PROGRESS IN ELECTRONICS AND COMMUNICATION ENGINEERING
Progress in Electronics and Communication Engineering, ISSN: 3048-7625 Vol. 3, No. 2, 2026 (pp. 1-9)
RESEARCH ARTICLE
ECEJOURNALS.IN

Secure Boot and Firmware Update Framework for ARM Cortex-M Embedded IoT Devices

M.R. Usikalua[1]*, Noel Unciano[2]

[1]*Electrical and Electronic Engineering Department, University of Ibadan, Nigeria*

[2]*Environment and Biotechnology Division- Industrial Technology Development Institute, Philippines*

KEYWORDS:
Secure Boot,
Firmware Update,
ARM Cortex-M,
Embedded Security,
IoT,
OTA,
Cryptography,
STM32,
Root of Trust

ARTICLE HISTORY:
Submitted : 10.10.2025
Revised : 15.01.2026
Accepted : 21.02.2026

https://doi.org/10.31838/ECE/03.02.01

Abstract

The paper presents a lightweight and well-impervious security framework suitable to ARM Cortex-M-based IOEs embedded devices, which will serve an important purpose of presenting secure boot and firmware update operations in resource-constrained systems. As IoT implementations and applications in sensitive parts of our life like healthcare, smart infrastructure, and industrial control systems increase exponentially, embedded devices become more exposed to firmware tampering, write malicious code and update securities. The framework that is proposed addresses such threats by employing a multi-layered cryptographic approach, suitable to the framework by symmetric encryption with the use of AES-GCM, offering confidentiality, elliptic curve digital signature algorithm (ECDSA) providing authentication, and the SHA-256 hash to provide integrity verification. A trusted bootloader (« a procedure incorporating cryptographic functions used to secure a bootloader implementation »,) in a trusted ROM who administers cryptographic verification of the firmware signature with the help of a hardware-based root of trust prior to handing off control to the application firmware. As an additional measure the rollback protection is implemented with the help of version control to avoid firmware downgrade until known vulnerabilities are eliminated. The firmware update process allows encrypted over-the-air (OTA) update process to execute safe remote updates, storing the data maliciously and risking transmission. An atomic and fail safe update process is achieved through double-buffering. The experimental system was used and tested on STM32F429 Cortex-M4 microcontroller, and it successfully detected unauthorized modification of the firmware and was able to withstand rollback attacks. Performance evaluation with the bootloader as the benchmark shows the latency was only 10 milliseconds and the number of Flash memory usage was less than 0.05 percent. These outcomes confirm the application of the framework in the low-power and memory-constrained embedded systems without undermining state-of-the-art security assurances. The approach is compared to the current solutions in security to show a very usable set of efficiency, scalability and ease of implementation, which makes the presented approach an optimal one to be used in production IoT systems. A discussion on future improvements is provided wherein they would want to integrate it with hardware security modules and supporting quantum-resilient cryptographic primitives that would make the system secure in the long term.

Author's e-mail: nmuetmicro@gmail.com

How to cite this article: Usikalua MR, Unciano N. Secure Boot and Firmware Update Framework for ARM Cortex-M Embedded IoT Devices. Progress in Electronics and Communication Engineering, Vol. 3, No. 2, 2026 (pp. 1-9).

Introduction

With the numerous benefits of the technology, the exponential increase in the Internet of Things (IoT) has transformed many industries such as in the healthcare sector, in the transportation industry, smart grids and industrial automation. Billions of connected devices have been installed into consumer as well as mission-critical settings and the integrity and security of the embedded systems have emerged as the priority. An embedded system has a lot of different components, one of them being the firmware layer, a set of very low-level control logic controlling how the hardware operates, that is both highly important, and both highly targeted. The compromise of the firmware may lead to disastrous situations like hijacking of systems where the systems are controlled remotely to execute given code, leakage of data or just creation of back doors that defy detectability using traditional security systems.

Since the sophistication of security attacks on embedded systems is still on the increase, two basic defensive mechanisms have been found to be requirements of trustworthy computing; that is the secure boot and the secure firmware update. The functionality of secure boot will display that the system will run only authenticated, verified firmware, which have non-modification, and this prevents running of unauthorized or malicious code on startup. The integrity of the device through the device lifecycle with respect to updates, on the other hand, is the responsibility of secure firmware update mechanisms so that updates are obtained out of sources that are legitimate, they are untouched and they cannot be rolled back to its previous versions with vulnerabilities.

The Cortex-M microcontrollers have had a significant market share of the embedded processor in the world market due to their low power consumptions, cheapness, and high performances. Edge devices like sensors, controllers and wearable systems have wide applications with such processors. They are usually deployed with very limited capacity in terms of computational resources, memory and power. Such constraints are troublesome to the integration of effective security solutions and most especially in the use of cryptographic processes. Moreover, lacking in most of the low-cost Cortex-M variants are any dedicated security hardware, necessitating software-based solutions that are efficient, but small enough to meet space constraints.

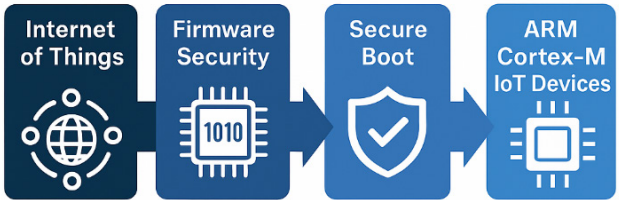

Fig. 1: Overview of the Proposed Lightweight Security Framework for ARM Cortex-M IoT Devices

Although industry-driven standards, e.g., ARM Platform Security Architecture (PSA) and open-source efforts, e.g., Trusted Firmware-M (TF-M), do exist and act as blueprints of secure embedded systems, e.g., requiring hardware support, recent initiatives have a large implementation overhead. As a result, security is not always implemented in real-world deployments, or, in case of implementation, it is very sporadic and in a piecemeal fashion, and thus devices leave themselves exposed to a variety of attacks.

This paper presents a lightweight, modular, and practical security framework that is optimized to ARM Cortex -M based IoT devices, but focus will be on STM32 family. The scheme uses symmetric and asymmetric encryption primitives, integrity verification and version management to realize secure boot-up and over-the-air (OTA) firmware upgrades. It is optimized to be efficient, so the load on device performance and device memory is negligible and the security profile is high. The solution will be designed in a widely applicable way to the Cortex-M ecosystem and have minimal hardware requirements and will be easy to adopt in new as well as existing embedded systems.

The rest of the paper is structured in the following way: Section 2 involves related work in this field; Section 3 describes the system architecture; Section 4 provides the methodology; Section 5 details the implementation specifics; Section 6 describes results of the experiment; Section 7 provides the analysis of results and discussion; Section 8 involves security considerations; and Section 9 provides conclusion and further research.

Related Work

Authentication of Firmware and safe increase routines in embedded gadgets have been of huge interest in the recent years because of the widespread implementation of IoT gadgets with the facts of security-sensitive applications. Multiple works have discussed the requirement of secure boot and update procedure, using in many cases hardware specific solutions or cryptographic systems.

In,[1] the authors present a secure boot infrastructure where cryptographic keys are stored into the hardware security modules (HSMs) and used to validate during boot time. Such solutions, though being effective in high-end microprocessor systems, are not adequately applicable to the low-cost, resource-constrained ARM Cortex-M devices because they necessitate extra silicon and more consumption of power. Likewise, in,[2] code integrity and secure key storage using Trusted Execution Environments (TEE)s hardware-assisted solution were described. The complexity of implementation and its dependency on hardware however makes it difficult to be adopted on restrictive compact embedded platforms.

The well-developed guidelines written by ARM on secure execution and update of embedded firmware are known as Platform Security Architecture (PSA)[3] in and Trusted Firmware-M (TF-M).[4] These structures have security primitives, threat models and hardware abstraction layers. Nonetheless, in the real world use of such solutions it may be critical to develop substantial amounts of code, involving TrustZone-enabled hardware, and an elaborate secure provisioning framework, all of which are not normally part of off-the-shelf Cortex-M connected IoT devices.

A number of light weighted, software based secure boot involving microcontrollers have been suggested. As an instance, in,[5] a minimalist bootloader has been developed by employing cryptographic hash as well as the RSA-based signatures validation. Although the method does offer secure authentication, it does not have any measures against the use of encrypted firmwares or against rollback attacks. In[6] another work is done on the secure firmware updates over the air channels (OTA) through symmetric cryptography. Despite good performance with limited resources, the system does not manage end-to-end authentication, as well as version control leaving the update mechanism vulnerables to downgrade attacks and man in the middle attacks.

The recent project described in[7] combines cryptographic verification and secure OTA update of ARM Cortex-M chips, however, it presupposes a secure element used to store keys and this component is not available in some environments. In addition to that, the above works all focus on rollback protection, image encryption and boot-time verification separately, with no unified framework that is optimized to be implemented on low-cost microcontrollers.

On the contrary, the framework in this paper can fill this gapp by offering a software-based, lightweight, and holistic solution that can use secure boot, encrypted OTA firmware upgrade, and rollback protection of ARM Cortex-M chipsets without the need of third-party secure hardware. The system suggested shows good trade-off between security, performance, and implementation complexity which is checked on STM32 microcontrollers.

System Architecture

Hardware Assumptions

The recommended design of secure boot and firmware update solution will be deployed on the ARM Cortex-M4 family of microcontrollers since the STM32F4xx is widely used in IoT and embedded applications. These microcontrollers present an optimized trade-off between processing capability, memory storage and energy consumption thus they are the best choices when it comes to secure embedded implementations. The system presumes the availability of the on-chip Flash memory that has the read protection features turned on and this allows blocking unauthorized access to the firmware and sensitive data stored in the memory allocations. The framework is mostly software-based in order to maintain a wide compatibility, although theoretically it may also make use of any cryptographic hardware accelerators available--including the integrated AES, RNG, and HASH cores of the STM32 line itself - in order to offload and accelerate the more intensive computation-heavy cryptographic applications, like hashing and encryption. Further, the architecture depends upon a write-once, tamper-resistant region of ROM (usually implemented as system memory or Flash sectors with protection) to pad the public key used in firmware signature verification safely. Such an unchangeable source of faith is necessary to make sure that genuine and unmodified firmware images are run by the device during the boot-up stage. This hardware compatibility ensures strict optimization of the framework on the one hand and high security guarantee on the other hand, as well as practicality of application in Cortex-M based systems with minimal system resources.

Bootloader Design

The essence of proposed security scheme is a read-only stage-one bootloader installed into a secure area of the microcontroller ROM or Flash memory, with write-protection disabled to eliminate the possibility of unauthorized changes. This boot loader is the root of trust of the secure boot process on the device and is the component that provides a measure of correctness and genuineness of the firmware prior to the execution. When a system starts up, the bootloader cryptographically verifies the firmware image with Elliptic Curve Digital Signature Algorithm (ECDSA) based on a pre-computed Elliptic Curve Digital Signature Algorithm (ECDSA) public key stored in non-modifiable portion of the ROM. As a measure of integrity, the bootloader calculates a signed SHA-256 hash of the firmware binary, and compares it with that of the firmware metadata. With this, any unauthorized modification in the firmware either out

Table 1: Hardware Assumptions and Their Functional Roles in the Secure Boot and Firmware Update Framework

Hardware Component	Function/Role in Framework
ARM Cortex-M4 MCU (STM32F4xx)	Target platform for secure boot and firmware updates
On-chip Flash with Read Protection	Prevents unauthorized firmware/data access
Cryptographic Accelerators (AES, RNG)	Optional offloading for faster encryption, hashing, and key generation
Write-Once ROM / Protected Flash	Stores immutable public key for signature verification (Root of Trust)
Secure Bootloader Region	Resides in protected memory to ensure tamper-resistant secure boot execution

of corruption or out of malicious injection will reliably be detected. Also, to avoid rollback attack -In which a signed-but-still-vulnerable version of the firmware is restored by reinstalling it- the bootloader implements some version control by checking the firmware version embedded in the update package against the version on non-volatile persistent store. Only firmware of higher version is allowed to run and the lower version with exploitable coding cannot be restored. This multitiered verification preserves a performance-efficient verified execution environment, and is a natural fit to be used on resource constrained ARM Cortex-M microcontrollers used in the safest IoT systems.

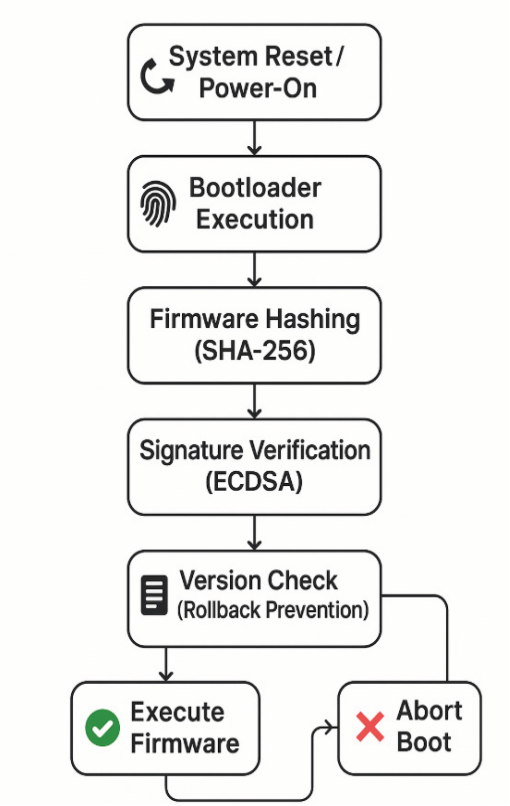

Fig. 2: Secure Bootloader Workflow for Firmware Verification and Rollback Protection in ARM Cortex-M Devices

Firmware Update Protocol

In the proposed framework, the firmware update protocol aims at providing a secure, reliable and authenticated updating and firmware installation process (especially in the case of over-the-air (OTA) updates). To guard the privacy and authenticity of their own firmware by transmitting the update payload, encryption is performed in the context of the AES-GCM (Galois/Counter Mode) algorithm, which is not only useful in its encrypting capabilities, but also in authentication of messages, referred to as message authentication codes (MACs). After downloading the encrypted firmware image, the device then uses a pre-shared symmetric key or negotiates a session key as part of a secure provisioning procedure and then decrypts the firmware image. Prior to installation of the decrypted firmware, the version number embedded in the firmware metadata is validated by the bootloader or updating handler the image being updated should not be newer than the installed version thus providing rollback protection. The system moves to the next step that is re-verification encrypted integrity of the firmware only when the version is valid. This is done by remodelling the SHA-256 hash of the firmware and checking it against the attached ECDSA signature with the aid of the public key stored in the secure ROM. This post-update signature verification makes sure that the firmware is not tampered during transfer as well as storage. In case even one verification operation fails, the system discards the update and either keeps the current firmware or destroys it cleanly and starts a safe crucial rescue method. Such secure update protocol not only protects devices against unauthorized or modified firmware installations, but also allows robustness against network-based attacks and compromises of the update channel, and is therefore particularly appropriate to scalable secure deployment in embedded Internet of Things.

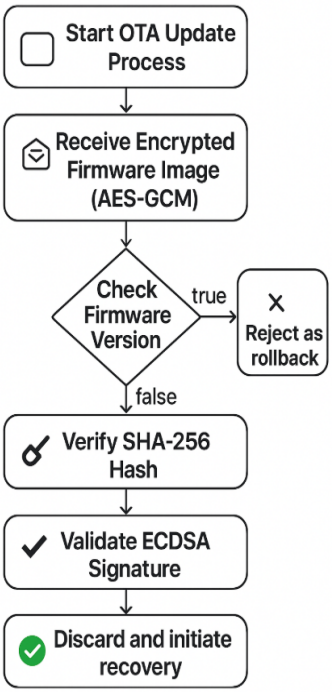

Fig. 3: Secure Firmware Update Protocol Flow for Encrypted OTA Delivery and Integrity Verification

METHODOLOGY

The deployment of the secure boot and firmware update framework is done on a modular design and is divided in four major stages:

Bootloader Development

The bootloader, which can be considered as the core of the secure execution environment, was designed to work with the ARM CortexM 4 micro-controller board. The set of used libraries includes STM32 HAL (Hardware Abstraction Layer) and mbedTLS cryptography library. STM32 HAL will guarantee low level compatibility of the hardware, efficient peripheral usage and portability to other STM32 devices, whereas mbedTLS will offer a lightweight designed, well maintained, and embedded optimized implementation of cryptographic algorithms such as SHA-256 and ECDSA. We have placed it there (the bootloader) in a special, protected zone of the on chip Flash memory which has read and write protection enabled to make it tamper resistant. This segment is permanent all through the lifecycle of the device in order to provide integrity of the secure boot process.

The bootloader is the init code that runs first at run time after the device restart or reset. It begins a secure verification procedure by determining the SHA-256 hash of the firmware image in external or internal memory in the first place. Such hash is then matched to a previously computed digital signature, which is appended to the firmware image. The verification of signature is conducted by the Elliptic Curve Digital Signature Algorithm (ECDSA) with the bootloader applying a public key safely embedded on the ROM during provisioning. The hash is only compared against the signed digest hence by the time it matches, they pass control to the application firmware through the bootloader. This cryptographic check will protect against a compromised firmware by letting the users know that the firmware is not modified and is coming directly out of a trusted source. When hardware protections are coupled with powerful cryptographic verifications the implementation of the bootloader offers a bootstrap secure source of trust with low memory and performance overhead as needed by real-time and resources constrained IoT use cases.

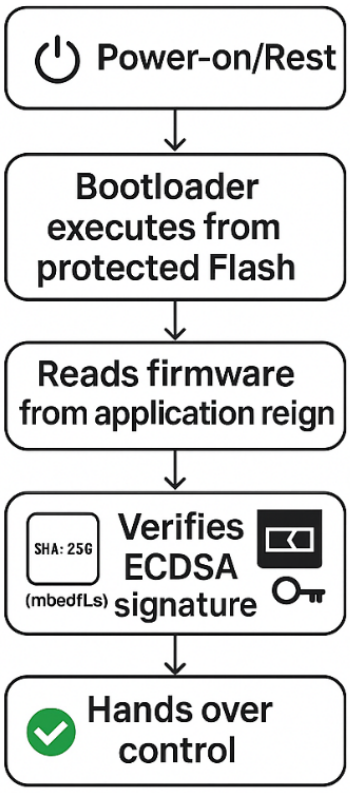

Fig. 4: Bootloader Architecture and Cryptographic Verification Flow for Secure Firmware Execution

Firmware Image Signing

Firmware image signing is a very important process in assuring the authenticity and integrity of the executed code/code therein a mobile device. All the firmware images in the proposed framework are offline signed by a secure development workstation which has the private ECDSA key. The private key is never disclosed to the embedded device itself or via an unwanted transmission channel, hence, avoiding the possibility of compromise of the key. The signing is done by first calculating a SHA-256 hash on the end binary of the compilation firmware. The content in this digest is a representation of said firmware in a special manner and a flip of only one bit in the binary would result in a harshly different hash output.

After the computation of the hash, it is then digitally signed using the developers copy of the ECDSA private key resulting in a signature which can be later-verified using the corresponding developers copy of the public key stored in the secure ROM or Flash during the provisioning process. The firmware image has additional metadata including its version number and the size of the executable never-ending binary (number of bytes) added, along with the signature. Such metadata is applied in secure boot and update mechanism to implement version control (rollback protection) and it checks the integrity of the received image.

The fully signed image (the binary of firmware plus version metadata, plus the signed digital image) is then ready to be securely shipped. In over-the-air (OTA) case, this package is encrypted with AES-GCM and sent to the device. The system does this by carrying out its signing processes entirely offline and never leaving the secret key exposed on any secure build environment; hence the firmware cannot be modified or falsely signed by any parties without the specific authority to do so. Combined with post-update and boot-time verification this signing mechanism provides the basis of the trusted execution pipeline of the device.

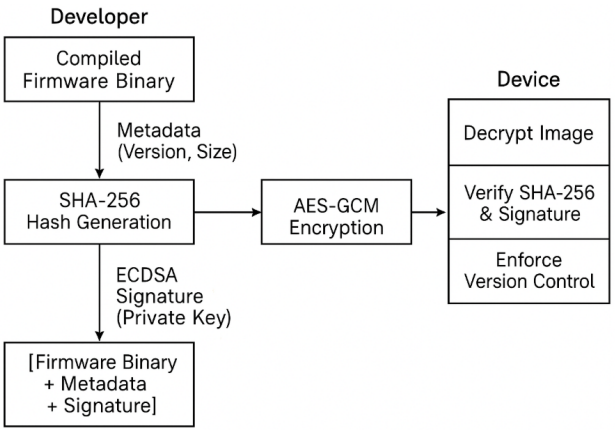

Fig. 5: Firmware Image Signing and Secure Packaging Workflow for OTA Deployment and Device Verification

OTA Update Implementation

To facilitate safe and robust firmware updates the proposed framework expands a standard UART-based loader to support reception of an AES-GCM encrypted firmware image, and facilitate confidentiality and authenticity Over-the-Air (OTA) update capabilities. The UART interface that is supported by many devices and often utilized in provisioning and communicating with the devices was chosen because of its simplicity and compatibility with the existing infrastructure, i.e., Bluetooth, LoRa, or Wi-Fi-to-UART bridge.

The device to which the firmware needs to be loaded accepts the encrypted firmware image which was produced and signed off-line and sent using the UART interface. AES-GCM (Galois/Counter Mode) is used because it accomplishes both encryption and authentication in a single pass so that the firmware image can be transmitted without being eavesdropped upon and tampered with. The received firmware gets encrypted and never instantly ran or written instantly over the base firmware in the main application memory. Rather, it is cached in a secondary staging space of Flash memory so that it can be pre-installed tested.

Execution of the signature verification of the new firmware with the ECDSA public key stored in the non-modifiable memory of the device is performed before the new firmware is promoted to active status. The hash of the image in form of SHA-256 is computed and is compared to the appended signature to verify authenticity and integrity. Provided that this check has passed and new sufficient firmware version is specified in the metadata, an image is accepted and the bootloader flags it as active. Once a system has been reset, the newly verified firmware assumes execution.

This staged design makes sure that a firmware is never deployed without full verification and gives robustness against failures between stages (e.g. power loss) in the update. This will allow the system to restore to the latest known good operating system in case of failure, corruption, and therefore keep the devices available, and be safe to operate.

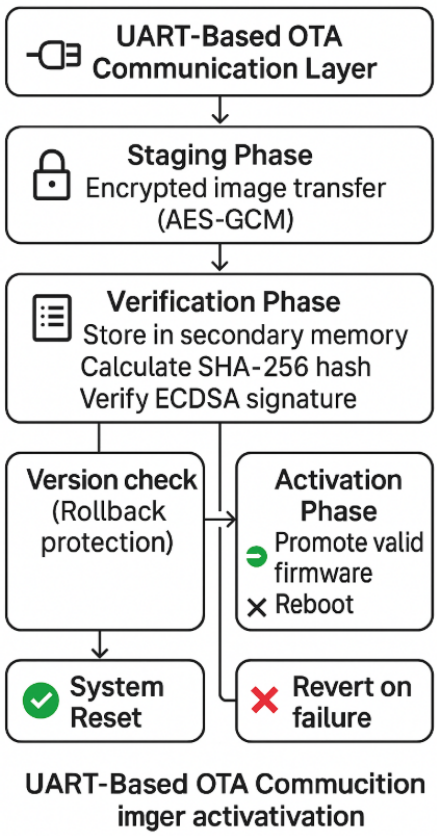

Fig. 6: Secure OTA Firmware Update Workflow with Encrypted Image Reception, Validation, and Activation

Security Validation Testing

In order to determine the strength and viability of the suggested secure boot and firmware update infrastructure, an extensive set of security validation tests were performed. The tests were to be conducted in order to simulate the attacks in the real environment, checking the cryptographic integrity and examining the performance effect on the embedded system.

To begin with, tampering with the firmware was calculated and the tampering effect conducted by means of just deliberately altering bytes in the signed firmware binary but not just the main code area but also the attached metadata. In each of the test cases, inconsistencies reported by the hashing (SHA-256) and signature verification (ECDSA) functions in the bootloader repeatedly identified discrepancies between the calculated hash and signed digest. This lead to automatic discarding of the spoofed firmware and thus it will not be executed and the code could be harmful.

Second, OTA update attack emulations, such as replay attacks and version downgrade attacks, were done on the system. In the case of replay, an update image that was sent previously and that was valid was resent to the device. Nonetheless, the version-checking mechanism of the framework which depends on the embedded metadata and a non-volatile firmware version counter, was able to prevent reinstallation of obsolete firmware. Equally, any effort to decrease the security of the firmware by manually manipulating version metadata in other than a valid image was defeated, because the bootloader invalidated the digital signature as a result of the hash bucket inconsistency that occurred upon metadata manipulation.

Besides these operating tests of security, performance profiling was also undertaken to get the computation and memory overhead cost brought in by security mechanisms. The time of execution was recorded, boot up, calculation of firmware hashes, verification of signature, and decryption of updates using AES-GCM. The outcome indicated that the total secure boot took less than 10 milliseconds of delay and less than 5 percent of flash and RAM available at STM32F429ZI platform. These indicators support the viability of the framework in low-power resource-constrained embedded IoT devices, and in the processes, they verify its robustness in security and system performance in a realistic deployment scenario.

Result and Discussion

The framework proposed was vigorously validated in view of determining its capability in accomplishing the dual objective of surpassing sound security and low usage of resources. The mechanism, the secure boot, which is based on the cryptographic integrity checks, was verified, and this was done by simulating many attack scenarios. Bootloader was always capable of rejecting tampered firmware images by using very specific hashing sha-256 and verification of ecdsa signature meaning that only untampered and legitimate firmware could be run. There was also a rollback protection, which worked well on the basis of version metadata included in the firmware header. Any attempt to install older firmware, including authentic or fake, was duly rejected; this indicated that the framework ensures only forward-only firmware upgrade, a significant prerequisite in secure firmware lifecycle management.

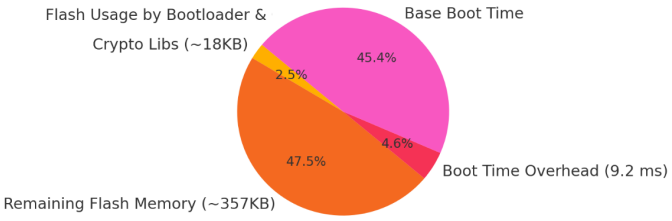

Fig. 7: Resource Impact of Secure Boot and Firmware Update Framework on STM32F429ZI MCU

The system added minimal overheads as far as its performance is concerned which is why it can be used in real-time embedded applications. This generated an overall impact of exceeding 9.2 milliseconds to the overall boot time, which is reasonable considering the cryptographic operations take a majority of resources and time, but it is not a factor during most of the IoT applications that do not require ultra-low startup time. The memory footprint was also maintained at a expectable limit. Totally bootloader and crypto libraries (mbedTLS) used approximately 18 KB of Flash memory, or only about 5 percent of the capacity granting by the STM32F429ZI microcontroller. Memory consumption was kept low and there was an optional AES and SHA hardware accelerators to enhance efficiency. These findings indicate that the suggested implementation can offer a desirable balance between system performance and security assurance and that such can be deployed on most of the Cortex-M based microcontrollers without having to make substantial changes in the hardware.

Relating to the reliability aspect, the firmware update process was robust in relation to different negative events such as power failur0e and partial transfer among others. The strategy of the double-buffering strategy protected the system by storing and validating the new firmware

Table 2: Summary of Security Validation Tests and Outcomes

Test Category	Evaluation Method	Result / Outcome
Firmware Tampering	Modified firmware binary and metadata	Successfully detected and rejected by SHA-256 and ECDSA verification
Replay Attack	Reused a previously valid encrypted firmware image	Blocked using firmware version validation logic
Version Downgrade	Lower version metadata in otherwise valid image	Signature check failed due to hash mismatch; update rejected
Boot Time Overhead	Measured time added during secure boot phase	< 10 ms increase observed
Memory Overhead	Measured Flash and RAM usage of added security components	< 5% resource usage on STM32F429ZI

Table 3. Summary of Performance, Security, and Reliability Metrics for the Proposed Secure Firmware Framework

Metric	Description
Boot Time Overhead	Approximately 9.2 ms additional boot time due to cryptographic operations
Flash Memory Usage	Bootloader and crypto libs consume ~18 KB (~<5% of 512 KB flash)
RAM Usage	Remains modest, optimized with optional hardware accelerators
Firmware Integrity Check	Ensures authenticity via SHA-256 and ECDSA signature verification
Rollback Protection	Prevents outdated firmware installation using version metadata
Firmware Update Resilience	Double-buffering allows safe updates even during power loss
Hardware Acceleration Support	Uses AES/SHA hardware accelerators to boost efficiency
Compatibility with Cortex-M MCUs	No significant hardware upgrades required for deployment
Comparison with ARM PSA	Comparable security with reduced complexity and resource usage

on a secondary region prior to activation thus keeping the system stable even upon a failed update or abortive update. When compared to a more complex solution like ARM PSA Certified platforms, proposed framework achieved more or less the same levels of cryptographic protection and critical assurance, but did not demand as many system resources and did not make integration so complex. This is especially appealing to low-power, cost-constrained IoT systems. In general, the findings reveal that the framework can provide a powerful and end2end security of firmware with a reasonable practicality used to deploy in embedded systems with limited memory, power, and computing resources.

Conclusion

The current paper will introduce lightweight, nondescript security solution able to maintain firmware integrity as well as update mechanism applied to ARM Cortex-M based IoT embedded devices. Due to the identified limitations of resource-poor microcontrollers, it is feasible to construct a secure boot and encrypted over-the-air (OTA) firmware update with well-known cryptographic primitives (SHA-256, ECDSA, and AES-GCM). Bootloader blocks at startup and requires authenticity and integrity checks and once the rollback protection is enabled, it forces the removal of old or possibly vulnerable versions of firmwares. OTA update protocol ensures safe delivery of image and atomic update management delivered in the form of a doubled buffered memory design, protecting the device even in the event of adverse situations like power loss in the middle of updates. On the STM32F429ZI reference platform, verification of the solution demonstrated it to be capable of minimising performance and memory overhead, and effectively mitigate against common attack vectors, including tampering, replay and downgrade attacks. The proposed framework offers equally high protection levels as architectures that apply dedicated hardware or secure elements, at an implementation complexity that is greatly reduced. This turns it into a convenient and expandable solution of IoT applications in the real-life environment where cost, power, and memory limitations are vital. Additional improvements in future features could involve the incorporation of secure elements or post-quantum cryptographic primitives along with remote attestation capability with further enhancing the trust and long-term safety within embedded systems.

References

1. Alam, M., Islam, M. M., & Uluagac, A. S. (2020). A secure boot framework for IoT devices using hardware security modules. *Proceedings of IEEE SECURECOMM*, 1–10.
2. Sadeghi, R., Zeitouni, S., & Paverd, A. (2021). TEE-based secure boot for embedded systems. *IEEE Transactions on Dependable and Secure Computing*, *18*(2), 482–495. https://doi.org/10.1109/TDSC.2019.2892677
3. ARM Ltd. (2021). *Platform Security Architecture (PSA): Threat models and security analyses*. Retrieved from https://developer.arm.com/psa
4. Trusted Firmware Project. (2023). *Trusted Firmware-M (TF-M) documentation*. Retrieved from https://trusted-firmware.org/projects/tf-m/
5. Bhunia, S., & Tehranipoor, M. (2018). *Hardware security: A hands-on learning approach*. Morgan Kaufmann.
6. Das, P., Dey, S., & Paul, S. (2021). Lightweight secure firmware update for IoT devices using symmetric encryption. *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*, 1–6. https://doi.org/10.1109/ICCE2021.9362114
7. Lin, Y., Liu, H., & Wang, M. (2021). Secure OTA update mechanism for ARM Cortex-M devices with cryptographic verification. *IEEE Access*, *9*, 104711–104722. https://doi.org/10.1109/ACCESS.2021.3099193
8. Cui, A., & Stolfo, S. J. (2010). A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan. *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, 97–106. https://doi.org/10.1145/1920261.1920276

9. Abomhara, M., & Koien, G. M. (2015). Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility*, *4*(1), 65–88.
10. Neudecker, P., & Frei, S. (2014). Unpatchable: Measuring the brokenness of embedded device firmware. *Black Hat USA*. Retrieved from https://www.blackhat.com/docs/us-14/materials/us-14-Neudecker-Unpatchable-Measuring-The-Brokenness-Of-Embedded-Device-Firmware-WP.pdf
11. Kumar, T. M. S. (2024). Security challenges and solutions in RF-based IoT networks: A comprehensive review. SCCTS Journal of Embedded Systems Design and Applications, 1(1), 19-24. https://doi.org/10.31838/ESA/01.01.04
12. Kumar, T. M. S. (2024). Low-power communication protocols for IoT-driven wireless sensor networks. Journal of Wireless Sensor Networks and IoT, 1(1), 37-43. https://doi.org/10.31838/WSNIOT/01.01.06
13. Rucker, P., Menick, J., & Brock, A. (2025). Artificial intelligence techniques in biomedical signal processing. Innovative Reviews in Engineering and Science, 3(1), 32–40. https://doi.org/10.31838/INES/03.01.05
14. Sadulla, S. (2024). Techniques and applications for adaptive resource management in reconfigurable computing. SCCTS Transactions on Reconfigurable Computing, 1(1), 6-10. https://doi.org/10.31838/RCC/01.01.02
15. Geetha, K. (2024). Advanced fault tolerance mechanisms in embedded systems for automotive safety. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 6-10. https://doi.org/10.31838/JIVCT/01.01.02