

Regular Article

Lightweight Two-Stage Secure Boot with Automatic Rollback for STM32 IoT Devices

Mai Duc Tho¹, Manh-Thang Nguyen¹, Dang Van Phuc¹, Thanh-Nam Pham²

¹ Academy of Cryptography Techniques, Hanoi, Vietnam

² Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam

Correspondence: Thanh-Nam Pham, ptnam@ictu.edu.vn

Communication: received 29 March 2026, revised 28 June 2026, accepted 05 August 2026

Online publication: 01 September 2026, Digital Object Identifier: 10.21553/rev-jec.455

Abstract– Secure firmware updates are critical for embedded and Internet of Things (IoT) devices, as any failure during the update process can render the system permanently unavailable. Existing secure boot solutions often depend on dedicated hardware security modules or offer only limited recovery mechanisms, which are impractical for low-cost, resource-constrained microcontrollers. This study presents a novel architecture for safe boot using a lightweight two-stage system with automatic rollback features, implemented on an STM32 microcontroller. The system allocates security and update functions between two bootloaders (BL1 and BL2) via a metadata-governed state machine and dual-bank flash memory, thereby obviating the need for supplementary hardware. We have implemented common security protocols within this framework, including AES for encryption, SHA-256 for integrity verification, and RSA-2048 for authentication. We also added a Time-based One-Time Password (TOTP)-based Multi-Factor Authentication (MFA) system to safeguard data transmission over UART. Experimental results conducted on the STM32-F746 IoT hardware platform showed that the proposed solution achieved significant efficiency in boot data recovery across diverse error scenarios and effectively thwarted unauthorized attacks and system degradation. The performance parameters for total system update latencies ranged from 5 to 15.5 seconds, while the cryptographic verification (RSA-2048 + SHA-256) latency consistently remained at or below 258.8 ms. The suggested methodology is appropriate for IoT devices that use low-capacity microcontroller units, such as STM32 platforms. The findings demonstrate that resource-limited IoT devices can achieve strong security and assured system resilience without specialized hardware.

Keywords– IoT Security, Secure Boot, Firmware Update, Rollback Recovery, Embedded Systems.

1 INTRODUCTION

In the era of the Fourth Industrial Revolution, security-critical applications such as industrial automation, smart grid infrastructure, and healthcare systems have seen a massive deployment of embedded systems and Internet of Things (IoT) devices. These devices are often deployed in remote or harsh environments for long operational lifespans, necessitating regular firmware updates to patch security vulnerabilities, fix software bugs, and introduce enhanced functionalities. However, the firmware update process itself serves as a primary attack vector. If not properly secured, it can be exploited for malicious firmware injection, unauthorized code execution, or downgrade attacks, all of which could lead to catastrophic system failures or compromised data integrity [1–3]. A critical yet often overlooked aspect of embedded security is the local update interface. Some developers believe that firmware updates performed via Universal Asynchronous Receiver-Transmitter (UART) are secure because they require physical access to the device. In reality, this "security by obscurity" approach is vulnerable. Attackers equipped with low-cost logic analyzers or sniffing devices can easily intercept unencrypted firmware packets during transmission. Without a robust version-control mechanism, an attacker with physical access can force the

system to accept an older, vulnerable firmware version - a process known as a rollback attack. To mitigate these specific risks, the proposed architecture suggests an additional layer of MFA utilizing TOTP [4]. By requiring a time-synchronized token derived from a shared secret and a Real-Time Clock (RTC), the firmware updates can only be initiated by authorized personnel, even if the UART port is physically accessible.

Recent research has introduced several frameworks to bolster firmware security. Standardized protocols, such as ASSURED, provide robust mechanisms for software distribution and verification [5, 6]. Additionally, hardware-based solutions such as the Arm Platform Security Architecture (PSA) establish a hardware-rooted chain of trust to ensure authenticity during system startup. Despite these advancements, significant challenges remain when deploying such mechanisms on resource-constrained microcontrollers. Many state-of-the-art solutions rely on heavy software stacks or dedicated Hardware Security Modules (HSMs), which significantly increase the Bill of Materials (BOM) and memory footprint. Moreover, there is a notable lack of lightweight, fail-safe recovery mechanisms that can automatically restore a functional state when a firmware update is corrupted or interrupted. To address these challenges, this paper proposes a lightweight, two-stage secure boot architecture specifically designed

for STM32 microcontrollers. The system achieves a high level of security and availability without requiring specialized external hardware components. The key contributions of this work are as follows:

- **Dual-Stage Secure Bootloader:** A robust architecture utilizing dual-bank flash memory and a metadata-driven state machine to ensure fail-safe firmware execution.
- **Multi-Layered Cryptographic Verification:** Integration of AES-128 for confidentiality, SHA-256 for integrity, and RSA-2048 for authenticity, optimized for the Cortex-M7 architecture.
- **MFA-Protected UART Interface:** Implementation of a TOTP-based authentication layer (RFC 6238) to secure the local update process against unauthorized physical tampering.
- **Automatic Rollback Mechanism:** A hardware-less recovery system using a Watchdog Timer to automatically revert to a known-good firmware version in the event of an update failure.

The rest of this paper is organized as follows: Section 2 reviews related works; Section 3 describes the proposed model and method; Section 4 and Section 5 present the experimental results and security evaluation, respectively, and finally, we have a Conclusion.

2 LITERATURE REVIEW

Secure boot and secure firmware update mechanisms have become essential components in modern embedded and Internet of Things (IoT) systems. These mechanisms ensure that only trusted firmware is executed on a device and that updates distributed throughout the device lifecycle are authenticated, integrity-protected, and resistant to rollback or tampering. Numerous architectures, frameworks, and research studies have been proposed to address these requirements, ranging from lightweight software solutions for resource-constrained microcontrollers to comprehensive security frameworks that combine hardware and software protection.

A line of research focuses on evaluating the performance of secure boot mechanisms on embedded platforms. Profentzas *et al.* [7] investigated the performance implications of secure boot implementations on devices such as Raspberry Pi and BeagleBone boards. Their study compared purely software-based verification methods with hardware-assisted approaches using Trusted Platform Modules (TPMs). The results indicated that software-based verification increased boot time only slightly, while hardware security modules introduced additional latency due to communication overhead. The trade-off between enhanced security and system performance is particularly significant in embedded systems with strict startup-time requirements.

In recent years, several standardized and open-source frameworks have been developed to facilitate secure boot implementation across various embedded platforms. One of the most widely adopted solutions is MCUboot, a portable secure bootloader designed for microcontrollers. MCUboot provides a common

infrastructure for firmware authentication, image validation, and safe firmware upgrade mechanisms. It supports multiple cryptographic algorithms, including RSA, ECDSA, and Ed25519, and manages firmware images with dual-slot flash layouts to enable secure firmware swapping and rollback. The bootloader is widely integrated with embedded operating systems such as Zephyr, Apache Mynewt, and RIOT OS, making it a practical solution for many embedded development environments.

Another important framework is Trusted Firmware-M (TF-M) [8], which implements the Secure Processing Environment defined in the Arm Platform Security Architecture (PSA). TF-M offers a complete security framework for ARM Cortex-M devices, including secure boot, cryptographic services, secure storage, and attestation. The framework separates the system into secure and non-secure execution domains using hardware features such as TrustZone-M. In many implementations, TF-M integrates with MCUboot as a secondary bootloader that verifies firmware images during boot.

Beyond boot-time verification, the security of firmware updates throughout the device lifecycle has also been widely studied. The Software Updates for Internet of Things (SUIT) [9] working group of the Internet Engineering Task Force (IETF) has developed a standardized framework for secure firmware updates in constrained devices. The SUIT architecture defines a manifest format that includes cryptographic hashes, version information, and installation instructions, enabling devices to verify the authenticity and integrity of firmware before installation.

Another widely recognized framework for secure software distribution is The Update Framework (TUF) [10], which was designed to protect software update systems against various attacks, including replay, freeze, and key compromise. TUF introduces a role-based trust model where multiple cryptographic keys are assigned to different roles, such as root, targets, snapshot, and timestamp. This separation reduces the risk that a single compromised key will affect the entire update infrastructure.

Although the frameworks discussed above significantly improve the security of firmware boot and update processes, they often rely on relatively complex infrastructures, hardware security features, or extensive software stacks. Such requirements may not be suitable for low-cost microcontrollers that have limited computational resources and memory capacity. Furthermore, while many solutions provide secure firmware verification, fewer studies focus on lightweight recovery mechanisms that guarantee system availability when firmware updates fail or become corrupted.

While the aforementioned frameworks focus heavily on firmware integrity and distribution, the security of the local physical interface during updates is often overlooked. Traditional UART-based updates are vulnerable to unauthorized access and firmware interception if the channel is not encrypted or authenticated [11]. Some researchers have proposed using secondary authenti-

cation factors to secure sensitive operations on embedded devices. For instance, the integration of TOTP mechanisms, as defined in RFC 6238, provides a robust defense against unauthorized physical access by requiring a time-synchronized token [4]. Furthermore, although rollback protection is a standard feature in frameworks like MCUboot, implementing a fully automated recovery process—one that detects execution failures via hardware timers (e.g., Watchdog Timers) and reverts to a previous stable state without manual intervention—remains a critical requirement for high-availability industrial IoT systems. Recent studies have continued to explore efficient boot architectures and secure firmware management for embedded systems. A recent work [12] proposed a single-Flash boot strategy for multi-MCU systems to reduce hardware cost and simplify firmware deployment. In addition, the open-source MCUboot project [13] is widely adopted as a reference implementation of dual-slot secure boot with rollback support for resource-constrained microcontrollers, and we therefore adopt it as a comparison baseline. Complementing this, Nosedá and Künzli [14] demonstrated that even secure-element-hardened MCUboot deployments can be compromised by low-cost fault-injection attacks, highlighting residual security challenges that motivate the additional protection layers introduced in this work.

To overcome the limitations of existing frameworks, this work proposes a lightweight, two-stage secure boot architecture with an integrated automatic rollback mechanism, specifically optimized for STM32 microcontrollers. Beyond standard cryptographic verification using AES-128, SHA-256, and RSA-2048, the proposed approach introduces a Time-based One-Time Password authentication layer to secure local UART updates against unauthorized physical access. By combining a dual-bank flash layout with a metadata-driven state machine and hardware timers, the system achieves robust firmware authentication and reliable autonomous recovery. This comprehensive security framework is designed to operate within the constraints of low-cost embedded devices without requiring specialized hardware security modules.

3 THE PROPOSED MODEL

In this section, we present the proposed Secure Boot architecture designed for the STM32 platform. The model focuses on ensuring the integrity, authenticity, and confidentiality of the firmware while addressing physical security risks through an MFA layer. The system is built upon a two-stage bootloader architecture (BL1 and BL2) to support autonomous rollback and fail-safe updates. Figure 1 depicts the overall system architecture, comprising the Client (a Windows Forms application) and the Target Device (an STM32 MCU). The workflow is divided into three critical phases:

- 1) **MFA Challenge:** Before any data transmission, the client must provide a valid TOTP (Time-based One-Time Password) synchronized with the de-

vice's internal RTC. This ensures that the UART interface remains secured against unauthorized physical access.

- 2) **Firmware Preparation:** The client processes the HEX file by calculating its SHA-256 digest, signing it with an RSA private key, and encrypting the entire package using AES-128 in CBC mode.
- 3) **Secure Processing:** On the device side, Bootloader 2 (BL2) manages the reception and decryption. Upon successful signature verification, the firmware is committed to the inactive flash bank, and the metadata is updated to trigger a boot switch by Bootloader 1 (BL1).

By integrating these cryptographic primitives with a metadata-driven state machine, the system ensures that only verified and authorized firmware images are executed, while maintaining a reliable fallback version in the alternate flash bank.

3.1 Adversary Model and Security Assumptions

In this research, we define a threat landscape based on the Asokan adversary model [5], assuming the attacker has full control over the communication channel. Following the security analysis in [15], we consider the following threat vectors: (1) **Firmware Manipulation:** An adversary may intercept the firmware package during transmission to inject malicious code or modify the binary to alter device functionality [4]. (2) **Rollback Attacks:** Attackers attempt to reinstall an older, officially signed but vulnerable firmware version to exploit previously patched security flaws [7]. (3) **Denial of Service (DoS) via Bricking:** An attacker might intentionally interrupt the power or communication during the update process, aiming to render the device unbootable. (4) **Unauthorized Local Access:** assuming the adversary can access physical debug interfaces (e.g., UART) to trigger unauthorized boot sequences. The proposed architecture is designed to mitigate these threats by establishing a multi-layered defense mechanism, as detailed in the following subsections.

3.2 Two-Stage Bootloader and Rollback Mechanism

The core differentiator of this model is the hierarchical separation of duties between two specialized bootloaders, designed to establish a "Root of Trust" and ensure high system availability.

Bootloader 1 (BL1 - Primary Bootloader): Acting as the immutable first-stage loader and Root of Trust [16], BL1 is responsible for the initial system integrity check. Upon power-up, BL1 reads the system metadata to identify the currently active flash bank. It performs a mandatory RSA-2048 signature verification and SHA-256 hash comparison.

- **Normal Execution:** If the signature is valid, BL1 jumps to the application's entry point.
- **Autonomous Rollback:** If the active bank is detected as corrupted or tampered with, BL1 automatically references the backup bank's metadata and switches the boot target.

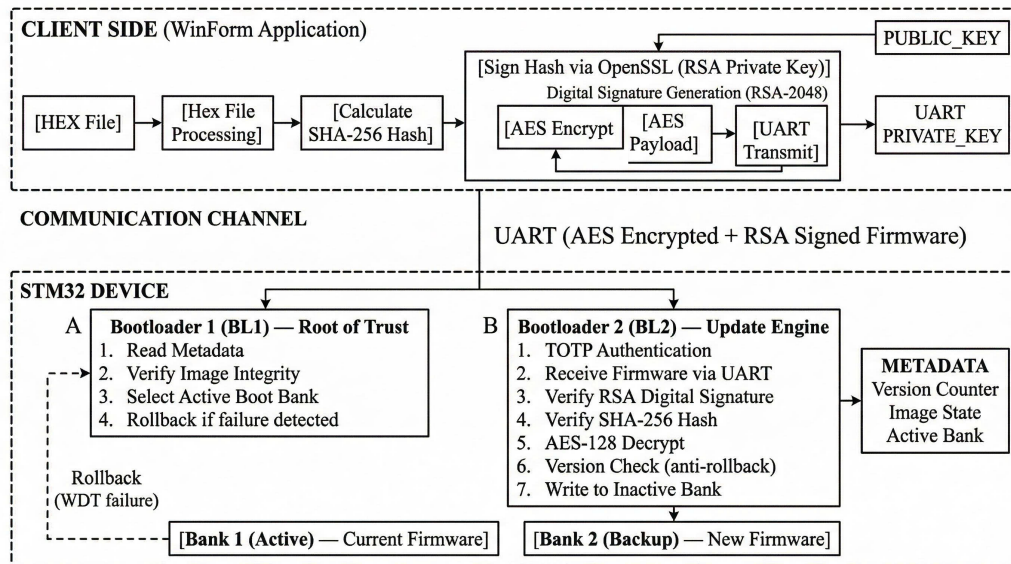


Figure 1. Overall System Block Diagram featuring MFA and Two-Stage Bootloader.

- **Recovery Mode:** If both banks fail verification, the system enters a fail-safe Recovery Mode, keeping the communication interface active for emergency updates, thereby preventing a "bricked" state.

Bootloader 2 (BL2 - Update Engine): BL2 is a mutable second-stage loader that manages the secure update pipeline. Unlike traditional bootloaders, BL2 integrates an MFA challenge. Before accepting firmware packets, it verifies a TOTP token provided by the user via UART. Once authenticated, BL2 performs AES-128 decryption and writes the decrypted data to the inactive flash bank. To protect against system hangs during firmware updates or during the first boot of a new firmware, BL2 initializes a Watchdog Timer (WDT). If the timer expires before the new firmware can confirm a successful boot, a hardware reset is triggered, and BL1 reverts the system to the stable backup version.

3.3 Metadata and Dual-bank Flash Management

The system employs a metadata-driven approach to synchronize the states between BL1 and BL2. The Metadata Header is stored in a dedicated, protected sector of the flash memory.

1. Metadata Structure: To provide clear technical depth, the fields within the metadata header are defined in Table I.

2. Dual-Bank Flash Strategy: The STM32F746 microcontroller is configured in a dual-bank organization, providing two symmetrical 512 KB regions. Our model utilizes an "Atomic Update" strategy:

- **Bank 1 (0x08000000):** Primary execution slot for the stable firmware.
- **Bank 2 (0x08080000):** Staging slot for new updates or backup images.

During an update, BL2 writes exclusively to the inactive bank, and the corresponding metadata entry is marked as RECEIVED. The Image State field is promoted to ACTIVE only after the newly written

Table I
TECHNICAL SPECIFICATION OF THE SYSTEM METADATA HEADER

Field Name	Size (Bytes)	Description
Magic Number	4	Identifies valid metadata structure.
Version Counter	4	Monotonic value for anti-rollback protection.
Firmware Size	4	Total length of the binary image in bytes.
SHA-256 Hash	32	Reference digest for integrity verification.
RSA Signature	256	PKCS#1 v1.5 digital signature.
Image State	4	Current status: ACTIVE, RECEIVED, or FAILED.

firmware has successfully completed its first Power-On Self-Test (POST). If a power failure or transmission interruption occurs before this confirmation, the metadata retains its previous state, and BL1 continues to boot the existing valid firmware from the currently active bank. This mechanism is designed to ensure that at least one functional firmware version remains available after interrupted or corrupted updates, thereby improving system availability.

3.4 Cryptographic Primitives and Security Protocol

The proposed system utilizes a hybrid cryptographic approach to balance high-level security with the computational constraints of the STM32F746 microcontroller. We integrate three core primitives for the firmware update process and a time-based mechanism for physical access control.

3.4.1 Symmetric Encryption (AES-128-CBC): To protect the firmware's intellectual property during UART transmission, AES-128 is employed as defined in FIPS 197 [17]. We specifically implement the Cipher Block Chaining (CBC) mode [18] to ensure that identical plaintext blocks yield distinct ciphertexts, thereby preventing pattern-based cryptanalysis. The

encryption of a plaintext block P_i is defined as follows

$$C_i = E_K(P_i \oplus C_{i-1}), \quad (1)$$

where C_i is the resulting ciphertext block, E_K denotes the AES encryption function under key K , and C_0 represents the Initialization Vector (IV). By leveraging the STM32's hardware cryptographic processor (CRYP), the decryption process achieves minimal CPU overhead, ensuring that firmware reception does not become a system bottleneck.

3.4.2 Message Integrity (SHA-256): Integrity is maintained via the SHA-256 hashing algorithm [19], which follows the Merkle-Damgård construction. It processes the firmware binary in 512-bit blocks to produce a unique 256-bit digest. The iterative compression function is modeled as

$$H_{i+1} = H_i + f(H_i, M_i), \quad (2)$$

where f is the compression function and M_i is the current message block. This primitive is critical for the Self-Test phase during the secure boot process, where BL1 recalculates the hash of the stored binary to detect flash degradation or unauthorized modifications.

3.4.3 Digital Signature (RSA-2048): Authenticity is verified using RSA-2048 with the RSASSA-PKCS1-v1_5 padding scheme [20]. The device stores the SHA-256 hash of the firmware, which is then verified against the digital signature s using the public key (n, e) through modular exponentiation

$$m = s^e \pmod{n}. \quad (3)$$

The device only stores the *Public Key*, ensuring that even if the hardware is physically compromised, an attacker cannot generate a valid signature for malicious firmware. This establishes a robust "Chain of Trust" that originates in the developer's environment and extends to the MCU's internal flash.

Software-based RSA-2048 dominates the cryptographic-verification latency, which ranges from 245.42 ms for a 4 KB image to 258.8 ms for a 13.5 KB image (Table II); the size-dependent component is contributed by SHA-256 hashing, while RSA-2048 is the constant, dominant cost owing to the absence of a hardware PKA. This delay is limited to cold boots or post-update initializations. In long-running IoT applications, this singular sub-second burden has a negligible effect on overall availability and does not impair the device's real-time operational performance throughout active runtime. Consequently, RSA-2048 continues to serve as a robust and secure foundation for this system. Subsequent research will investigate ECDSA secp256r1 for enhanced latency optimization.

3.4.4 Multi-Factor Authentication (TOTP): To secure the physical UART interface against unauthorized access, we implement the Time-based One-Time Password (TOTP) algorithm as defined in RFC 6238 [21]. This mechanism generates a dynamic 6-digit code based on a shared secret K and the current Unix time T , derived

from the internal RTC

$$TOTP = \text{Truncate}(\text{HMAC} - \text{SHA1}(K, T)). \quad (4)$$

The time step X is set to 30 seconds to provide an optimal balance between usability and security. This layer prevents unauthorized personnel from initiating firmware updates, even if they possess physical access to the device's serial pins. The RSA public key is stored in the protected BL1 region, while the shared secret key K used for TOTP generation is provisioned during device initialization and stored in protected Flash memory. To reduce the risk of unauthorized key extraction, STM32 Readout Protection (RDP) and Flash protection mechanisms may be enabled. Since the STM32F746 does not support TrustZone-M, the current implementation relies on its built-in protection features. Advanced invasive physical attacks remain outside the scope of this work.

4 RESULTS AND DISCUSSION

This section evaluates the performance and security effectiveness of the proposed two-stage Secure Boot system on the STM32F746 platform, focusing on authentication timing, resource consumption, and resilience against common attack vectors.

4.1 Experimental Setup

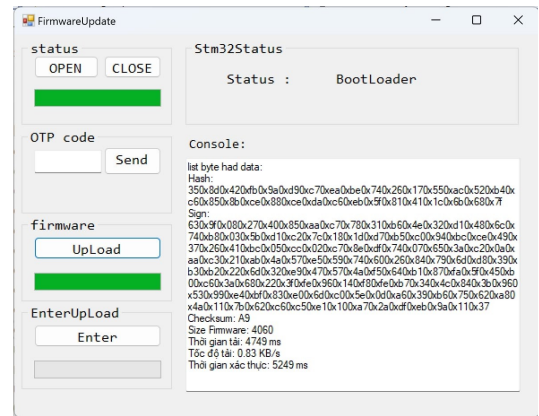


Figure 2. Experimental execution log for a 4,060-byte firmware update.

The experimental architecture comprises a Host PC running a custom Windows Forms management application and a Target Device utilizing the STM32F746 Discovery Kit (ARM Cortex-M7). Communication is established via a UART interface at 9,600 bps, using the onboard ST-LINK V2.1 virtual COM port for stable data exchange. To evaluate the system's performance in typical IoT scenarios, a dataset of firmware images ranging from 4KB to 13.5KB was employed, representing lightweight control applications and RTOS-based tasks. The PC application manages data fragmentation and packet delivery, while the STM32 handles real-time reception and integrity verification. This setup enables precise measurement of transmission latency, throughput, and the reliability of the update process under

constrained bandwidth. By simulating these real-world conditions, the environment provides a robust framework for testing Over-The-Air (OTA) update mechanisms or the efficiency of serial bootloaders.

4.2 Performance Evaluation and Bottleneck Analysis

The execution time of the secure update process is divided into three main phases: MFA verification, UART transmission, and Cryptographic authentication (Hash + RSA). The measured results are summarized in Table II. Authentication time increases with firmware size, but the growth is dominated by UART transmission overhead. However, the combined RSA-2048 + SHA-256 cryptographic verification takes 245.42 ms for a 4 KB image and 258.8 ms for a 13.5 KB image. This latency is dominated by software RSA-2048 due to the lack of a dedicated Public Key Accelerator (PKA), whereas SHA-256 and AES-128 are near-instantaneous thanks to the on-chip HASH and CRYPT peripherals.

Table II
SYSTEM EXECUTION TIME BREAKDOWN

Phase	Firmware 4 KB	Firmware 13.5 KB
TOTP MFA Verification	45 ms	48 ms
UART Transmission	4.749 s	15.232 s
Cryptographic Verification (RSA-2048 + SHA-256)	245.42 ms	258.8 ms
Total Time	5.039 s	15.538 s

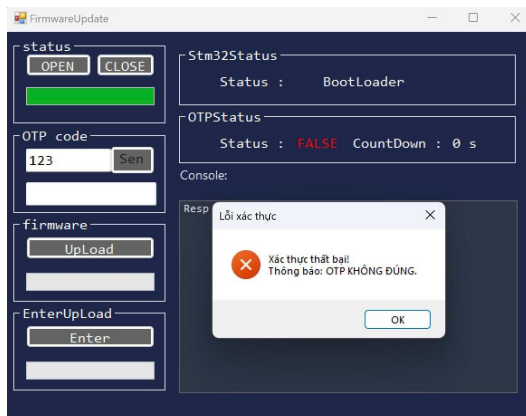


Figure 3. Serial terminal logs displaying MFA and RSA verification steps.

4.3 Security Case Studies

To validate the system's robustness, three attack scenarios were simulated based on the experimental results observed in the terminal logs (Figure 3).

- 1) Unauthorized Access (MFA Failure): When an incorrect TOTP token was entered, BL2 immediately terminated the UART session. This confirms that the MFA layer effectively secures the physical interface against unauthorized update attempts.
- 2) Integrity Violation (Signature Failure): A single byte in the firmware binary was modified before

transmission. The system successfully detected the mismatch between the recalculated SHA-256 hash and the decrypted RSA signature, resulting in an "Authentication Failed" status.

- 3) Power Loss (Atomic Update Failure): Power was intentionally disconnected during the flash writing process. Upon reboot, BL1 detected that the image had not reached the *ACTIVE* state (i.e., it remained in the *RECEIVED/FAILED* state) and successfully executed the rollback mechanism, booting the previous stable firmware from the alternate (backup) bank.

Figure 4. System log showing successful RSA signature verification and jump to application by BL1.

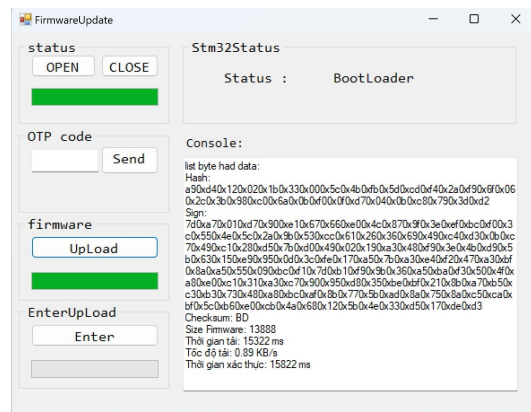


Figure 5. MFA challenge execution in BL2 requiring a valid TOTP token for firmware reception.

Table III compares the proposed framework with representative software-based [11], MCUboot-based [13], TrustZone-M-based [8], and hardware-assisted [15] secure firmware update solutions. The proposed framework achieves a balanced trade-off between security, implementation complexity, and deployment cost for resource-constrained IoT devices. While the FPGA/PUF-based solution in [15] offers strong hardware-assisted protection and faster cryptographic processing, it requires dedicated hardware resources

Table III
COMPARISON OF SECURITY FEATURES AND IMPLEMENTATION CHARACTERISTICS BETWEEN THE PROPOSED MODEL AND RELATED WORKS

Criteria	This work	TF-M [8]	Keleman <i>et al.</i> [11]	MCUBoot [13]	Falas <i>et al.</i> [15]
Approach	Two-stage Hybrid Boot	PSA Secure Processing	Software-based	Dual-slot Secure Boot	Hardware-based (PUF)
Firmware Size	13.5 KB	Not Reported	Typical (10–15 KB)	Not Reported	Linux kernel (MB)
Main Algorithms	RSA-2048, SHA-256, AES	RSA/ECDSA, SHA-256	RSA-2048, SHA-256	RSA/ECDSA, SHA-256	Device Fingerprint
Crypto Verification Time	≤ 258.8 ms	Configuration-dependent	350–500 ms	Configuration-dependent	< 100 ms (HW Accel.)
MFA Integration	Yes (TOTP)	None	None	None	None
Firmware Update Recovery	Auto-Rollback	Integrated MCUBoot	No	Rollback Support	Limited
Resiliency	Auto-Rollback (Active)	Integrity + Isolation	Basic Integrity	Integrity + Rollback	Integrity + Confidentiality
Hardware Overhead	Lightweight (Optimized)	High	Medium (Generic Overhead)	Medium	High (Costly Hardware)
Cost Efficiency	High	Low	Medium	High	Low

and incurs higher implementation costs. The MCU-boot dual-slot approach [13] provides authenticated firmware updates and rollback support via a dual-slot architecture; however, it does not incorporate multi-factor authentication to authorize updates. Moreover, as shown by Nosedá and Künzli [14], even secure-element-hardened MCUboot can be defeated by low-cost fault injection, which further motivates the layered design adopted here.

In contrast to the software-based approach in [11], the proposed architecture employs an optimized two-stage bootloader design that achieves a cryptographic verification time of 258.80 ms while maintaining a lightweight memory footprint. Furthermore, the proposed framework is the only solution among the compared approaches to integrate TOTP-based multi-factor authentication, providing an additional layer of protection against unauthorized firmware update attempts. Combined with the automatic rollback mechanism, the proposed architecture enhances system availability and resilience by enabling reliable recovery from failed or malicious firmware updates without requiring additional security hardware.

5 SECURITY EVALUATION AND DISCUSSION

This section evaluates the proposed architecture's resilience against the threats defined in Section 3 and discusses its performance trade-offs relative to existing hardware-based solutions.

5.1 Resilience Against Adversary Models

The effectiveness of our multi-layered defense mechanism is analyzed based on the primary threat vectors identified in [5, 15]:

Firmware Manipulation and Authenticity: By employing RSA-2048 digital signatures in conjunction with SHA-256 integrity checks, any unauthorized modification to the binary—even by a single bit—will result in a signature mismatch. Unlike the framework in [15], which uses FPGA-based crypto engines, our software implementation ensures that Bootloader 1 (BL1) validates authenticity before execution, maintaining a rigid Chain of Trust without specialized hardware.

Mitigation of Rollback Attacks: Our metadata-driven approach incorporates a monotonic version counter. During the update process, BL2 compares the incoming firmware's version against the metadata

stored in the protected sector. If the version is lower than the current one, the update is rejected, denying rollback attempts, which are often used to reintroduce known vulnerabilities.

Prevention of Bricking via Dual-Bank Redundancy:

A critical concern in remote IoT deployments is system failure during an update. Our state-machine-driven rollback mechanism ensures that the active firmware in Bank 1 remains untouched until the new image in Bank 2 is fully verified and successfully boots, as confirmed by the Watchdog Timer (WDT). This approach addresses the "minimal downtime" and "availability" objectives discussed in [15] by ensuring a functional fallback version is always present.

Physical Interface Security via MFA: A unique contribution of this work is the TOTP-based MFA layer. While standard industrial bootloaders often leave UART/SWD interfaces exposed, our mechanism introduces a temporal barrier to them. Even with physical access to the device, an adversary cannot trigger a firmware update without a dynamic token synchronized with the internal RTC, effectively securing the physical attack surface.

Under the adversary model defined in Section 3.1, firmware authenticity is guaranteed through RSA-2048 signature verification, while integrity is ensured by SHA-256 hash validation. Firmware validation metadata and digital signatures prevent unauthorized firmware modification. System availability is maintained through dual-bank Flash redundancy and watchdog-assisted automatic rollback. Unauthorized local firmware update attempts are restricted through TOTP-based authentication, assuming the shared secret remains protected.

5.2 Resource and Performance Analysis

The practicality of the system was measured on the STM32F746 platform to evaluate the overhead of the added security layers:

Memory Footprint: The entire security suite—comprising BL1, BL2, and the cryptographic libraries—occupies approximately 48 KB of Flash memory. This is remarkably efficient compared to Trusted Execution Environment (TEE) based solutions like TF-M, which often exceed 100 KB, making our model highly suitable for resource-constrained microcontrollers.

Boot Latency: Experimental results indicate that the per-boot security overhead—cryptographic verifica-

tion (RSA-2048 + SHA-256, 245.42 ms at 4 KB rising to 258.8 ms at 13.5 KB) plus TOTP MFA validation (45–48 ms)—remains approximately 0.29–0.31 s, consistent with the breakdown in Table II. For industrial monitoring and IoT applications where sub-second cold starts are not mission-critical, this overhead is a justifiable trade-off for the enhanced security posture.

5.3 Comparative Discussion

Unlike existing frameworks such as MCuboot [13], which primarily focus on image swapping and basic integrity checks, our solution introduces an MFA layer and a hardware-independent Root of Trust. While Falas *et al.* [15] achieve high performance through FPGA fabric, our architecture eliminates the need for expensive external components. By leveraging the STM32's internal CRYPT and a dual-stage software design, we provide a cost-effective yet robust template for secure firmware lifecycle management in the industrial sector.

6 CONCLUSION

This paper presented a lightweight, two-stage secure boot architecture with an automatic rollback mechanism specifically designed for STM32-based IoT devices. By logically separating the update engine (BL2) from the Root of Trust (BL1) and employing a metadata-driven state machine, the proposed system establishes a robust Chain of Trust without requiring external hardware security modules. A significant contribution of this work is the integration of a TOTP-based MFA layer, which effectively secures the UART interface against unauthorized physical access. Experimental results on the STM32 platform validated the framework's efficiency, with cryptographic verification (RSA-2048 + SHA-256) overhead of at most 258.8 ms and total update latencies ranging from 5 to 15.5 seconds. Furthermore, the dual-bank flash configuration successfully recovered from the evaluated failure scenarios, demonstrating improved system resilience against update failures and firmware tampering. The future work will focus on transitioning to Elliptic Curve Cryptography (ECC) to reduce computational latency and energy consumption. We aim to implement secure time synchronization to enhance TOTP reliability and to evaluate the system's robustness against advanced physical threats, such as side-channel analysis and voltage glitching.

REFERENCES

- [1] A. Marchand, Y. Imine, H. Ouarnoughi, T. Tarridec, and A. Gallais, "Firmware integrity protection: A survey," *IEEE Access*, vol. 11, pp. 77 952–77 979, 2023.
- [2] L. Catuogno and C. Galdi, "Secure firmware update: Challenges and solutions," *Cryptography*, vol. 7, no. 2, 2023.
- [3] L. Zhao, H. Shuang, S. Xu, W. Huang, R. Cui, P. Bettadpur, and D. Lie, "A Survey of Hardware Improvements to Secure Program Execution," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–37, Oct. 2024.
- [4] O. F. Adejo, "The Importance of Secure Firmware Updates in Maintaining System Integrity," *Progress in Electronics and Communication Engineering*, Sep 2024.
- [5] N. Asokan, T. Nyman, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik, "Assured: Architecture for secure software update of realistic embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2290–2300, 2018.
- [6] P. Gupta, A. Sinha, P. K. Srivastava, A. Perti, and A. K. Singh, "Security Implementations in IoT Using Digital Signature," in *Proceedings of the Innovations in Electrical and Electronic Engineering*, 2021, pp. 523–535.
- [7] C. Profentzas, M. Günes, Y. Nikolakopoulos, O. Landsiedel, and M. Almgren, "Performance of secure boot in embedded systems," in *Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 198–204.
- [8] "Trusted Firmware-M Implementation Overview — SimpleLink™ CC13XX/CC26XX SDK TI 15.4-Stack User's Guide 6.41.00.00 documentation." [Online]. Available: https://software-dl.ti.com/simplelink/esd/simplelink_cc13xx_cc26xx_sdk/latest/exports/docs/ti154stack/html/security-tfm/tfm_architecture.html
- [9] B. Moran, H. Tschofenig, and H. Birkholz, "A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices," Internet Engineering Task Force, Request for Comments RFC 9124, 2022, num Pages: 40.
- [10] "TUF Getting started," section: docs. [Online]. Available: <https://theupdateframework.io/docs/getting-started/>
- [11] L. Keleman, D. Matić, M. Popović, and I. Kaštelan, "Secure firmware update in embedded systems," in *Proceedings of the 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, 2019, pp. 16–19.
- [12] F. Peng, M. Lei, X. Jiang, X. Kong, and Z. Qian, "Design of multi-mcu system based on single flash boot strategy," in *Proceedings of the Symposium on Advances in Electrical, Electronics and Computer Engineering*, 2023.
- [13] "M Cuboot," access: 2026-June-20th. [Online]. Available: <https://docs.mcuboot.com/design.html>
- [14] M. Nosedá and S. Künzli, "Attacking Secure-Element-Hardened MCuboot Using a Low-Cost Fault Injection Toolkit," in *Proceedings of the International Conference on Information Technology and Communications Security*. Springer, 2023, pp. 126–143.
- [15] S. Falas, C. Konstantinou, and M. K. Michael, "A hardware-based framework for secure firmware updates on embedded systems," in *Proceedings of the 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2019, pp. 211–216.
- [16] A. Regenscheid, "Platform firmware resiliency guidelines," National Institute of Standards and Technology Special Publication 800-193, Tech. Rep., 2018.
- [17] J. Daemen and V. Rijmen, "Advanced encryption standard (AES)," Federal Information Processing Standards Publication (FIPS) 197, Tech. Rep., 2001.
- [18] I. Hammad, K. El-Sankary, and E. El-Masry, "Advanced Encryption Standard (AES) Implementation in Embedded Systems," in *Embedded Systems: Hardware, Design, and Implementation*. John Wiley & Sons, 2012, pp. 291–317.
- [19] NIST, "Secure hash standard (shs)," Federal Information Processing Standards Publication (FIPS) 180-4, Tech. Rep., 2015.
- [20] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2," RSA Laboratories' Public-Key Cryptography Standards (PKCS), RFC 8017, 2016.
- [21] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "TOTP: Time-based one-time password algorithm," Internet Requests for Comments, IETF, RFC 6238, May 2011. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6238>



Hanoi, Vietnam. His research interests include Machine Learning, Deep Learning, and their applications in IoT security.

Duc-Tho Mai graduated from Hanoi University of Science and Technology (HUST), Vietnam, in 2010 and 2012 to receive an Electronics and Telecommunications B.Sc. degree and a Master of Science degree in Telecommunications Engineering, respectively. He achieved a Doctoral degree in the Computer and Network Engineering department at the University of Electro-Communications (UEC), Tokyo, Japan, in March 2023. He is also a Lecturer at the Academy of Cryptography Techniques,



learning, and IoT security.

Manh-Thang Nguyen received a B.S. degree in Applied Mathematics and Computer Science from Lipetsk State Pedagogical University, Lipetsk, Russia, in 2013, and a Ph.D. degree in Information Security from the Academy FSO, Oryol, Russia, in 2020, respectively. He is currently a Lecturer in the Department of Information Security at the Academy of Cryptography Techniques (ACT), Hanoi, Vietnam. His research interests include network security, database security, machine



Van-Phuc Dang received a B.S. degree in Electronics and Telecommunications from the Academy of Cryptography Techniques (ACT) in 2025. His research interests include mixed analog/digital design.



Thanh-Nam Pham received a Ph.D. degree in Information Engineering and Computer Science from Feng Chia University, Taiwan, in 2018. He is currently with the Department of Electronics and Communications Technology at Thai Nguyen University of Information and Communication Technology, Vietnam. His research interests include Internet of Things technologies and applications.