



wolfBoot

resilient, quantum-resistant secure boot for all architectures

Daniele Lacamera - wolfSSL

www.wolfssl.com

© Copyright 2025 wolfSSL - CC-BY-SA.





Open Source
Internet Security

LIGHTWEIGHT. PORTABLE. C-BASED.

- Up to **TLS 1.3** and **DTLS 1.3**
- **20-100 kB footprint**
- **1-36 kB RAM** per session
- Up to **20X Smaller** than OpenSSL
- Long list of supported operating systems
- Certified FIPS 140-3, DO-178 Support, MISRA-C
- Best-tested crypto
- 24x7 Support
- Dual-licensed
- **Secure boot, MQTT, SSH, TPM 2.0, JSSE, commercial support for curl**

SYSGO
EMBEDDING INNOVATIONS

RENESAS

Mentor
Graphics

arm

TEXAS
INSTRUMENTS

Green Hills
SOFTWARE

0x5

MICROCHIP

NXP

ST
life.augmented

XILINX

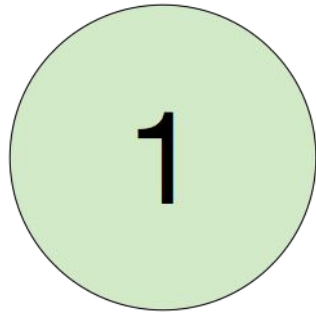
intel

DDC-I

HEX-Five

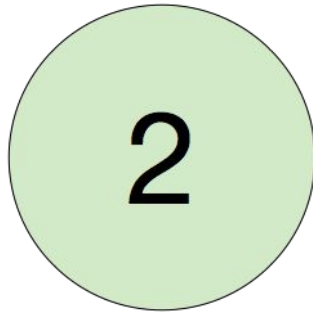
Main Areas of Focus

Data at Rest



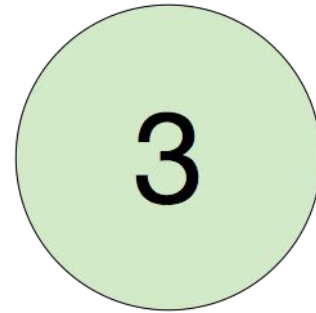
- Secured with **Cryptography**

Data in Transit



- Secured with **SSL/TLS, SSH**
- Possible Transfer Mediums:
TCP/UDP/Bluetooth/Serial/etc

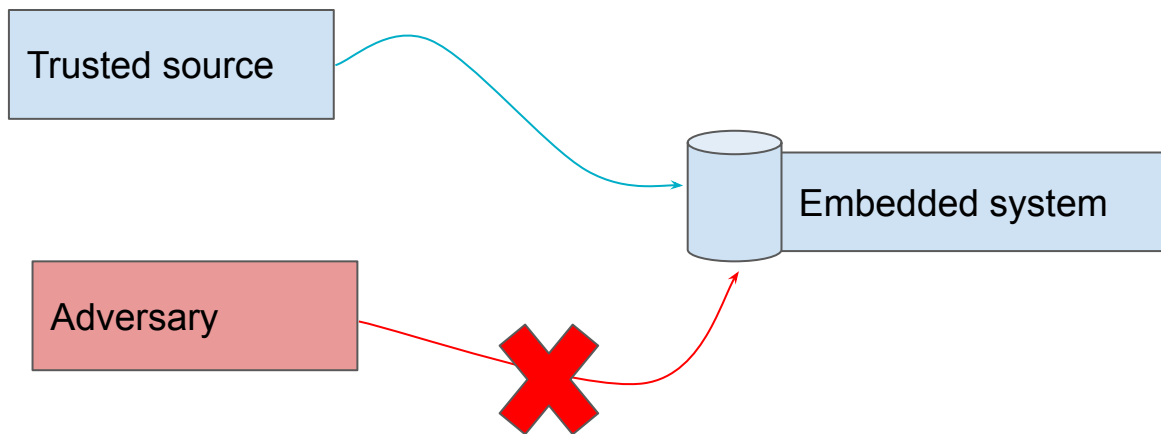
Firmware Updates



- Secured with **SSL/TLS, crypto, MQTT**
- Prevent malicious firmware flashing and updates

Secure boot

- **Secure boot prevents running unauthorized software onboard**
 - Cryptographic signature attached to all authorized software
 - A secure bootloader will refuse to run non-authentic code



wolfBoot

- **Est. 2018**

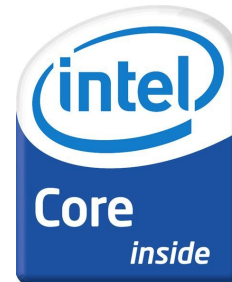
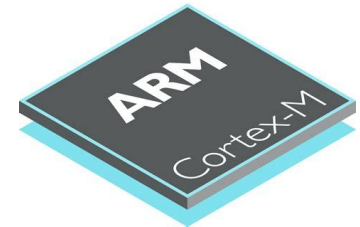
- Project started on Cortex-M
- Based on RFC 9019 (back then “suit” draft) for IoT
- Presented at FOSDEM 2020 in the IoT devroom
- ECC, RSA, Ed25519, Ed448, ML-DSA, LMS, XMSS

- **Portability**

- Several architectures supported (MCUs and CPUs)
- 35+ targets with examples and documentation
- Can run as library anywhere

- **Safety**

- No dynamic allocations
- No IRQ handling
- Execution flows predictable at compile time
- Protected against glitch attacks and fault injections



wolfBoot

- **Unique features**

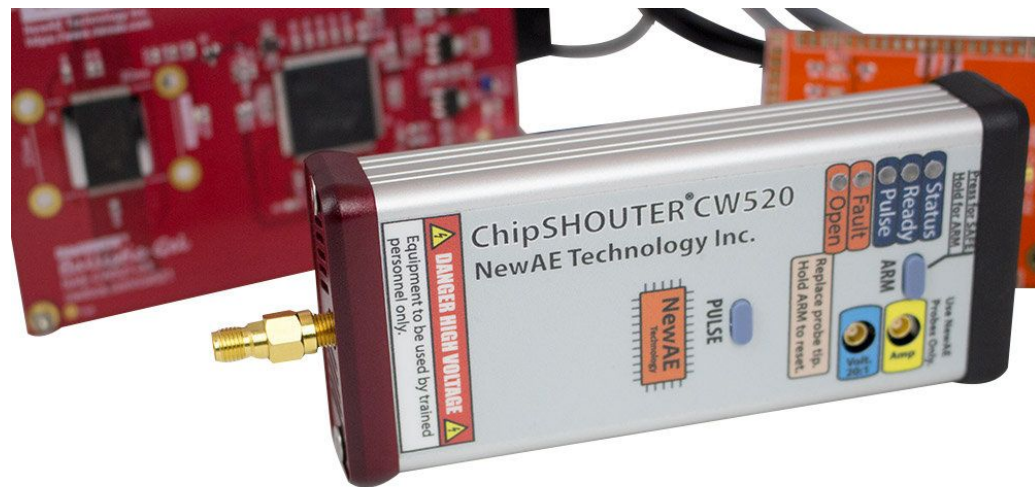
- Delta updates with emergency fallback
- End-to-end encryption with Chacha or AES
- TPM integration (offload crypto, store secrets, measured boot)
- Architecture (and compiler) specific EMFI mitigations

- **Certified security**

- FIPS 140-3 cryptography with wolfCrypt
- DO-178C up to DAL-A

- **Flexibility**

- Portable key tools (Linux/Mac/Windows)
- Multiple partitions/multiple keys
- Support for offloading signing to HSM
- Customized header fields



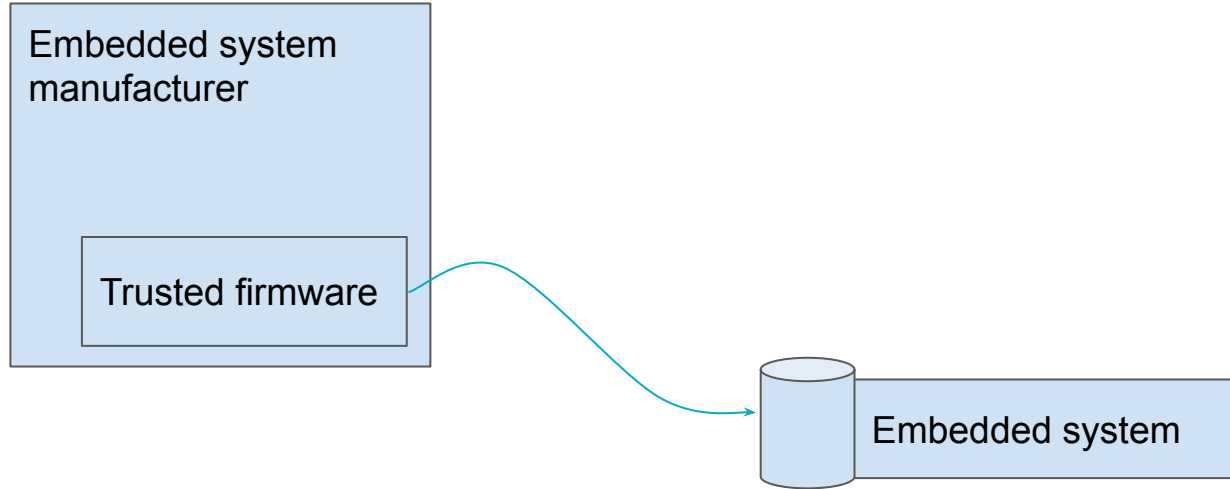
Specifications

- **wolfBoot follows RFC9019**
 - Small parsers
 - Manifest header
 - Public-key based authentication
 - Trust anchor
 - Hash based integrity verification
 - Update transfers are managed by the application

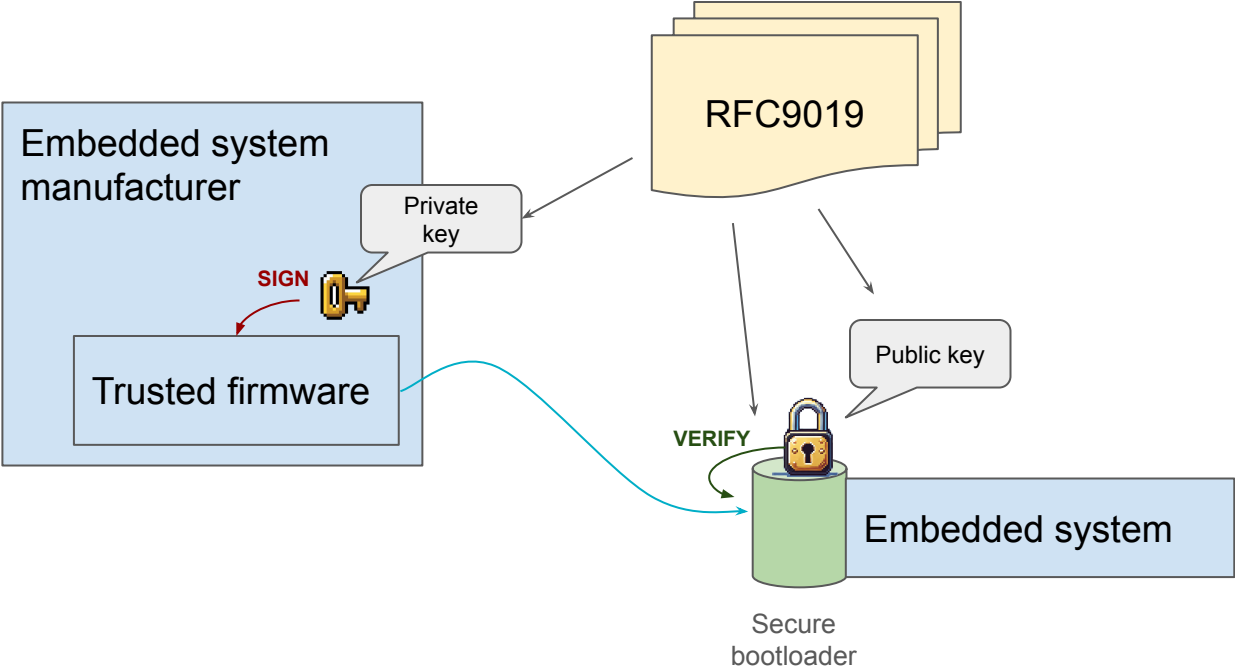
Specifications

- **Keypair is generated once**
 - Private key stays on the server and is never shared
 - Public key is stored on the target and accessible by the bootloader
- **Security depends only on the unique private key(s) used to sign the updates**
 - Those are never distributed, and stored in the cloud/back-end/HSM etc.
- **Sign tool attaches a manifest header to the image**
 - The manifest is used to validate the integrity and the authenticity of the image
 - Firmware version is also part of the verified payload and cannot be altered without the key
- **wolfBoot relies on a trust anchor to embed the public keys used to verify the signature**

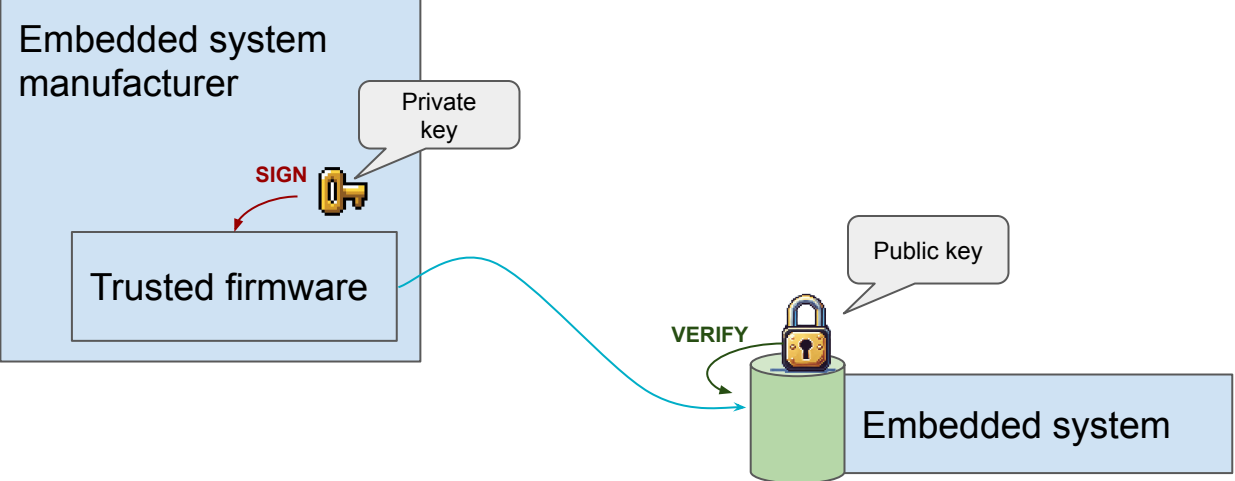
Secure boot: trusted firmware



Secure boot: RFC9019 (2021)



Secure boot: signed firmware



TA store requirements

Secure bootloader requirement for TA store:

“A trust anchor store must resist modification against unauthorized insertion, deletion, and modification.”

- RFC9019

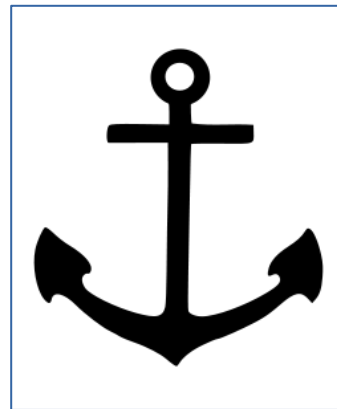
Default: keystore is included in wolfBoot binary.

Relies on FLASH write protection provided by the manufacturer:

- Sufficiently secure for most use cases
- Does not prevent all HW attacks

Suggested alternatives:

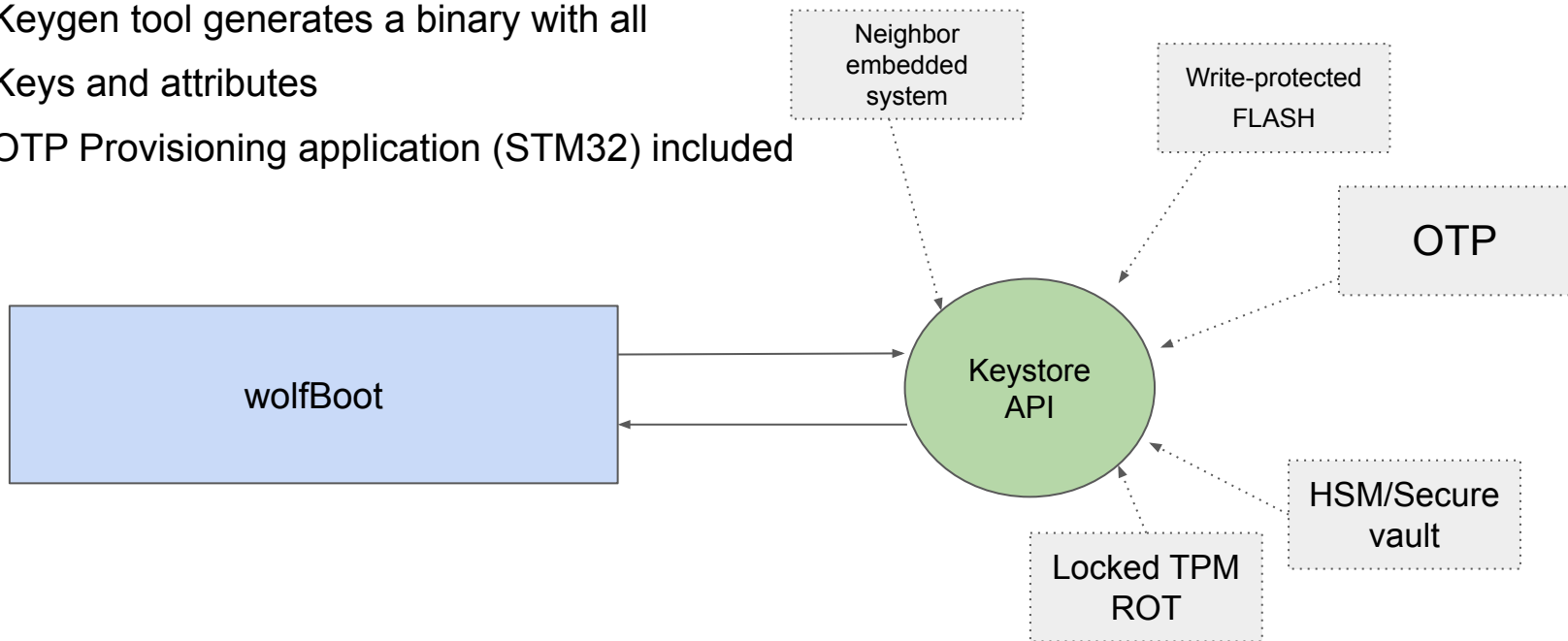
- Trust anchor store implemented with [secure element or TPM](#)
- [OTP](#) memory (designed for this purpose)
- OOB Trusted Provisioning Authority ([TPA](#))
- (Manufacturer may install trust anchors [specific for the device](#))



Management of the keystore

The 'keystore' contains the Trust Anchor

- Can be placed anywhere
- If not directly mapped in memory, public keys can be retrieved with a simple API
- Keygen tool generates a binary with all Keys and attributes
- OTP Provisioning application (STM32) included



Types of architectures supported

- **Microcontroller devices — typical use cases**
 - Boot/update/swap on the same FLASH
 - Boot on XIP, UPDATE and SWAP on a secondary flash (e.g. SPI)
 - When hw-assisted swap is supported: BOOT and UPDATE on XIP flash, duplicated wolfBoot

- **CPU-based systems — some of the most common features**
 - Separate stages to initialize systems if required
 - A/B partition election at boot, usually bootloader is on a different NVM
 - Always load kernel to RAM
 - Multiple partitions/Interaction with other boot stages

Support for TPM 2.0

How **wolfBoot** integrates with **wolfTPM**:

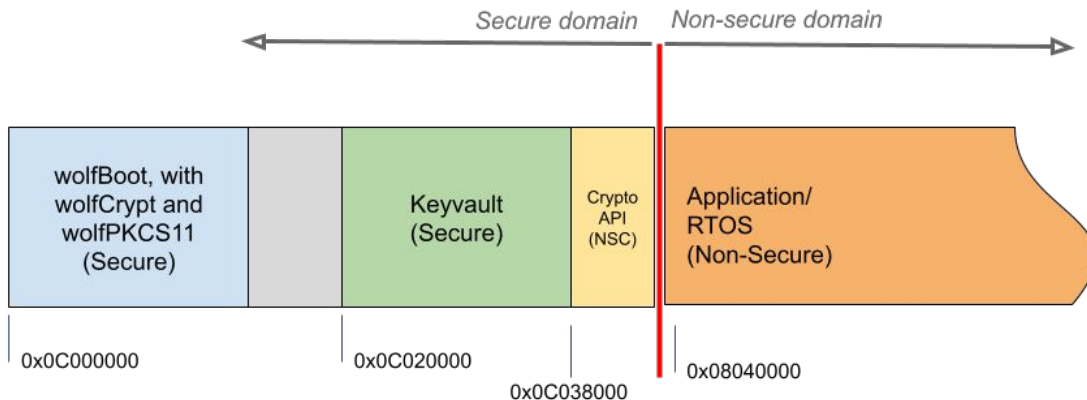
- **Root of Trust (ROT)**
 - Hash of public key to NV (optionally locked) with auth
- **Cryptographic offloading**
 - ECDSA/RSA verification in hardware
- **Measured Boot (PCR Extend)**
- **Sealing a secret** e.g. for encrypting or unlocking a disk
 - Seal based on externally signed policy
- **Platform authentication lockout**
- **Parameter Encryption support (AES-CFB)**

Support for secure vault in TrustZone-M

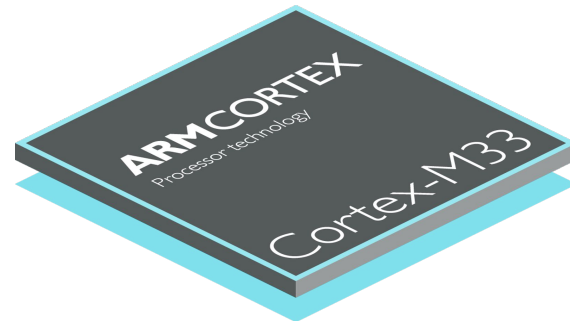
wolfBoot manages the secure domain in TrustZone-M (ARMv8-M architecture)

- Separate domains, stage application/RTOS in non-secure world
- Provides non-secure callable (NSC) API to access crypto functions
 - Applications may use standard API (PKCS11) to access cryptography
 - TLS sockets and other applications in non-secure domain can use pre-provisioned keys and certificates in the Keyvault, generate key pairs, etc.
 - Private keys and secrets are never accessible from non-secure domain

Fully replaces TF-M (removing a lot of unnecessary bloating...)



arm
TRUSTZONE



Port on Intel Tiger Lake

wolfBoot Boot Flow on 11th Gen Intel Core i7

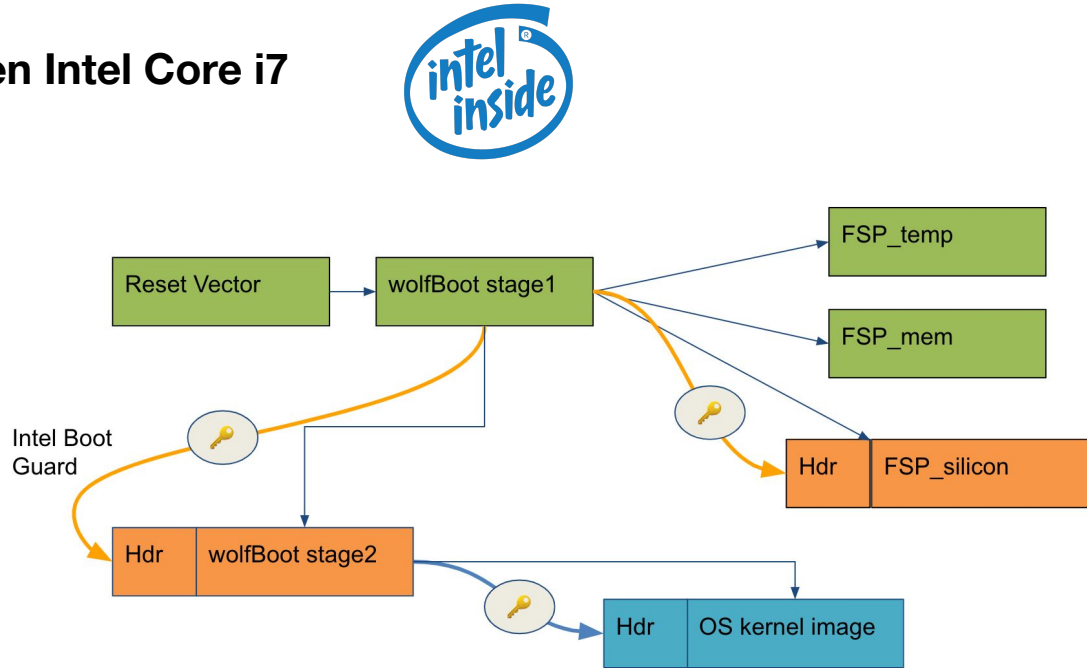
- Execution starts from reset vector
- **Green image** is signed with Intel tools and verified by Intel Boot Guard


wolfBoot Stage 1:

- Loads and executes FSP
- Memory init in FSP
- **Verifies authenticity of FSP_S**
- **Verifies wolfBoot stage 2**

wolfBoot Stage 2:

- Selects image to boot
- **Verifies kernel image**
- Loads and stages kernel



 Verified by Intel Boot Guard

 Verified by stage1

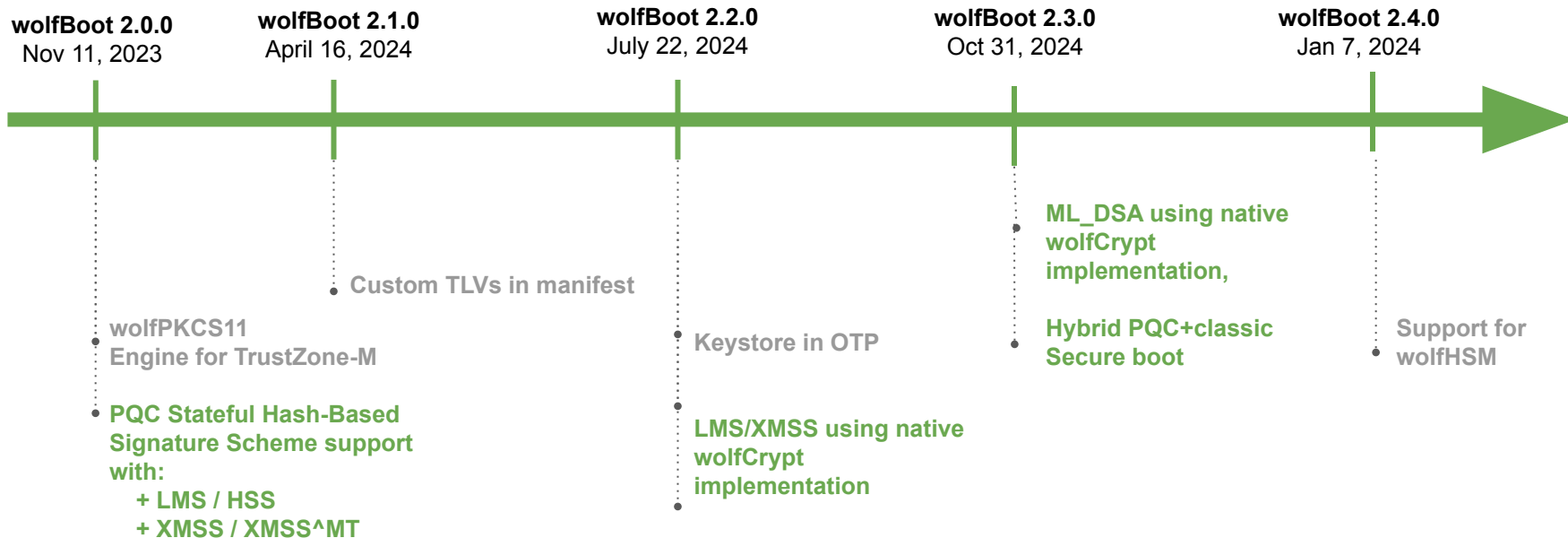
 Verified by stage2

Post-quantum secure boot

Securing the boot process with quantum-resistant cryptography



wolfBoot and Post-Quantum Cryptography



Stateful Hash-Based Signatures

- 2016 NIST Post-Quantum Standardization for Public-Key algorithms required signature algorithm submissions to be “**stateless**”
 - “**stateful**” signature algorithms did not meet the API requirements, standardization was **separate from 2016 competition**, coordinated along with **IETF**
- **Stateful Hash-Based Signatures:**
 - Not vulnerable to quantum computers
 - Well studied and very old
 - Better performance than stateless algorithms for sign/verify, but **very slow keygen**
 - **Require careful state management; misuse is easy and catastrophic**
 - Appropriate for applications where **private key resides in an HSM and private key operations are offline (Ex: firmware signing)**
 - Gave a head start to digital signature scheme PQC migration

Stateful Hash-Based Signatures

- **IETF** standardized both of the following Stateful Hash-Based Signature algorithms:
 - **XMSS** ([RFC 8391](#)) - eXtended Merkle Signature Scheme
 - **LMS** ([RFC 8554](#)) - Leighton-Micali Hash-Based Signatures

Internet Research Task Force (IRTF) Request for Comments: 8391 Category: Informational ISSN: 2070-1721	INFORMATIONAL Errata Exist A. Huelsing TU Eindhoven D. Butin TU Darmstadt S. Gazdag genua GmbH J. Rijneveld Radboud University A. Mohaisen University of Central Florida May 2018
---	---

XMSS: eXtended Merkle Signature Scheme

Abstract

This note describes the eXtended Merkle Signature Scheme (XMSS), a hash-based digital signature system that is based on existing descriptions in scientific literature. This note specifies Winternitz One-Time Signature Plus (WOTS+), a one-time signature scheme; XMSS, a single-tree scheme; and XMSS^{MT}, a multi-tree variant of XMSS. Both XMSS and XMSS^{MT} use WOTS+ as a main building block. XMSS provides cryptographic digital signatures without relying on the conjectured hardness of mathematical problems. Instead, it is proven that it only relies on the properties of cryptographic hash functions. XMSS provides strong security guarantees and is even secure when the collision resistance of the underlying hash function is broken. It is suitable for compact implementations, is relatively simple to implement, and naturally resists side-channel attacks. Unlike most other signature systems, hash-based signatures can so far withstand known attacks using quantum computers.

Internet Research Task Force (IRTF) Request for Comments: 8554 Category: Informational ISSN: 2070-1721	INFORMATIONAL Errata Exist D. McGrew M. Curcio S. Fluhrer Cisco Systems April 2019
---	--

Leighton-Micali Hash-Based Signatures

Abstract

This note describes a digital-signature system based on cryptographic hash functions, following the seminal work in this area of Lamport, Diffie, Winternitz, and Merkle, as adapted by Leighton and Micali in 1995. It specifies a one-time signature scheme and a general signature scheme. These systems provide asymmetric authentication without using large integer mathematics and can achieve a high security level. They are suitable for compact implementations, are relatively simple to implement, and are naturally resistant to side-channel attacks. Unlike many other signature systems, hash-based signatures would still be secure even if it proves feasible for an attacker to build a quantum computer.

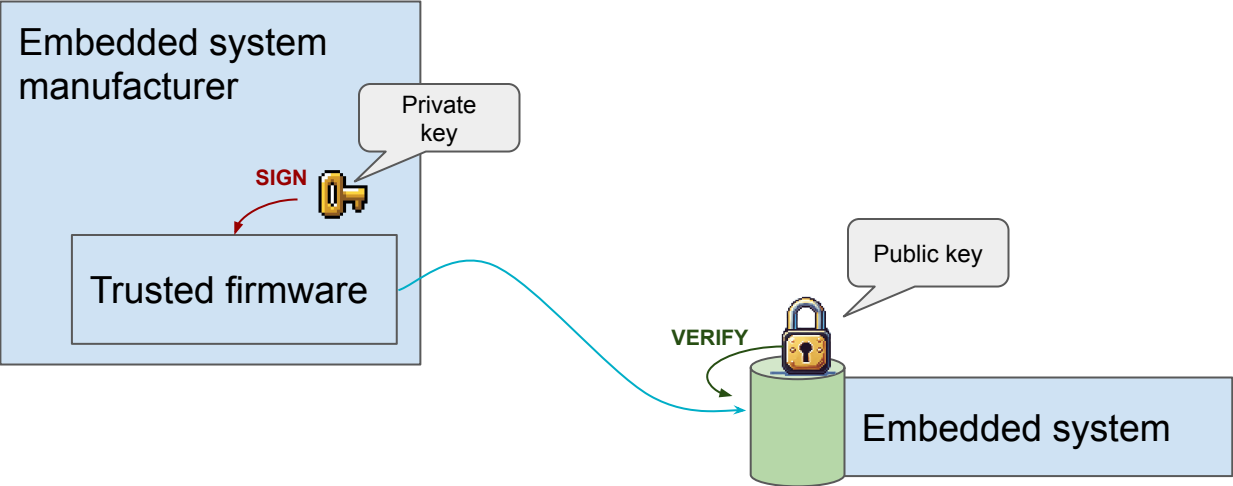
This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF. This has been reviewed by many researchers, both in the research group and outside of it. The Acknowledgements section lists many of them.

Status of This Memo

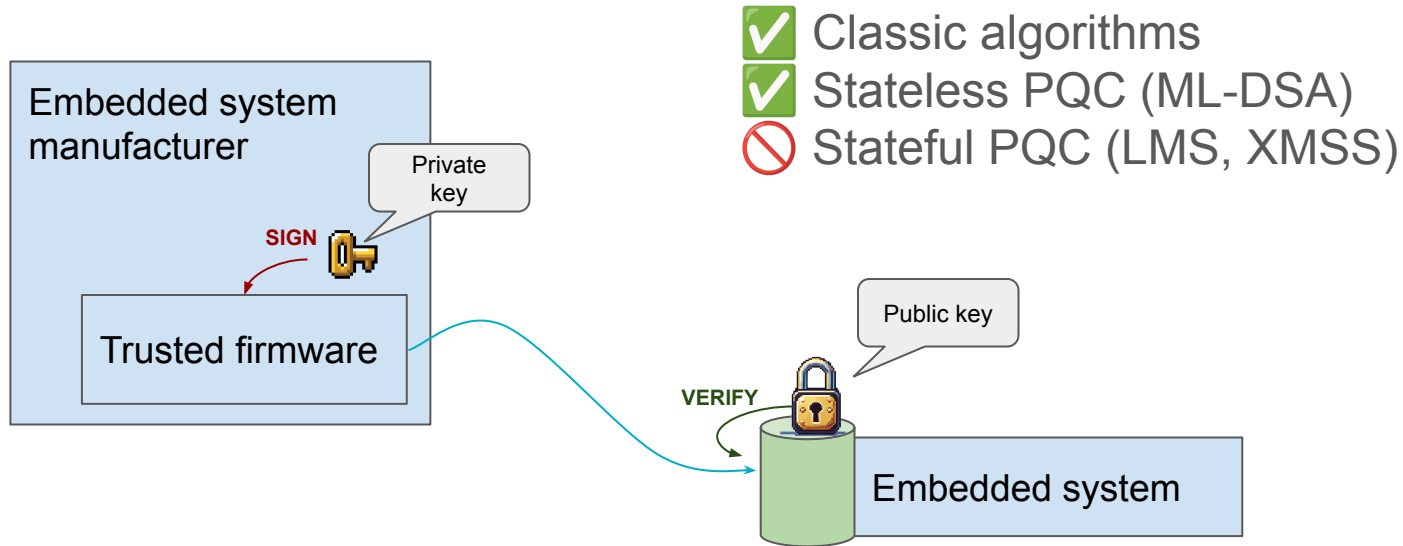
This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force

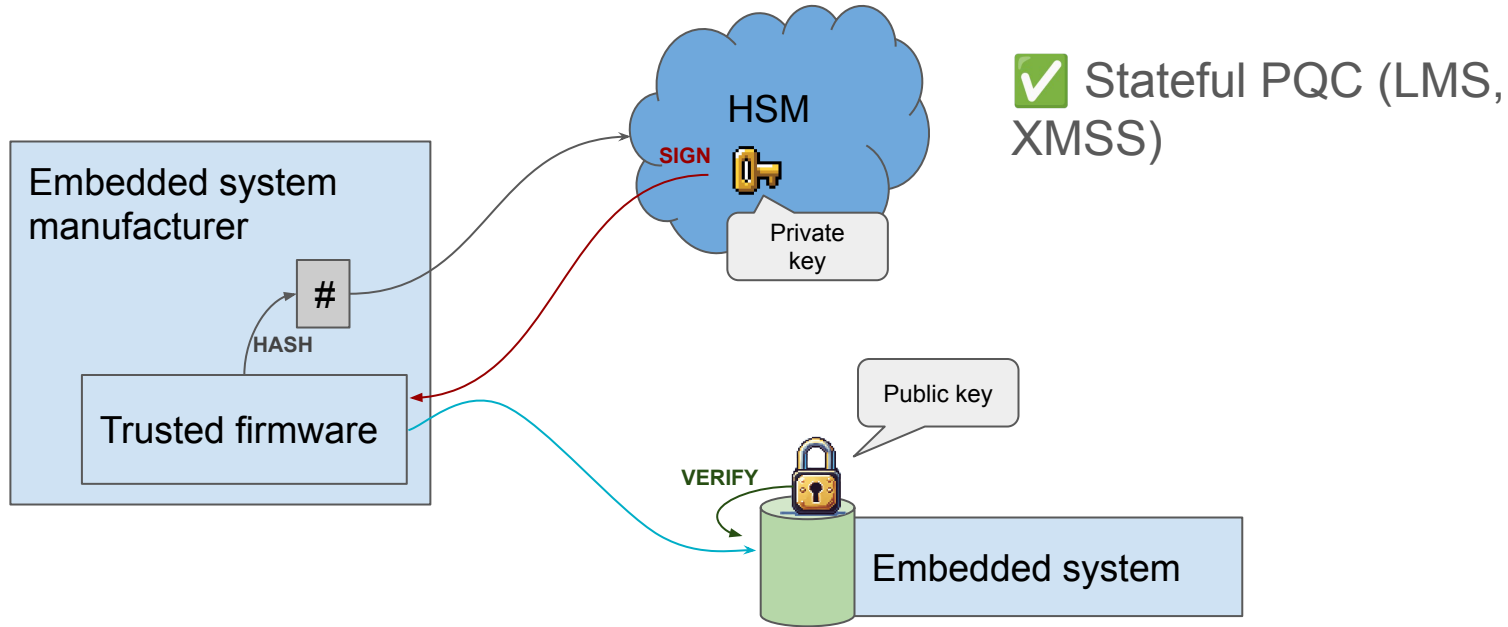
Secure boot: signed firmware



Secure boot: signed firmware



Secure boot with stateful hash-based crypto



Nist approved PQC algorithms

- **FIPS 203**
 - Specifies **ML-KEM**
 - Based on **CRYSTALS-KYBER**
- **FIPS 204**
 - Specifies **ML-DSA**
 - Based on **CRYSTALS-Dilithium**
- **FIPS 205**
 - Specifies **SLH-DSA**
 - Based on **SPHINCS+** winner

Nist approved PQC algorithms

- **FIPS 203**

- Specifies **ML-KEM**
- Based on **CRYSTALS-KYBER**

- **FIPS 204**

- Specifies **ML-DSA**
- Based on **CRYSTALS-Dilithium**

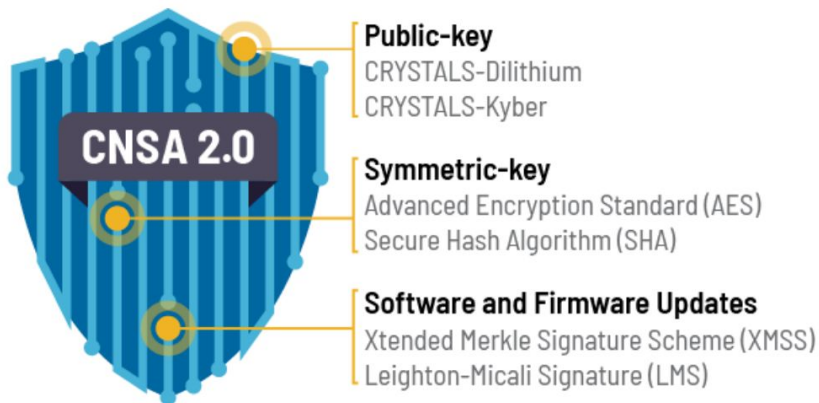


- **FIPS 205**

- Specifies **SLH-DSA**
- Based on **SPHINCS+** winner

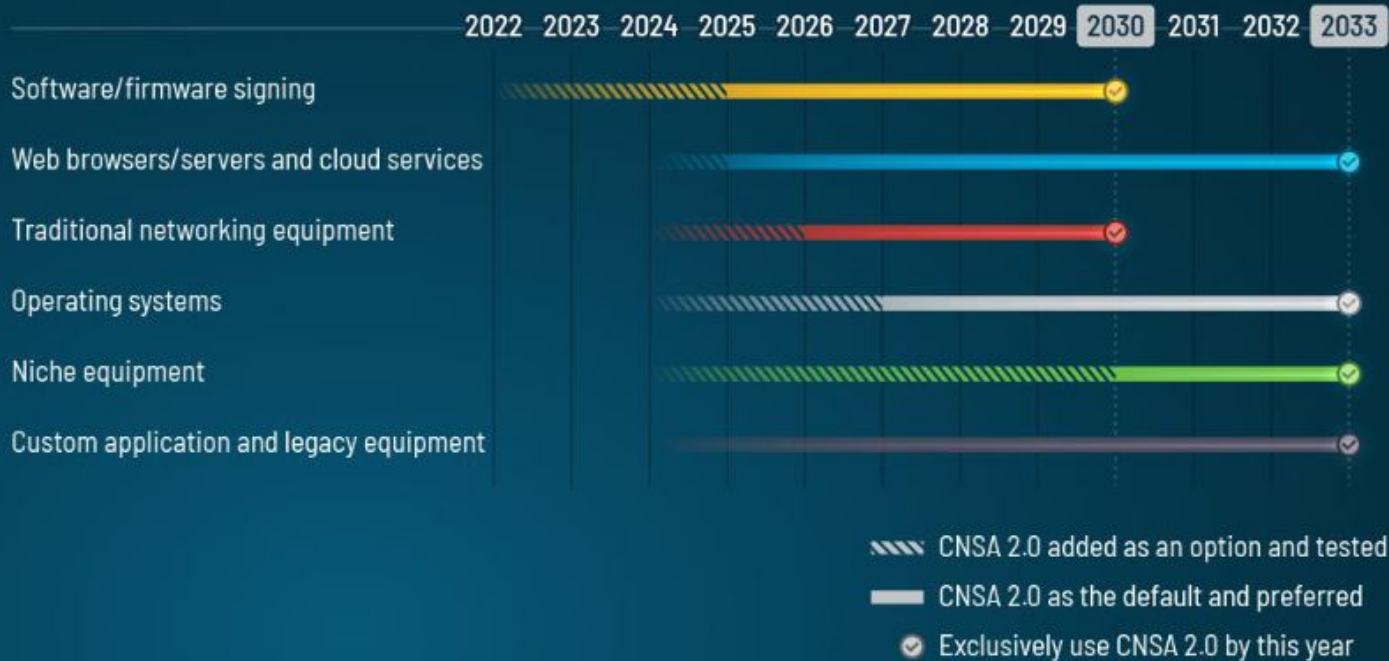


Announcing the Commercial National Security Algorithm Suite 2.0



- Notifies parties involved in **National Security Systems (NSS)** that **new requirements are coming**
- Requirements will **mandate a switch to post-quantum algorithms by 2030**
- **Mandate starting migration to PQC secure boot by 2025**
- Mandate supporting post-quantum algorithms **exclusively by 2033**
- Released **September 2022**

CNSA 2.0 Timeline





COMMISSION RECOMMENDATION

of 11.4.2024

on a Coordinated Implementation Roadmap for the transition to Post-Quantum Cryptography

(§ 2.7) *The Post-Quantum Cryptography Coordinated Implementation Roadmap should be available after a period of two years following the publication of this Recommendation, which will be followed by the development and further adaptation of Post-Quantum Cryptography transition plans of individual Member States, in accordance with the principles set out in the Post-Quantum Cryptography Coordinated Implementation Roadmap.*

- “Should” define a roadmap by April 2026, but no mentions of algorithms yet
- Germany, France, The Netherlands leading the effort

Post-Quantum Migration strategies

- **Immediate Concerns**

- Data Harvesting (Harvest Now, Decrypt Later)

- Encrypted data **harvested now** by malicious actors, then **decrypted later** when quantum computers are available

- Long-Lived Devices (Deploy and Forget)

- Devices being **deployed** to the field and then **forgotten** will be susceptible to attack later when quantum computers are available

- **Migration paths**

- **Use Hybrid Signature schemes** (ex: ECDSA + PQC via dual algorithm signature verifications)

- **Use Hybrid Key Establishment** (ex: ECDHE + PQC)

- **Double the symmetric key size** (use 256-bit cipher)

- **Bonus: Stay FIPS 140-3 compliant** (NIST Certificate #4718; sunsets in 2029)

Post-Quantum Migration strategies

- **Immediate Concerns**

- Data Harvesting (Harvest Now, Decrypt Later)

- Encrypted data **harvested now** by malicious actors, then **decrypted later** when quantum computers are available

- Long-Lived Devices (Deploy and Forget)

- Devices being **deployed** to the field and then **forgotten** will be susceptible to attack later when quantum computers are available

- **Migration paths**

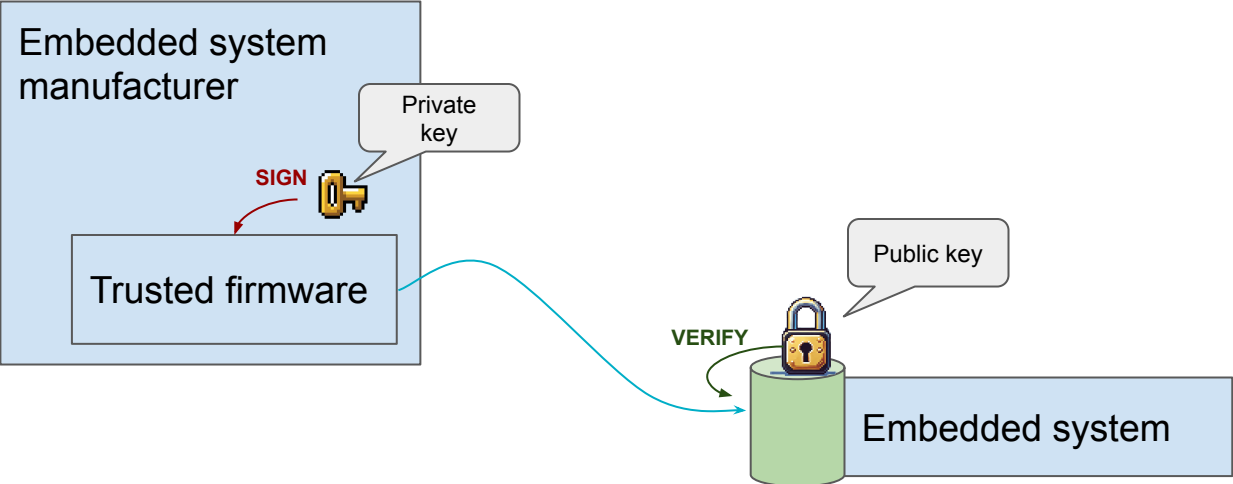
- Use Hybrid Signature schemes (ex: ECDSA + PQC via dual algorithm signature verifications)

- **Use Hybrid Key Establishment** (ex: ECDHE + PQC)

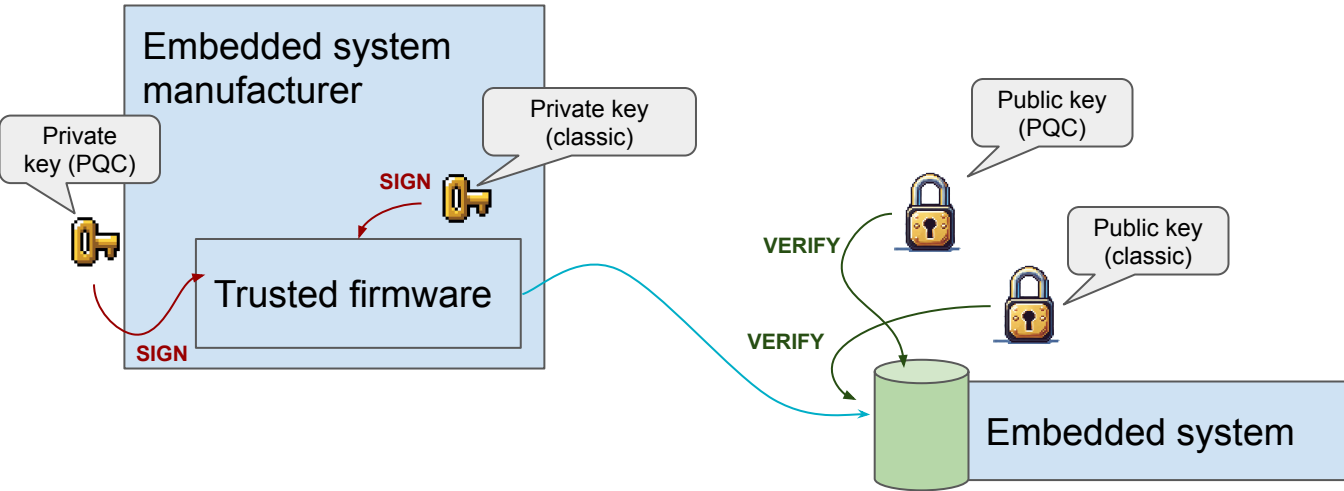
- **Double the symmetric key size** (use 256-bit cipher)

- **Bonus: Stay FIPS 140-3 compliant** (NIST Certificate #4718; sunsets in 2029)

Secure boot: signed firmware



Secure boot: hybrid firmware authentication



ML-DSA-87 + ECDSA521 is “best in class” according to current recommendations from NIST/CNSA

Questions?

facts@wolfssl.com

www.wolfssl.com

