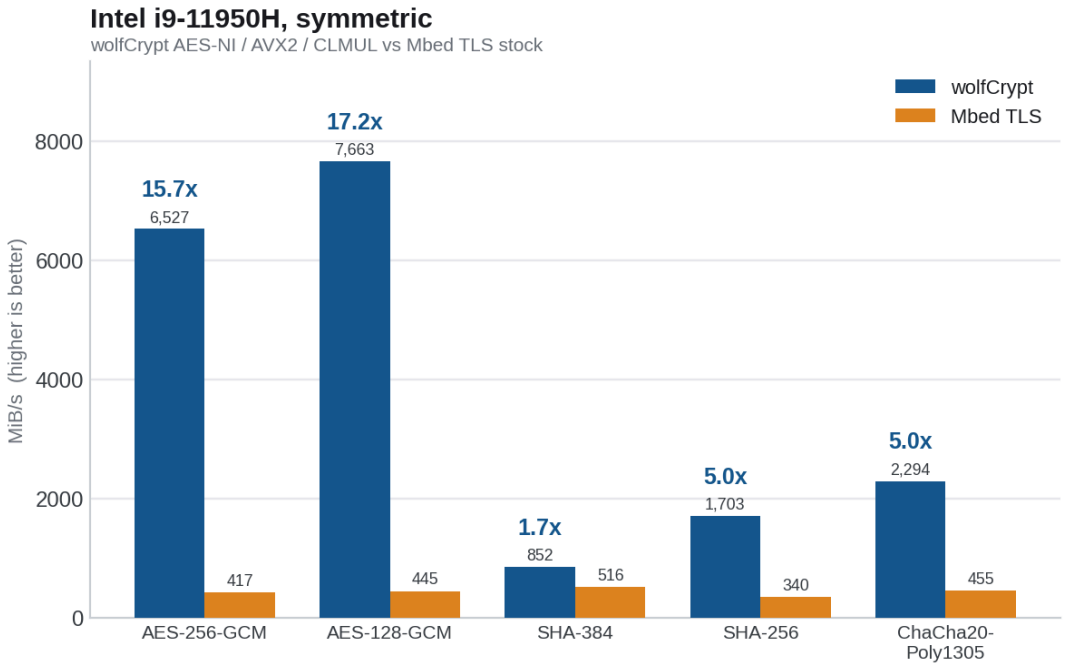

Figure 1. Intel i9-11950H, symmetric throughput (MiB/s).

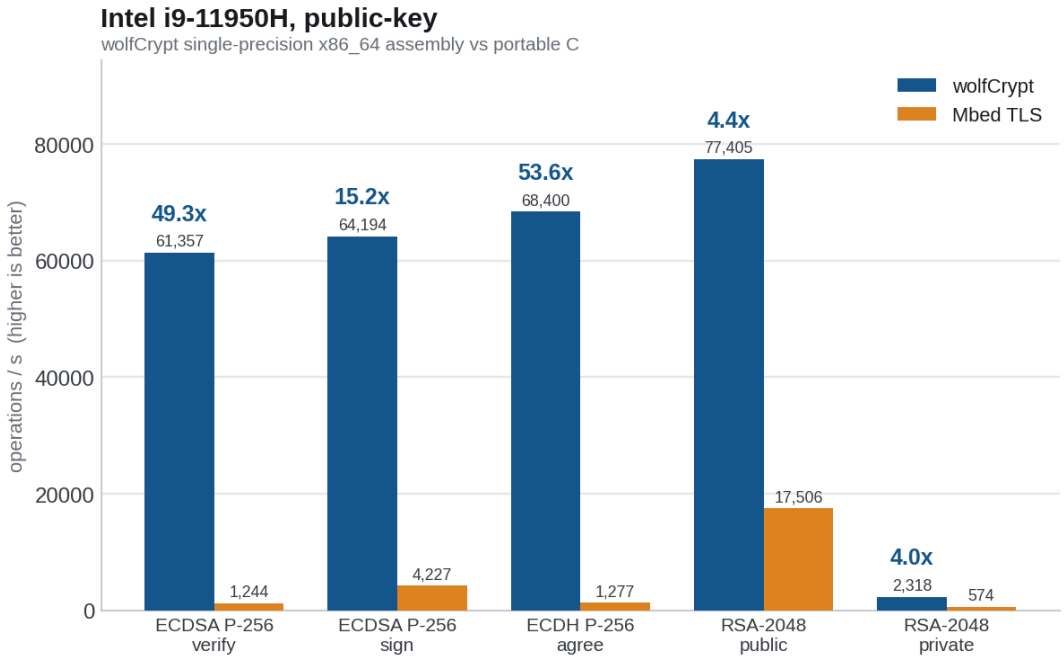

Figure 2. Intel i9-11950H, public-key operations per second.

Operation	wolfCrypt	MbedTLS	wolfSSL Advantage
AES-256-GCM	6527 MiB/s	417 MiB/s	**15.7x**
AES-128-GCM	7663 MiB/s	445 MiB/s	**17.2x**
SHA-384	852 MiB/s	516 MiB/s *	**1.7x**
SHA-256	1703 MiB/s	340 MiB/s	**5.0x**
ChaCha20-Poly1305	2294 MiB/s	455 MiB/s	**5.0x**
ECDSA P-256 sign	64,194/s	4,227/s	**15.2x**
ECDSA P-256 verify	61,357/s	1,244/s	**49.3x**
ECDH P-256 agree	68,400/s	1,277/s	**53.6x**
RSA-2048 public	77,405/s	17,506/s	**4.4x**

* MbedTLS's benchmark only measures SHA-512, not SHA-384; they share the same core and run at the same speed, so it is the fair SHA-384 comparison.

The biggest gaps are in AES-GCM (wolfCrypt's AES-NI plus carry-less-multiply GHASH vs a table-based GCM, 15 to 17x) and the public-key operations (single-precision assembly vs portable C, up to 53x on ECDH).

ARM Raspberry Pi 5 (Cortex-A76 @ 2.4 GHz)

wolfCrypt with `--enable-armasm --enable-sp=yes,asm`. The configure log confirms the ARMv8 crypto extensions are engaged, and AES-256-GCM runs at 1.34 cycles/byte (hardware AES plus PMULL; software AES would be about 25). The same MbedTLS 3.6.6 sources, built on the Pi.

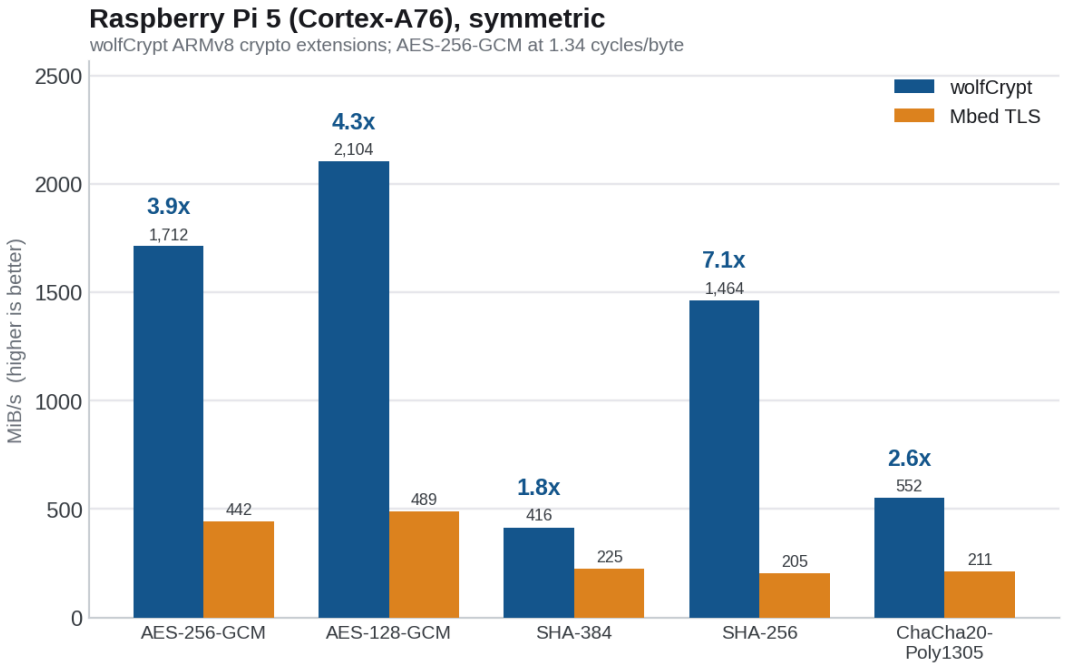

Figure 3. Raspberry Pi 5, symmetric throughput (MiB/s).

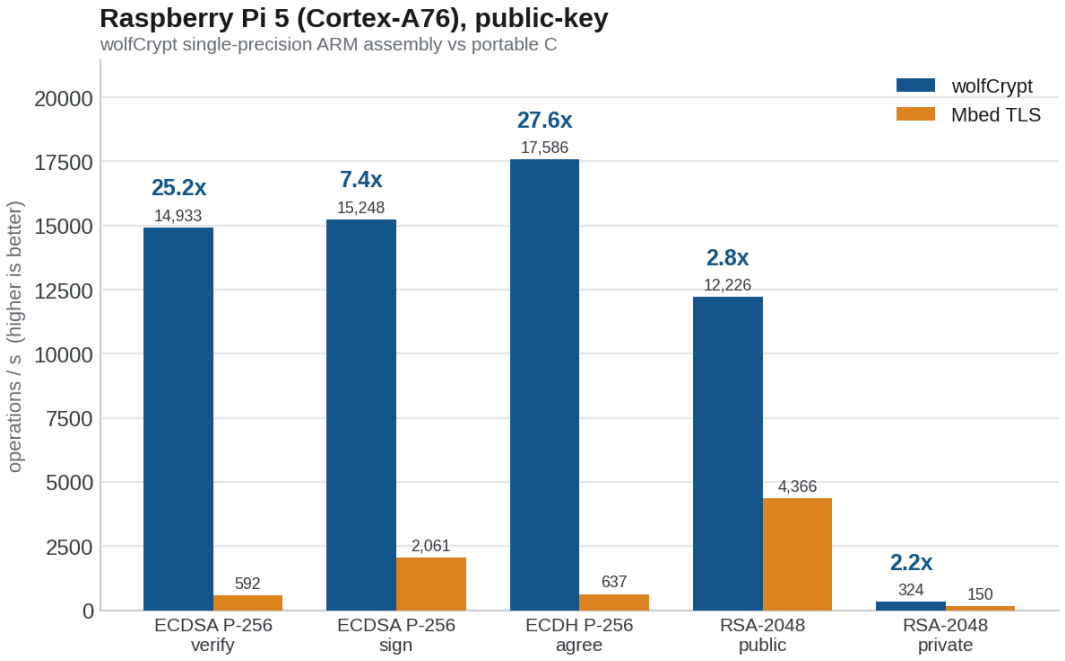

Figure 4. Raspberry Pi 5, public-key operations per second.

Operation	wolfCrypt	MbedTLS	wolfSSL Advantage
AES-256-GCM	1712 MiB/s	442 MiB/s	**3.9x**
AES-128-GCM	2104 MiB/s	489 MiB/s	**4.3x**
SHA-384	416 MiB/s	225 MiB/s *	**1.8x**
SHA-256	1464 MiB/s	205 MiB/s	**7.1x**
ECDSA P-256 verify	14,933/s	592/s	**25.2x**
ECDH P-256 agree	17,586/s	637/s	**27.6x**
RSA-2048 public	12,226/s	4,366/s	**2.8x**

* MbedTLS's benchmark only measures SHA-512, not SHA-384 (MbedTLS supports both; same core and speed). The ARM gap is smaller than Intel's only because MbedTLS also gets ARMv8 AES on the Pi, whereas on x86 its GCM is stuck on a software GHASH.

STM32H563 (Cortex-M33 @ 250 MHz, bare metal)

Both libraries cross-compiled for the Cortex-M33 and timed on-device with the DWT cycle counter. wolfCrypt uses its Thumb2 assembly for the symmetric and hash primitives (AES, SHA-256, ChaCha20, Poly1305) and its SP Cortex-M assembly for the public-key operations (ECC, RSA). No STM32 hardware crypto block is used: the M33 core has no AES or SHA instructions, so this is hand-written software assembly on both sides. The instruction cache is enabled, as on any real STM32H5 firmware; without it the flash runs at 5 wait states and every result is about 3x too slow.

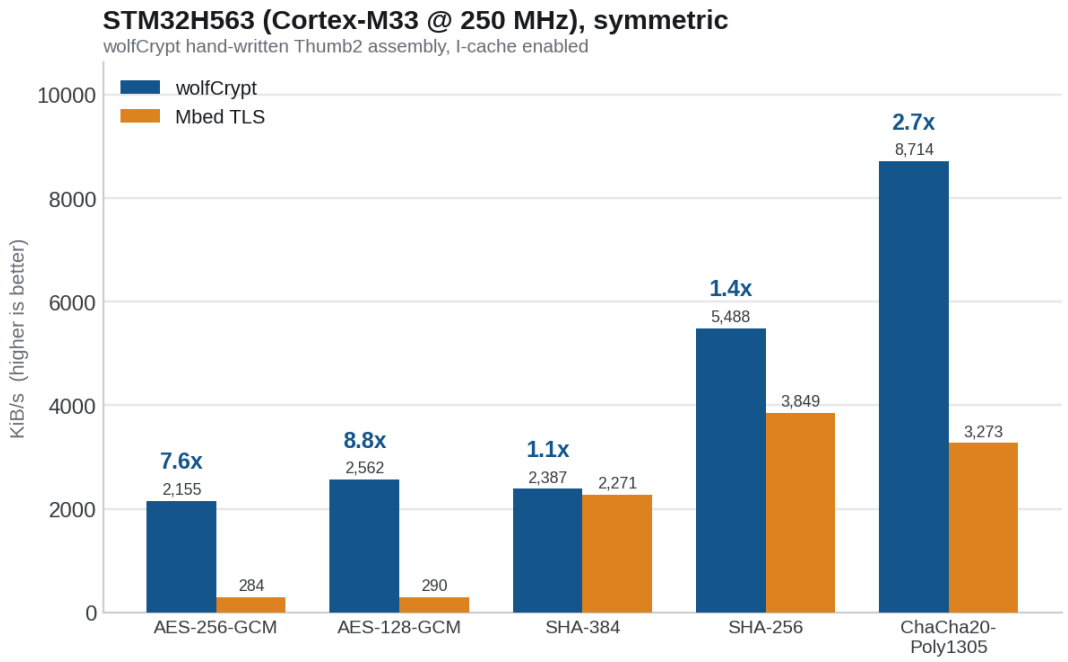

Figure 5. STM32H563, symmetric throughput (KiB/s).

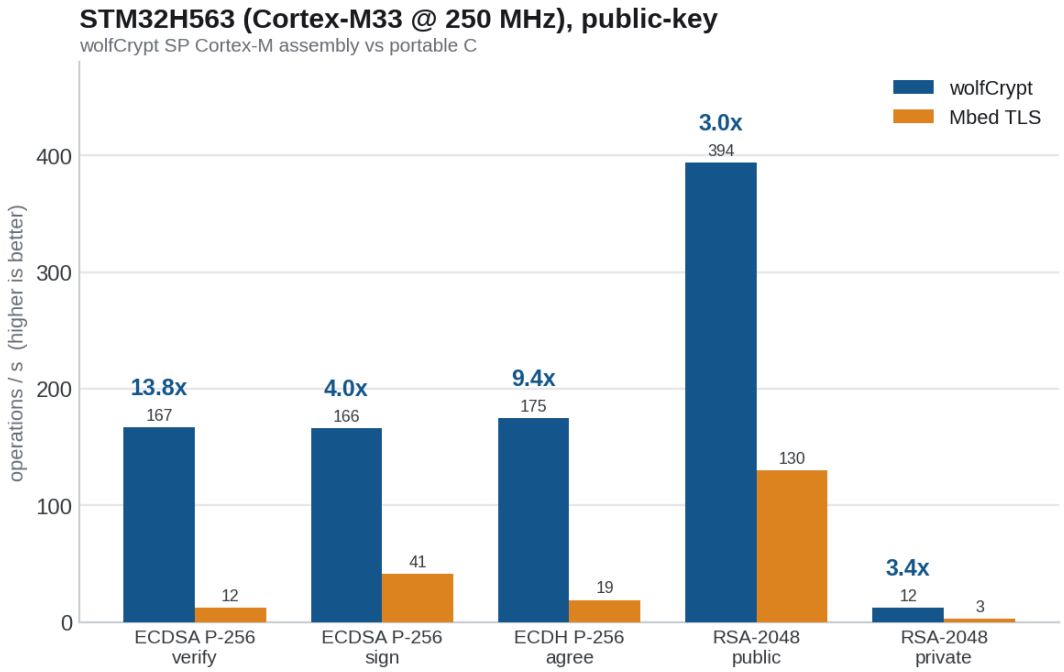

Figure 6. STM32H563, public-key operations per second.

Operation	wolfCrypt	MbedTLS	wolfSSL Advantage
AES-256-GCM	2155 KiB/s	284 KiB/s	**7.6x**
AES-128-GCM	2562 KiB/s	290 KiB/s	**8.8x**
AES-128-CBC	5175 KiB/s	2832 KiB/s	**1.8x**
ChaCha20-Poly1305	8714 KiB/s	3273 KiB/s	**2.7x**
SHA-256	5488 KiB/s	3849 KiB/s	**1.4x**
SHA-384 / SHA-512	2387 KiB/s *	2271 KiB/s	**1.05x**
ECDSA P-256 verify	167/s	12.1/s	**13.8x**
ECDH P-256 agree	175/s	18.6/s	**9.4x**
RSA-2048 public	394/s	130/s	**3.0x**

* SHA-384/512 carries an MCU-specific caveat, and this number is not representative of the assembly's true speed. SHA-512 (which SHA-384 uses) operates on 64-bit words; wolfCrypt's fully-unrolled Thumb2 SHA-512 transform is about 12.6 KB, larger than the STM32H563's 8 KB instruction cache. Run from cached flash it thrashes the cache and drops to 1858 KiB/s, slower than the C transform. Placed in SRAM (a `RAMFUNCTION`, no flash wait states) the same assembly runs at 2387 KiB/s and beats both. This cache ceiling is unique to this small MCU: on the Intel and Raspberry Pi platforms (48 KB / 64 KB L1 caches) the SHA-512 assembly is never cache-bound, which is why their SHA gaps are larger. We report the SRAM number here; on cache-rich or TCM-equipped targets the assembly wins outright with no special placement.

On the MCU wolfCrypt is faster on every algorithm: 1.8x on AES-CBC, 7 to 9x on AES-GCM, 2.7x on ChaCha20-Poly1305 (all hand-written Thumb2 assembly that fits the cache), and overwhelmingly on the public-key operations that gate secure boot and attestation (ECDSA verify 13.8x, ECDH 9.4x) thanks to its SP Cortex-M assembly.

RISC-V Microchip PolarFire SoC (SiFive U54-MC @ 600 MHz)

wolfCrypt built lean and tuned for speed on the U54: a focused algorithm set with --enable-riscv-asm (hand-written RV64 assembly for AES, SHA-2, SHA-3, ChaCha20 and Poly1305) plus single-precision math for RSA and ECC, the whole library is about 712 KB of code. The MPFS250T's four U54 application cores are a plain rv64imafdc design with no AES, SHA, bit-manipulation or vector-crypto instructions, so, as on the STM32, this is hand-written software assembly on both sides, not a hardware crypto block. MbedTLS is built from the same sources in its stock default configuration.

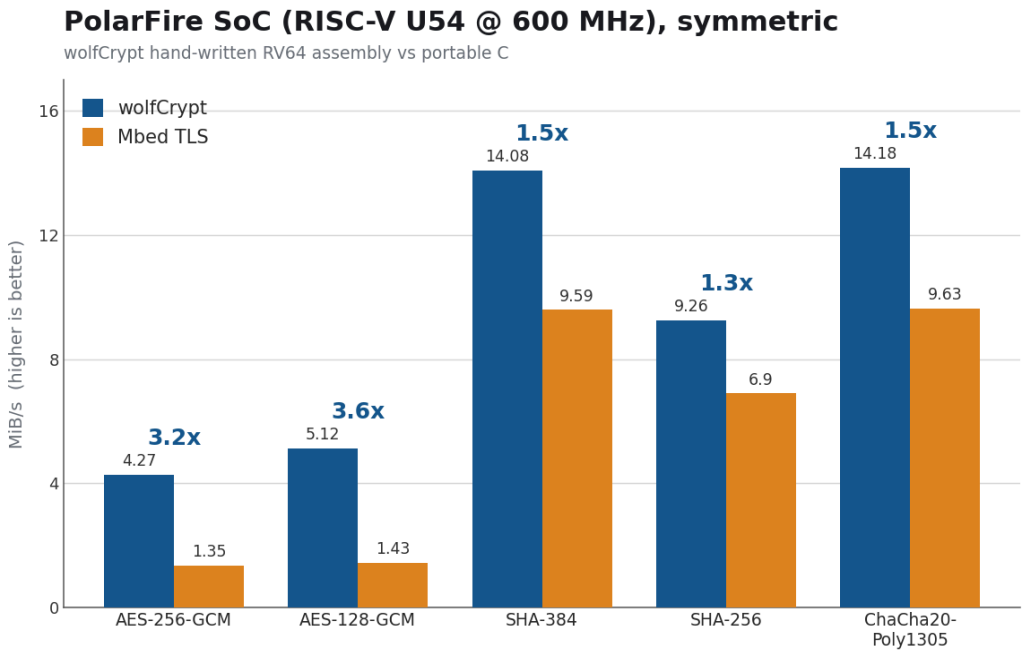

Figure 8. PolarFire SoC U54, symmetric throughput (MiB/s).

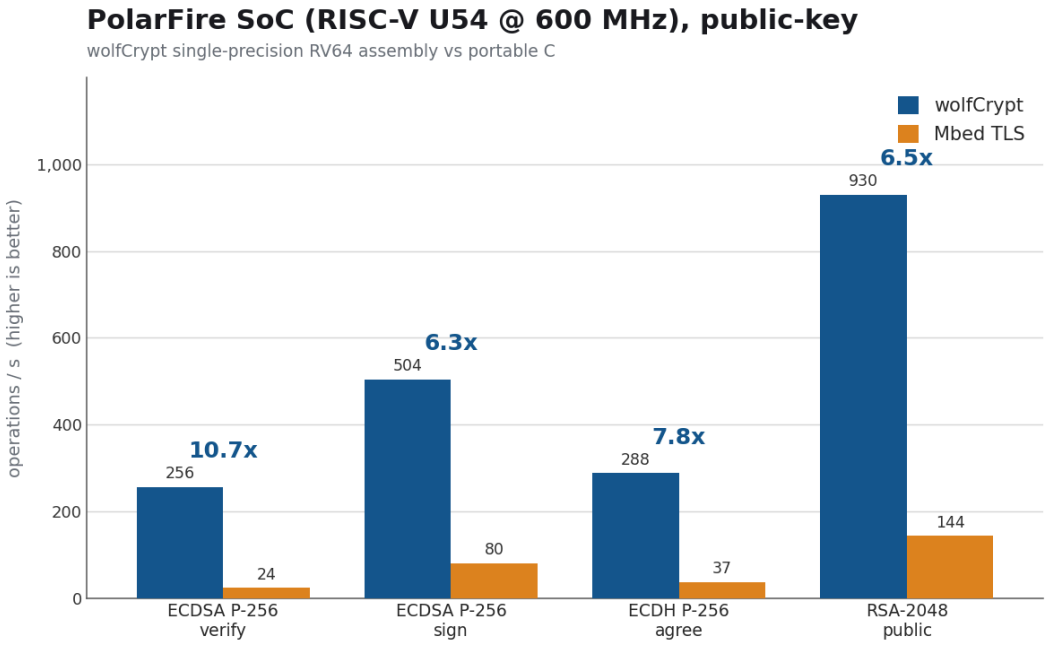

Figure 9. PolarFire SoC U54, public-key operations per second.

Operation	wolfCrypt	MbedTLS	wolfSSL Advantage
AES-256-GCM	4.3 MiB/s	1.3 MiB/s	**3.2x**
AES-128-GCM	5.1 MiB/s	1.4 MiB/s	**3.6x**
ChaCha20-Poly1305	14.2 MiB/s	9.6 MiB/s	**1.5x**
SHA-384	14.1 MiB/s	9.6 MiB/s *	**1.5x**
SHA-256	9.3 MiB/s	6.9 MiB/s	**1.3x**
ECDSA P-256 verify	256/s	24/s	**10.7x**
ECDH P-256 agree	288/s	37/s	**7.8x**
RSA-2048 public	930/s	144/s	**6.5x**

* MbedTLS only measures SHA-512, not SHA-384; they share the same core and run at the same speed, so it is the fair comparison. With no crypto instructions on the U54 both libraries run software-only here.

wolfCrypt is faster on every operation. The public-key gaps are the largest, 6.5x on RSA-2048 public, 7.8x on ECDH and 10.7x on ECDSA verify, because wolfCrypt's single-precision math outruns MbedTLS's portable bignum; the symmetric gaps, up to 3.6x on AES-128-GCM, come from wolfCrypt's RISC-V assembly and a faster GHASH. Enabling the RISC-V assembly on its own lifts the software primitives directly over wolfCrypt's portable C: ChaCha20 +33%, Poly1305 +25%, SHA-256 +24%, and table-free constant-time AES jumps from 0.12 to 6.4 MiB/s.

What wolfSSL supports that MbedTLS does not

The same speed-tuned build also covers a large set of algorithms with no MbedTLS equivalent. Most important today: a complete post-quantum suite.

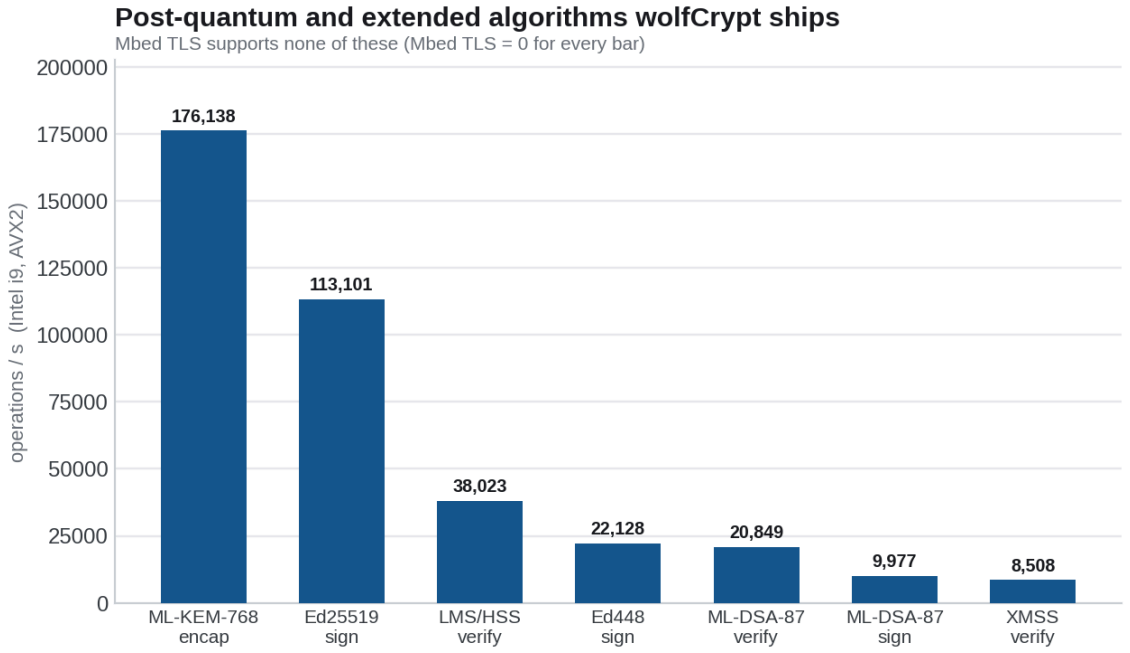

Figure 7. Post-quantum and extended algorithms wolfCrypt ships, ES256-class throughput on Intel i9. MbedTLS supports none of these.

Category	wolfSSL	MbedTLS
Post-quantum signatures	ML-DSA-44/65/87 (FIPS 204; ML-DSA-87 verify 20,849/s), LMS / HSS, XMSS / XMSS-MT, SLH-DSA (FIPS 205)	None
Post-quantum KEM	ML-KEM-512/768/1024 (FIPS 203; ML-KEM-768 encap 176,138/s)	None
EdDSA	Ed25519 (sign 113,101/s), Ed448	None
Identity-based (MIKEY-SAKKE)	ECCSI, SAKKE	None
Extra symmetric / hash / KDF	AES-SIV, AES-GCM-STREAM, AES-OFB, SHAKE128/256, RIPEMD-160, BLAKE2b / 2s, SipHash, scrypt, SRTP / SRTCP KDF	None

Drop-in PSA crypto via wolfPSA

wolfCrypt also implements the Arm PSA Crypto API through wolfPSA, so the speed and coverage above are available without leaving a PSA-based design. On a device already built around Mbed TLS's PSA crypto (for example under TF-M), you can swap in wolfPSA as the provider behind the same `psa_*` calls the application already makes: the PSA architecture stays as it is, whilst still getting wolfCrypt capabilities and performance. You keep the PSA-based design and gain the FIPS

140-3 path, the post-quantum suite, and wolfCrypt's assembly speed in one stack, dropping straight into TF-M or PSA applications.

In practice this means the gaps in the charts above are reachable without a rewrite. The application keeps calling the standard PSA API, wolfPSA routes each operation to wolfCrypt, and the faster AES-GCM, single-precision public-key math, and post-quantum algorithms come with it.

Beyond speed: wolfSSL vs MbedTLS

Speed and algorithm coverage are only part of the story. wolfSSL is a commercially supported, certified library maintained by its original authors, with broad platform reach and a deep feature set. MbedTLS is a community project forked from PolarSSL / XySSL, without the same certifications, breadth, or support. The full comparison:

Technology

Capability	wolfSSL	MbedTLS
Ownership	wolfSSL Inc. (single owner)	Multiple owners; forked from XySSL / PolarSSL
Development team	Original developers still on the project	Not maintained by the original developers
Portability	Portable out of the box: Windows (Win32/64), Linux, macOS, Solaris, *BSD (FreeBSD, NetBSD, OpenBSD), QNX, embedded Linux (Yocto, Buildroot, OpenWRT), iOS, Android, and most RTOSes (FreeRTOS, SafeRTOS, Zephyr, ThreadX / Azure RTOS, VxWorks, Micrium uC/OS-II/III, Nucleus, Green Hills INTEGRITY, SEGGER embOS, Keil RTX / CMSIS-RTOS, TI-RTOS, NXP MQX, Apache Mynewt, NuttX, RIOT, ChibiOS, DDC-I DEOS, TRON / ITRON / uITRON, uT-Kernel), plus bare metal / no-OS, Linux kernel module, Arduino, ARM Mbed OS, Contiki-NG, TenAsys INtime, EBSnet, Nintendo / game consoles (devkitPro), Haiku, HP-UX, NonStop, MontaVista, TinyOS, and AIX	Fewer targets out of the box: Win32/64, Linux, macOS, *BSD, OpenWRT, iOS, Android, SEGGER embOS, Xbox
Standards	SSL 3.0 to TLS 1.3; DTLS 1.0 / 1.2 / 1.3	TLS 1.2 / 1.3; DTLS 1.2
Assembly and CPU acceleration	ARM AArch32 / AArch64 and Cortex-M Thumb-2 assembly, NEON, and ARMv8-A crypto	AES-NI/CLMUL + Armv8 AES/SHA + bignum asm

Capability	wolfSSL	MbedTLS
	extensions (AES, SHA-1 / SHA-256, SHA-512, SHA-3, SM3 / SM4) plus ARM ML-KEM post-quantum assembly; Intel and AMD x86-64 AES-NI, VAES, PCLMULQDQ (GHASH), AVX1 / AVX2, BMI2 / ADX, RDRAND / RDSEED and Intel QuickAssist (QAT), with AVX2 assembly for ChaCha20 / Poly1305 / X25519; RISC-V RV64 with scalar (Zk) and vector (Zvk) crypto extensions; PowerPC assembly; and single-precision (SP) math in C and assembly for RSA / ECC / DH	
Hardware crypto and secure elements	STMicroelectronics STM32 (F1/F2/F4/F7, L1/L4/L5/U5, H5/H7, WB, WL, MP13/MP25) PKA / CRYP / HASH and STSAFE-A110; NXP CAAM (i.MX 6/7/8, RT1170), DCP, LTC, Kinetis mmCAU, Coldfire SEC, CASPER and EdgeLock SE050; Microchip PIC32 MX / MZ, ATECC508A / 608A / 608B and TA100; Espressif ESP32 / S2 / S3 / C3 / C6 (AES / SHA / RSA); Renesas RX65N / RX72N (TSIP), RA6M4 / RZ (SCE) and Synergy; Silicon Labs Series 2 / EFR32 (SE, CRYPTO, TRNG); Nordic nRF52840 and nRF51 with ARM CryptoCell-310; Analog Devices / Maxim MAX3266x, MAXQ1065 / MAXQ1080; Cypress / Infineon PSoC 6; Texas Instruments TM4C (Tiva); Xilinx / AMD Zynq UltraScale+ and Versal; Cavium / Marvell Nitrox (III / V) and OcteonTX; Tropic Square TROPIC01; TPM 2.0 (wolfTPM); IoT-SAFE; PKCS#11 HSMs; Linux /dev/crypto, AF_ALG and KCAPI; and the Arm PSA Crypto API for TF-M, NXP SECO (i.MX 8 Security Controller), Raspberry Pi Pico (RP2040), NVIDIA CUDA (GPU AES), AUTOSAR crypto driver (CSM / CryIf), and Renesas RSIP	STMicroelectronics STM32 (CRYP / HASH / PKA); NXP (CAAM, ELS / SSS); Espressif ESP32 / S2 / S3 / C3 / C6 (AES / SHA / RSA); Nordic nRF (Arm CryptoCell CC310 / CC312); Silicon Labs EFR32 Series 1 / 2 (SE / CRYPTO); Renesas RA (SCE); Cypress / Infineon PSoC 6; Microchip ATECC608A and NXP SE050 (PSA secure-element drivers); Arm PSA Crypto API and TF-M
Command-line utility (wolfCLU)	**Yes**	**No**

Capability	wolfSSL	MbedTLS
Certifications	FIPS 140-3 (#4718, #5041); DO-178C DAL-A; ISO 26262 ASIL D	No
Certificate revocation	CRL, OCSP, OCSP stapling	CRL
Crypto abstraction layer	Yes	No
TLS inspection (sniffer)	Yes	No
Compression	zlib	No
OpenSSL compatibility layer	Yes, 1,600+ functions, actively maintained	Limited, out of date
Post-quantum	ML-KEM (FIPS 203), ML-DSA (FIPS 204), SLH-DSA (FIPS 205), LMS/HSS, XMSS/XMSS-MT, Falcon FN-DSA	No
Supported open-source projects	OpenSSH, stunnel, wpa_supplicant, lighttpd, cURL, OpenVPN, NGINX, mongoose, and 900 others	Limited
Quality assurance	API tests, peer review, static analysis, many compilers and platforms, hardware-accelerated testing, fuzzing (internal fuzzer, AFL, libFuzzer, TLS Fuzzer), big / little endian, known user configs, external validation	Compilation tests, code coverage, regressions, test vectors, interop, build configs, memory checks, fuzzing, static analysis
AI	Internally developed Skoll and Fenrir advanced AI static-analysis scanning tools	None

Support, documentation, and licensing

Capability	wolfSSL	MbedTLS
Documentation	Complete: full manual, API reference, build instructions, extensions reference, tutorials, examples, benchmarking	Partial: build instructions, API reference, source
Vulnerability response	Fixes typically available within days	Fixes can take months, or not at all
License	Dual GPLv3 / commercial	Dual GPLv2 / Apache 2.0
Royalty-free	Yes	Yes
Commercial support (up to 24/7)	Yes, from the original developers: email, phone, and forums; tiers from presale to 24/7; response within 3 business days	Community forums only

Features

Capability	wolfSSL	MbedTLS
Randomness / entropy	wolfRAND, NIST DRBG (SHA-256)	DRBG (SHA-1 / SHA-256)
Extra hash / cipher functions	AES-SIV / CFB / OFB, AES-GCM-STREAM, SHAKE128/256, BLAKE2b / 2s, RIPEMD-160, SipHash, ECIES, SM2 / SM3 / SM4	None of these
Public-key acceleration	Single-precision math, ECC fixed-point cache	ECC NIST modulo-p speedups
TLS extensions	SNI, Max Fragment, ALPN, Trusted CA Indication, Truncated HMAC, Secure Renegotiation, Session Ticket, Extended Master Secret, Encrypt-then-MAC, Quantum-Safe Hybrid, OCSP stapling (v1/v2), Supported Groups, Post-Handshake Auth, 0-RTT Early Data, Cookie; Standards now includes QUIC	Max Fragment, Encrypt-then-MAC

Bottom line

Built the same way on the same hardware, wolfCrypt is faster than MbedTLS across the board, by 15 to 53x on the public-key operations that gate secure boot and TLS on x86, by 3 to 28x on a Raspberry Pi 5, by up to 14x on a bare-metal STM32H563, and by 6 to 11x on a PolarFire RISC-V with the largest gaps wherever wolfCrypt has hand-written assembly available. wolfCrypt also ships a complete post-quantum suite (ML-KEM, ML-DSA, SLH-DSA, LMS, XMSS), EdDSA, and a wide set of extended algorithms that MbedTLS does not have at all. On top of the speed and coverage, it is FIPS 140-3 validated (certificates #4718 and #5041), DO-178 DAL-A, ISO 26262 ASIL D capable, MISRA C checked, supports the Arm PSA Crypto API for TF-M, and is backed by 24/7 commercial support from its original developers.

Versions and platforms

Libraries: wolfSSL v5.9.1 (dual GPLv3 / commercial), MbedTLS 3.6.6. Tested June 2026.

Builds: wolfSSL tuned for maximum speed on each target. Intel `--enable-intelasm --enable-sp=yes,asm`; Raspberry Pi `--enable-armasm --enable-sp=yes,asm`; STM32 Thumb2 plus SP Cortex-M assembly. MbedTLS in its fastest stock configuration (`MBEDTLS_HAVE_ASM`). Identical sources built from source on every target.

Platforms: Intel i9-11950H, x86_64, GCC 12.2. Raspberry Pi 5, Cortex-A76 at 2.4 GHz, Debian 12, GCC 12.2. STM32H563 (NUCLEO-H563ZI), Cortex-M33 at 250 MHz, bare metal, arm-none-eabi-gcc 13.2, instruction cache enabled. PolarFire SoC Video Kit (MPFS250T), 4x SiFive U54-MC at 600 MHz, Yocto Linux 6.12, GCC 13.3.

Measurement: `wolfcrypt/benchmark` vs `programs/test/benchmark` on the desktop and server targets; on the MCU, the same operations timed with the on-chip DWT cycle counter. Block size 1024 B, about 1 s per algorithm. Symmetric in MiB/s or KiB/s, public-key in operations per second.