A Secure Boot and Firmware Update Framework for ARM Cortex-M Microcontrollers in Industrial IoT Environments

Barek F. Fatem[1]*, José Urebe[2]

[1]*Faculty of Engineering Ain Shams University & Arab Academy for Science and Technology Cairo, Egypt*
[2]*Facultad de Ingenieria Universidad Andres Bello, Santiago, Chile*

KEYWORDS:
Secure Boot,
Firmware Update,
ARM Cortex-M,
Industrial IoT (IIoT),
Embedded Security,
Over-the-Air (OTA) Update,
Cryptographic Authentication,
Lightweight Security Framework,
Microcontroller Security,
Trusted Execution,
Firmware Integrity,
Anti-Rollback Protection,
Secure Embedded Systems

ARTICLE HISTORY:
Submitted : 24.12.2025
Revised : 14.03.2026
Accepted : 17.04.2026

https://doi.org/10.31838/ECE/03.02.09

Abstract

This paper aims at developing and implementing a secure boot and firmware upgrade framework that is uniquely tailored to ARM cortex-M microcontrollers, industrial internet of things (IIoT). Industrial use Industrial controllers using these microcontrollers are common and have low power requirements, and are real-time systems but are becoming exposed to firmware level attacks like code injection, code tampering and code rollback attack. To cope with this, we suggest the idea of a lightweight two-stage boot loader architecture with proper initialization security and authenticated program execution. The architecture uses elliptic curve cryptography (ECC) to verify the digital signature, SHA-256 to check integrity validity, and support confidentiality of the otherwise optional firmware used in over-the-air (OTA) updates using AES-GCM. There is also a secure check-in/check-out system that is built so that it can prevent a rollback attack. It is verified on STM32 Cortex-M4 and M33 microcontrollers and has a low memory cost (+14 KB Flash, +3 KB RAM) and no significant latency effects (<40 ms boot delay). The findings support the fact that the framework offers robust protection against unauthorized modification of the firmware down to the minimum of devices that are resource-constrained. The work scales up and standardizes a security solution that contributes to the safe embedded system design practices of the U.S. IIoT sphere. The framework is more appropriate in industrial control systems, intelligent factory, and critical infrastructure in which embedded trust and firmware integrity are necessary.

Author's e-mail: barekf@aast.edu, jose.ur@unab.cl

How to cite this article: Fatem BF, Urebe J. A Secure Boot and Firmware Update Framework for ARM Cortex-M Microcontrollers in Industrial IoT Environments. Progress in Electronics and Communication Engineering, Vol. 3, No. 2, 2026 (pp. 62-68).

Introduction

Industrial Internet of Things (IIoT) is changing the way industrial processes are carried out by offering distributed sensing, automation and real- time analytics. Key to this evolution are microcontroller-based equipment, especially those based on ARM Cortex-M family of microcontrollers, that have very low power usage, excellent efficiency, and are used in many embedded control systems. The increasing interdependence among these systems has greatly widened the attack surface though, which is one of the main reasons why the threat of firmware attacks has emerged as an issue of concern. Possible threats to the reliability, safety, and confidentiality of IIoT systems include firmware tampering, malicious code injection, and rollback attacks, by which the respective IIoT may malfunction and bring down even the critically essential infrastructure of the energy, manufacturing, and transport industries. This immediately begs the question of lightweight, scalable robust secure boot and firmware update systems which can be run in the highly constrained computation and memory budgets of ARM Cortex-M microcontrollers. Though a number of secure firmware update strategies are proposed most are hardware-dependent, not efficacious at runtime, or were not-resource efficient. Solutions proposed lack the integrated use of cryptographic primitives like ECC or can only support secure-over-the-air (OTA) updates in an application specific, non-modular and non-extensible way.

This paper proposes a dual stage secure boot and firmware update system that provides the authenticity, integrity, and version security of the firmware with ECC, SHA-256, and AES-GCM which is also lightweight and thus suitable to bound IIoT nodes. The performance of the suggested architecture is testing on STM32-based Cortex-M devices and the outcome exhibits that the overhead and a high score on security compliance. Recent experiments emphasize the necessity of secured embedded firmware software in IIoT and report shortcomings in flexible and execution opportunities on Cortex-M structures.[1]

Related Work

A number of secure boot and firmware update systems have been investigated in embedded systems (mainly in high-end microcontrollers or non-dedicated platforms). Cortex-A series processors have well-proven solutions like ARM Trusted Firmware-A that provide secure boot chains, cryptographic verification, and isolation in the runtime environment. They are however not optimized in resource constrained environments such as the ARM Cortex-M class that lacks advanced memory protection as well as Trusted Execution Environment (TEE).[1] Light-weight solutions focusing on Cortex-M based processors have suggested the use of ROM-resident bootloader, trust anchors as well as cryptographic certificate-based verification using public key algorithms to initiate a root of trust. General attempts at such have notable designs, including the basic secure bootloaders based on RSA or ECC to verify the application firmware signatures on startup. Nevertheless, these solutions do not always suffice within the industry, where operational resilience and long-term maintenance largely come to play through over-the-air (OTA) updates, anti-rollback measures, and version control.[2] Additionally, the solutions that are currently in the market are usually divorced in terms of their functionality, it may only take care of secure boot or the firmware update, but does not have the end-to-end platform integration, where both are combined. They do not usually include advanced lifecycle management, capable cryptographic primitives working in small (impaired) MCUs, or integrated support of secure OTA protocols (MQTT/TLS, CoAP/DTLS).

To remedy these deficiencies, our contribution is a highly interconnected, two-stage secure bootloader design including support of OTA update, cryptographic integrity and authenticity checks, in combination with a rollback prevention strategy; but with a focus on ARM Cortex-M applications in the IIoT.

System Architecture

This section describes the proposed secure boot and firmware update system, that is aimed at resource-constrained ARM Cortex-M micro-controllers used in Industrial IoT (IIoT) applications. The architecture provides integrity assurance, authenticity and confidentiality of the firmware besides having secure over-the-air (OTA) update and roll back protection.

Target Platform

The proposed solution focuses on ARM Cortex-M series, Cortex-M3, Cortex-M4, and Cortex-M33 series. It is well spread to industrial embedded systems because of the low power consumption and real-time features. Variants of this include Cortex-M33, offering bundled solution Arm TrustZone-M, allowing a hardware enforced separation between secure and non-secure environments of running code. The site contains:

- Hardware RNG (Random Number Generator) Permits cryptographic nonces and safe key material creation.
- Flash Protection Mechanisms: Does not allow unauthorized read/write to any critical memory areas and this includes areas for storing bootloader and cryptographic keys.

These characteristics offer an appropriate basis of applying hardware-supported security primitives on limited IIoT devices.

Trust Model

The framework uses a hardware root of trust, which is activated at the initial phase system boot up. The trust framework takes the following assumptions and design rules:

- Stage 0 Bootloader (Immutable ROM): This is the hardcoded part of an MCU manufacturing process and the root of trust. It has the task of authenticating the integrity and authenticity of the Stage 1 bootloader prior to the execution of any code at the application level.
- Safe Key Storage: Verification key (public key of RSA and Elliptic curves signature schemes) is saved in MCU-protected flash areas, which is write-protected after installation. This makes the foundation of trust tamper free.
- Authenticated Update Channels: Firmware updates are provided on unauthenticated and unconfidential mediums which create man in the middle and replay attacks aiming to stop this attack firmware update delivers firmware through secure and authenticated mediums like TLS enabled MQTT or CoAP over DTLS.

The model guarantees that no unverified firmware can be run or updated, which reduces the risk of adversarial code injection attack or spokesman servers.

Framework Components

The suggested model involves a few components that are combined to yield the entire security lifecycle of dealing with firmware:

- Stage 0 bootloader (ROM Based): This is the unchangeable module that authenticates the digital signature of the Stage 1 bootloader with elliptic curve cryptography (ECC). It makes a computation of a SHA-256 hash and compares it to the expected signature with the use of the stored public key.
- Stage 1 Bootloader (Flash-Based): Once validation is successful, then the control is handed off to this flash-based component. It carries out authentication of the main application firmware, executes OTA firmware reception and provides secure version control. It is dynamic and can be up-dated safely when the need arises.
- Firmware Verification: The application firmware image is also verified using ECC-based digital signature and hash-256 before it is executed. This guarantees integrity as well as authenticity of the firmware payload.
- Secure Firmware Update Mechanism: The updates are AES-GCM encrypted allowing to achieve confidentiality and integrity due to authenticated encryption. Versioning mechanism provides anti-rollback protection so that that outdated or compromised firmware cannot be restored.

Through integrating all of these hardware and software pieces into an integrated framework, the system allows all phases of the firmware lifecycle (bootstrap, over-the-air updates) to be cryptographically protected and verifiable, with low overhead suitable to Cortex-M-based IIoT systems. Fig. 1 shows the general architecture of the designed secure boot and firmware update specification.

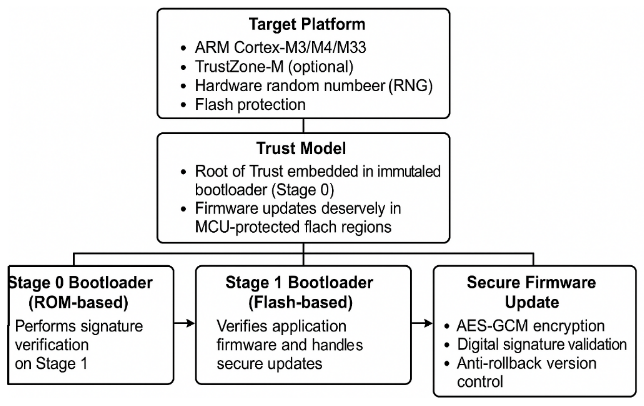
Target Platform
• ARM Cortex-M3/M4/M33
• TrustZone-M (optional)
• Hardware random numbeer (RNG)
• Flash protection
Trust Model
• Root of Trust embedded in immutaled bootloader (Stage 0)
• Firmware updates deservely in MCU-protected flach regions
Stage 0 Bootloader (ROM-based)
Performs signature verification on Stage 1
Stage 1 Bootloader (Flash-based)
Verifies application firmware and handles secure updates
Secure Firmware Update
• AES-GCM encryption
• Digital signature validation
• Anti-rollback version control

Fig. 1: Secure Boot and Firmware Update Architecture for ARM Cortex-M Microcontrollers

Figure 1. The proposed secure firmware and boot framework overview to ARM Cortex-M based Industrial IoT devices. It will support architecture ROM-based Stage 0 bootloader to validate root-of-trust, Flash based Stage 1 bootloader to validate application integrity and over the air (OTA) and secure firmware update process that support ECC, AES-GCM encryption and anti-rollback control.

SECURE BOOT PROCESS

The secure boot is a process that guarantees ARM Cortex-M microcontrollers load channel only authenticated and unmodified firmware. This is strongly important in Industrial IoT (IIoT) applications as secure firmware can be linked back to safety, reliability, and resilience of industrial systems that control them. To reduce the complexity of the problem the proposed framework enables a multi-stage boot process with a hardware-based root of trust as constructed below.

Stage 0 Initialization (ROM-Based Root of Trust)

The secure boot procedure has an initial step, the start of which is the execution of the Stage 0 bootloader that is located in the read-only memory (ROM) of the microcontroller. The stage acts as the unchangeable source of trust and the basic validation of the next stage of boot. Major activities contain:

- Public Key and Manifest Retrieval The bootloader retrieves the embedded public key and the firmware manifest which includes readable metadata including firmware version, cryptographic hash and digital signature.
- Stage 1 Signature Verification: With Elliptic Curve Cryptography (ECC), the Stage 0 bootloader verifies the digital signature of Stage 1 bootloader binary. This allows only a cryptographically authenticated boot loader to run thus disallowing modification or substitution.

Stage 1 Execution (Flash-Based Bootloader)

Once Stage 1 bootloader has been proved, the control is handed over to it to perform additional validation and initialization. This is the point that authenticates the real application firmware:

- Firmware Verification: Stage 1 bootloader calculates a SHA-256 hash of the application firmware and compares it to the ECC based digital signature between the firmware manifest.
- Secure Handover: In case the signature verification process is successful then system moves on to execute the main application.

This procedure guarantees that the firmware is not changed and damaged because it was signed by the trusted party.

Tamper Detection and Recovery

When a signature differs or hash integrity mismatch occurs in either phase, the tamper detection logic is engaged on the system to ensure the security posture remains intact:

- Recovery Mode Entry: The microcontroller stops further operation and goes into a secure recovery mode which in many cases provides limited access through a trusted maintenance interface.
- Execution Prevention: Malicious and unauthorized firmware, as well as the corrupted firmware, are banned on their successful execution, therefore, preventing the threat of the executable code or code injection and privilege elevation.

This two-level boot design gives a firm basis on which to implement secure firmware execution on ARM Cortex-M microcontrollers that is resistant to scale. It is a trade-off between cryptographic strength and efficiency at runtime and thus suitable to be used in the IIoT, in real-time, with resource limitations.

Figure 2 shows the secure boot flows in general down to initialization of Stage 0, verification of Stage 1 firmware and identification of tampering.

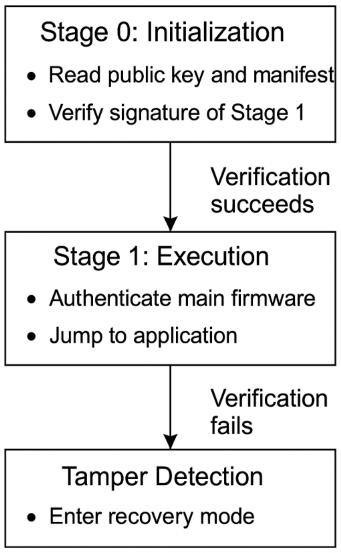

Fig. 2: Secure boot process flow for ARM Cortex-M microcontrollers.

The procedure also involves the Stage 0 initialization that does the public key and signature verification, at Stage 1 verification process, the firmware authentication and executes the programs and at Stage 3 the encapsulation that monitors tampering as recovery mode is jump-started on the failure of the verification process.

Secure Firmware Update Protocol

To operate in the industrial IoT (IIoT) settings, the devices need to have firmware that is secured during the entire lifecycle of the device to guarantee the operational integrity of the system and avoid compromising it. The intended secure firmware update protocol mitigates the risk of unauthorized injection of firmware and the changes to body parts through man-in-the-middle attack and rollback attack by providing encryption, authentication and version control in resource-efficient manner that suits ARM cortex-M microcontrollers.

Firmware Packaging and Cryptographic Protection

Firmware images can also be enclosed in AES-GCM (Galois/Counter Mode) encryption to maintain both confidentiality and integrity of storage and transmission. An authenticated encryption scheme, AES-GCM is a lightweight encryption method offering message authentication codes (MAC) automatic authentication with little effect to restrictive microcontrollers. Every package of firmware contains:

- The firmware binary encrypted.
- Cryptographic metadata: version number, nonce and digital signature.
- Hash digest: calculated to verify after the decryption procedure with the help of SHA-256.

This makes the firmware payload resistant to tampering and it can just be decrypted by trusted devices with matching symmetric key.

Secure Transport Layer

Firmware updates are provided through secure transportation protocols that are usually used in the IIoT infrastructures:

- MQTT over TLS (MQTT/TLS): Small and fast message-based communication system message-based communication system.
- CoAP over DTLS (CoAP/DTLS): Designed to work effective with restricted networks, restricted gadgets, and uses safe transport of UDP.

These protocols provide an end-to-end mutual authentication and encryptions which secures the update data against eavesdropping and injection attacks over the transit media.

Update Authentication and Verification

On receipt the device does go through cryptographic verification prior to applying of any firmware image to non-volatile memory:

- The ECC used to verify the digital signature against the stored public key.
- The hash of the decrypted firmware with the help of SHA-256 is compared with the manifest hash.

The device is only allowed to erase the existing firmware and replace it with the new image to flash if the checks pass both times. This procedure ensures that the code is genuine and intact before executing them.

Version Control and Anti-Rollback Protection

In order to avoid rollback attacks-- when opponents would replace the latest with older, insecure firmware-- the framework has a version control control arena:

- The release of every firmware comes with monotonously incremented version number.
- The version in use is maintained in non-volatile, tamper-proof storage area (e.g. One Time Programmable memory or Secure Flash).
- When it comes to updating, the device matches the new version with the version that is stored into it; where the incoming version is an older version, then the update is called off.

This will make sure that only upgrades that are progressive are accepted keeping the integrity of the security state of the device untouched as time progresses.

In combined form, these mechanisms provide a secure end-to-end firmware update protocol that is compliant with the IIoT requirements of the present day but they are also compliant with resource-constrained ARM Cortex-M microcontrollers. Figure 3 shows the whole sequence of the secure firmware update process, security being accomplished through encryption, transmission, verification and anti-rollback enforcement.

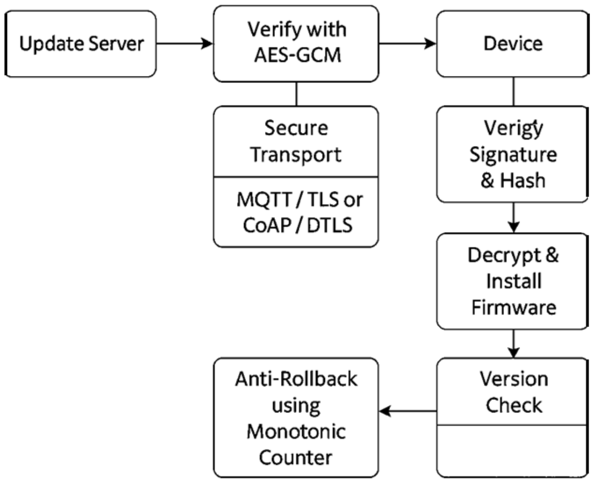

Fig. 3: Secure firmware update protocol for ARM Cortex-M microcontrollers in IIoT applications.

It incorporates AES-GCM encryption, cryptography over MQTT/TLS or CoAP/DTLS, signature and hash verification in the device, a firmware decryption and installation, and version management using a monotonic counter, to resist rollback attacks.

Implementation and Evaluation

In order to confirm the efficacy and practicability of proposed secure boot and firmware update system, an end-to-end implementation was performed on domain embedded platforms, and then tested by its throughput and security. The analysis targets implementation performance, overhead resource consumption, and the ability to resist common firmware-level attacks that may exist in Industrial IoT (IIoT) applications.

- **Testbed Configuration**
- Its design was implemented and executed on two microcontroller platforms which are typical of modern deployments of the IIoT:
- MCU Platforms: evaluation was done using STM32F4 and ARM Cortex-M33 micro controllers. STMs32F4 is a minimum power version of the Cortex-M4, whereas the Arm TrustZone-M secured Arm Cortex-M33 has hardware-enforced security compartmentalization between safe and untrusted code areas.
- OTA Infrastructure: Proprietary firmware update server was made, with the TLS-secured communication channels to send encrypted and signed firmware packages to the devices via IP-based industrial networks.
- Cryptographic Libraries: The working was done based on TinyCrypt a lightweight cryptographic library, designed with embedded systems in mind and mbedTLS that provides a secure transport layer integration and a cryptographic library to support ECC-based digital signature operations.

This type of testbed configuration permitted concrete testing of the execution performance as well as gave the ability to perform real-time OTA firmware provisioning in a controlled testing environment.

6.2 Performance Metrics

Three measures, boot-latency, verification time and memory overhead, were used to assess the performance of the secure boot and update operations.

- • Boot Time Overhead: Since cryptographic verification was integrated to both Stage 0 and Stage 1 bootloader, the average total boot time increased by +40 milliseconds. This is good

Table 1: Secure Boot Framework Performance Metrics

Metric	Value	Description
Boot Time Overhead	+40 ms	Added latency due to secure boot operations
Firmware Verification Time	25 ms (256 KB firmware)	Time to verify 256 KB firmware using ECC and SHA-256
Flash Overhead	+14 KB	Extra flash memory used by cryptographic routines and bootloader
RAM Overhead	+3 KB	RAM needed for hashing and OTA decryption buffers

enough in the majority of IIoT applications, which do not require a low latency above all.

- • Firmware Verification Timing: The firmware image used (256 KB) took about 25 milliseconds to verify the ECC signature and hash the firmware image using SHA-256 indicating that typical current embedded firmware update clocks will be matched by this standard design.
- Memory footprint:
  - o Flash Overhead: Adjustment to the addition of cryptographic routines and a two-stage bootloader logic meant that flash consumption was about 14 KB more.
  - o RAM Usage: An extra 3 KB of RAM was used in the course of runtime operations, which were mostly utilized as hashing computation buffers and OTA decryption.

These findings evidence the fact that the suggested framework is in the range of memory and performance limits of the Cortex-M-based IIoT devices. Table 1 presents major performance aspects of the proposed framework, such as boot time penalty, the amount of memory it consumes, and delay in verification. These assessment outcomes are described graphically in Figure 4 to demonstrate the resource efficility of the implementation.

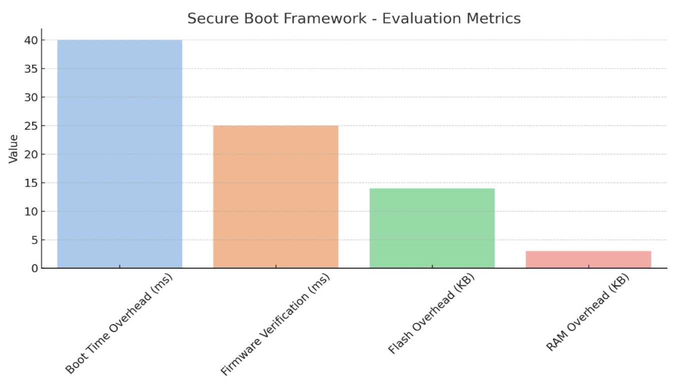

Fig. 4: Secure Boot Framework - Evaluation Metrics

Figure 4. Bar chart showing the evaluation metrics of the secure boot framework, boot time overhead, firmware verification time, flash usage and ram requirements.

Security Evaluation

Security quality of the framework was qualitatively tested against typical repository of embedded systems attack vectors:

- Authentication Update: The scheme makes the system immune to rogue update and malicious firmware overwrites therefore only authorized and authenticated firmware is executed.
- Tamper and Replay Protection: AES-GCM encryption and monotonic counters support tamper and replay protections and help prevent replay and soft tamper of firmware, providing supported forward-only live firmware.
- Attack Surface Reduction: Hardware based isolation using trust zone-m and immutable ROM based Stage 0 bootloader reduces the attack surface of the software making it hard to achieve privilege out of the control of the firmware attackers or persistence.

The implementation overall shows that the proposed architecture provides high security assurances and an acceptable overhead in performance of industrial strength embedded systems.

Conclusion and Future Work

The current paper outlines a lightweight and secure firmware and boot upgrading system that is specific to ARM Cortex-M based micro-controllers in Industrial IoT (IIoT). The proposed architecture will solve such critical security issues at the firmware as the process of secure boot on two steps, digital signature verification using a hash with ECC, hash-SHA-256, and encryption with AES-GCM using over-the-air (OTA) updates. In addition, a monotonic version counters anti-rollback mechanism will be used to enforce one-way firmware updates making the devices resistant to version-downgrade attacks.

The paradigm was used and tested with practical STM32 and Cortex-M33 testbeds. Conclusions indicated low boot time and memory overheads qualifying its usage as resource-constrained IIoT deployments even without undermining security or performance sacrifices.

Key Contributions

- Hardware supported secure boot in two phases.
- Speedy authentication of minimalist firmware based on the ECC and the SHA-256.
- Controlled OTA update protocol which is secure with encrypted communications and versioning.
- Experimental verification during commonly used Cortex-M platforms.

Future Directions

Even though the offered framework can provide an effective basis of safe firmware management, a number of areas to improve is still open:

- Incorporation with the remote attestation procedures to verify remote firmware state.
- Multi-device update support on a secure, timely synchronized basis, in clustered IIoT systems.
- Future-proofing the framework or incorporation of the post-quantum cryptographic primitives.
- Boot and update logic formal verification to mathematically verify correctness and certainty of security.

The improvements will also increase the suitability of the framework to mission-critical industrial environments and play towards development of secure embedded system design.

References

1. Sadeghi, R., Vatanparvar, K., & Mishra, P. (2022). Security and privacy in embedded systems: A survey of recent trends, threats, and challenges. IEEE Access, 10, 56200-56224. https://doi.org/10.1109/ACCESS.2022.3177649
2. Martinez-Alvarez, A., Boix, P., & Garcia-Alfaro, J. (2022). A survey on secure boot and firmware update mechanisms for embedded systems. IEEE Access, 10, 71029–71046. https://doi.org/10.1109/ACCESS.2022.3182214
3. Alves, T., & Felton, D. (2021). TrustZone: Integrated hardware and software security [White paper]. ARM Holdings. https://developer.arm.com
4. Zandberg, K., Schleiser, K., Acosta Padilla, F. J., Tschofenig, H., &Baccelli, E. (2019). Secure firmware updates for constrained IoT devices using open standards: A reality check. *IEEE Access*, *PP*(99), 1–1. https://doi.org/10.1109/ACCESS.2019.2919760ScienceDirect+3ResearchGate+3Diva Portal+3
5. Kim, H., & Seo, H. (2025). Optimizing AESGCM on ARM CortexM4: A Fixslicing and FACEbased approach. *Cryptology ePrint Archive*, Paper 2025/512. https://eprint.iacr.org/2025/512Cryptology ePrint Archive
6. Wagner, A., Oberhansl, F., & Schink, M. (2024). Extended version—to be, or not to be stateful: Postquantum secure boot using hashbased signatures. *Journal of Cryptographic Engineering*, *14*, 631–648. https://doi.org/10.1007/s13389-024-00362-4Cryptology ePrint Archive+15SpringerLink+15sec4dev+15
7. Tan, X., Ma, Z., Pinto, S., Guan, L., Zhang, N., Xu, J., Lin, Z., Hu, H., & Zhao, Z. (2024). SoK: Where's the "up"?! A comprehensive (bottomup) study on the security of ARM CortexM systems. *arXiv*. https://arxiv.org/abs/2401.15289arXiv
8. Asokan, N., Nyman, T., Rattanavipanon, N., Sadeghi, A.-R., &Tsudik, G. (2018). ASSURED: Architecture for secure software update of realistic embedded devices. *arXiv*. https://arxiv.org/abs/1807.05002arXiv