SCCTS Journal of Embedded Systems Design and Applications, ISSN: 3048-8753 Vol. 3, No. 2, 2026 (pp. 16-25)
RESEARCH ARTICLE ECEJOURNALS.IN

Secure Over-the-Air (OTA) Firmware Update Mechanism for Resource-Constrained Embedded Devices

Raveendra H Patil[1]*, Hartwig Henry Hochmair[2]
[1]Agricultural & Biological Engineering Department, University of Florida, USA
[2]University of Florida, Geomatics Program, USA

Keywords:
Secure OTA Update,
Firmware Integrity,
Embedded Security,
ARM Cortex-M,
AES-GCM, ECDSA,
LoRaWAN,
Anti-Rollback,
Resource-Constrained Devices

Author's Email:
ravi.patil@ufl.edu,
hhhochmair@ufl.edu

https://doi.org/10.31838/ESA/03.02.03

Received : 16.01.2026
Revised : 10.02.2026
Accepted : 05.03.2026

Abstract

Software embedded design systems are also penetrating more and more important areas of application: the Internet of Things (IoT), remote health monitoring, industrial automation, and smart infrastructure, current safe efficient and secure firmware update mechanisms are paramount. The ability associated with over-the-air (OTA) updating of firmware is strictly required in preserving the functioning of devices, exploiting such gaps, and empowering the features after release. But in highly constrained systems (those with limited memory, processing capacity and available energy), conventional secure update interfaces can incur unacceptable overheads or they may not provide complete protection against sophisticated cyber-attacks. In this paper the authors introduce a simple yet lightweight and robust OTA update mechanism built specifically to be used on such constrained platforms. This suggested framework combines the familiar AES-GCM-authenticated encryption to guarantee data confidentiality and integrity, a source authentication technique based on checking digital signatures secured using ECDSA, and a versioning protocol providing the anti-rollback protection using monotonic counters. As part of validating its practicality, the solution is applied and tested with ARM Cortex-M microcontrollers, i.e., STM32 and nRF52840, where communication is performed over low-power wireless networks like LoRaWAN and BLE. A signature of firmware packages via custom-built update server (TLS) will provide secure transfer of updated packages. Experimental scrutinisms indicate that the secure OTA system has very limited overhead in resources (about 14 KB extra flash usage and 3 KB of RAM) and the average update speed is 3.8 seconds with a 256 KB binary image using LoRaWAN. In addition, the system is well adapted to identify and discard a tainted or old firmware image, hence ensuring device immunity against attacks including firmware manipulation, replay, and man-in-the-middle intrusion. The performance highly conforms to the contention that the framework provides a robust degree of security and performance coupled with resource usability hence being ideal to use in real fully embedded applications that have strict limitations. This contribution consists of a scalable, secure OTA approach that builds resilience to the operational integrity and trustworthiness of the distributed embedded system in dynamic and security critical settings.

How to cite this article: Patil RH, Hochmair HH (2026). Secure Over-the-Air (OTA) Firmware Update Mechanism for Resource-Constrained Embedded Devices. SCCTS Journal of Embedded Systems Design and Applications, Vol. 3, No. 2, 2026, 16-25

Introduction

The spread of embedded systems in contemporary technological ecosystems is revolutionary, especially in Internet of Things (IoT), intelligent healthcare, automation of industries, cars, and environmental management. Being frequently situated in inaccessible or remote locations, such devices have to keep working during lengthy intervals, and a possibility of their firmware upgrade is an important part of their life cycle. Not only do firmware updates increase functionality and performance, they are required to iron out vulnerabilities, guarantee compliance to continually developing standards and they also ensure security in comparison to new threats.

Nevertheless, the firmware updates of embedded devices pose several security and operational issues, namely in the case of resource-constrained settings, as devices have low computational capabilities, memory and supply reserves. In these situations, conventional methods of firmware update that can be based on physical access or use of considerable amounts of system resources will not be feasible. Over-the-Air (OTA) firmware update is a scalable and efficient method and a solution as well as it can be used remotely to update any device without the need to perform manual processes. In spite of these virtues, OTA mechanisms create additional sources of cyber-attacks, especially when the update procedure does not offer strong security using measures.

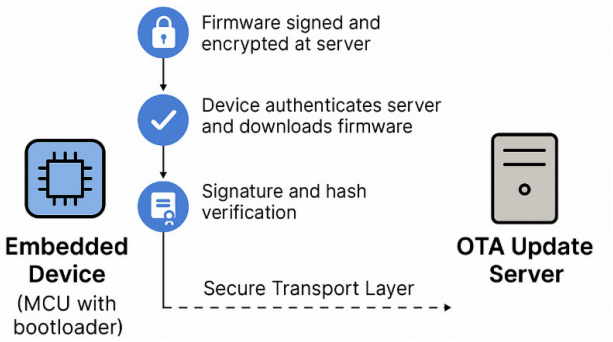

Fig. 1: Secure OTA Firmware Update Workflow

The major security risks related to OTA updates are firmware tampering an adversary may insert some malicious code; rollback attack an adversary can install outdated and vulnerable firmware to use known vulnerabilities; and eavesdropping monitoring the update data that may contain sensitive information. Improperly secured OTA processes may have a devastating effect on the integrity, confidentiality and availability of the embedded system susceptible to disastrous failures in mission-conscious systems.

These challenges demand that new lightweight, secure and verifiable update protocols with the ability to run inside the resource-constrained embedded system envelope are designed. This paper lays out a solution to this problem, a secure OTA firmware update of the low-power embedded devices. The framework also integrates highly secure cryptographic standards, such as AES-GCM to encrypt and authenticate transmission of data, ECDSA based on source verification through signatures and a version control system that guards against firmware downgrade attacks. The introduced solution is adopted on ARM Cortex-M microcontrollers that allow achieving high resource constraint and safety interdependencies.

This work is important since the following contributions are made:

- An OTA firmware update mechanism specifically designed to run atop a resource-constrained embedded system.
- Integration of AES-GCM and ECDSA to make the data confidential, authentic and ensure integrity.
- -Rollback protection realized by version-controlled anti-rollback secure monotonic counters.
- Examination of the proposed framework on physical embedded platforms based on the performance analysis of memory utilization, power comparisons, update latency.

This paper solves the problem of efficiency and security challenges and is thus a practical guide on

implementing a secure firmware update in wide-scale embedded application.

Related Work

Firmware update mechanisms traversed through Over-the-Air (OTA) have been tested extensively in the embedded systems and, in particular, the IoT devices category, where security and functionality of the systems after deployment remain the core concerns. There are numerous OTA frameworks currently represented in existing literature, but they resound with a lot of challenges, especially in resource-limited settings.

In,[1] the setup of a generic OTA update framework of wireless sensor networks (WSNs) has been proposed focusing on modularity and scalability. Nonetheless, cryptographic protection was not used, leaving the devices vulnerable to tampering, as well as injection attacks. In a similar vein, OTA technique recommended by Rahman et al.[2] was lightweight, allowing transmissions of compressed firmware images to low-power IoT nodes in order to reduce the overhead. This method was effective, but it was very vulnerable to replay attacks, as strong authentication was not represented by symmetric encryption.

Recent developments have brought safeguarded OTA practices including cryptographic components. To illustrate, Zhang et al.[3] proposed the use of blockchain in converting an existing firmware update system to gain trusted firmware provenance. The method has good tamper resistance and hence overheads and is inappropriate in devices that have ultra-low power consumption. Similarly, Armknecht et al.[4] suggested a formal model of secure firmware distribution using the principle of a digital signature and public key infrastructures (PKI). Besides its excellent security features, due to memory and computation intensiveness, it is less applicable to a limited microcontroller.

Some of the latest studies have tried to secure firmware updates in embedded devices by having transport-layer protocols like TLS and DTLS[5-7] to offer new encrypted channels of communications during the updating activities. Nevertheless, such protocols demand a lot of computation and memory due to which they are not applicable in cases where devices are having constrained RAM and processing power. Method like differential update techniques[8] emphasizes on the minimized size of the firmware image to save bandwidth but many times low end security features like encryption, authentication, and rollback protection, are ignored.

An in-depth survey of the deployment environment of current OTA mechanisms shows a few consistent weaknesses: significant usage of resources, because of complicated cryptographic protocols, use of symmetric-only secure protocol with a weak authentication mechanism, susceptibility to downgrade attack when no version control exists, and lack of flexibility when working with ultra-low power microcontrollers such as Arm Cortex-M0 and M3, etc. Such shortcomings make it clear that an OTA framework that will satisfy both security requirement and maintenance of low system load with a minimal weight is urgently needed. A sentence to that effect is: to this end, this study presents a sensitive wide-ranging OTA scheme that provides end-to-end confidentiality, integrity acumen, authenticity, and rollback resistance at the same time, preserving compatibility with the buy-upon embedded construct.

System Architecture

Components:

The same proposed secure mechanism of the Over-the-Air (OTA) firmware update is designed based on 3 major components: the embedded device, the OTA firmware update server and a secure transport to offer reliable communications between them. The embedded device (usually an IOT-constrained microcontroller unit (MCU) e.g. ARM Cortex-M0/M3/M33 with secure bootloader can be used to verify authenticity and integrity of a firmware prior to execution. This bootloader is the system root of trust which does signature checking as well as version checks in the course of updating. OTA update server is a central point that creates, sign, encrypts and delivers firmware update to intended devices. It stores firmware repositories in versioned and, with the help of cryptographic libraries, signs each firmware image using ECDSA in a way that only legal updates using confirmed sources will be accepted by the devices. The firmware is also bundled with the required metadata along with the server (including version numbers, timestamps, and cryptographic hashes) that the embedded system again verifies.

The framework uses a secure transport layer to guarantee firmware confidentiality and integrity on the network, and uses lightweight but secure protocols, e.g. MQTT over TLS or CoAP over DTLS. It selects these protocols because they are efficient and work well with limited environments to enable them to communicate in an encrypted manner without the major performance penalty. These three components combined allow an end-to-end secure OTA update experience that is provable in a holistic manner, scalable, and secure against a wide range of common attack vectors, such as man-in-the-middle attacks, firmware tampering and rollback attacks.

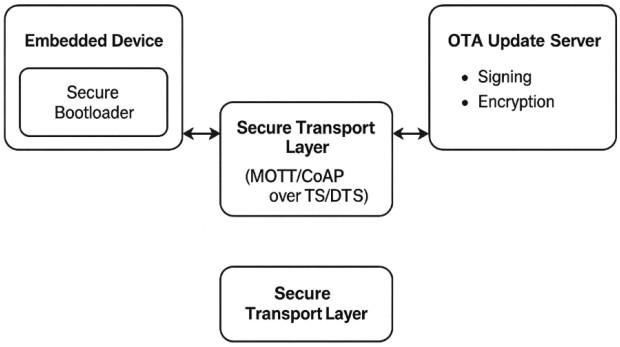

Fig. 2: System Architecture for Secure OTA Firmware Update

Firmware Image Format:

Firmware image format provided by the proposed OTA update framework carefully develops to guarantee a secure, provable, and efficient application on the resource-constrained embedded devices. It is divided into three parts namely header, encrypted payload and a signature. The most critical of the metadata needed to validate the update and gain control thereof is the firmware version number, which is necessary to enforce anti-rollback protection; a timestamp, helping specify how recently the update was carried; and a cryptographic hash (SHA-256) of the plaintext firmware, to verify the integrity of the decrypted payload prior to install. The actual firmware binary is then stored encrypted with AES-GCM (Advanced Encryption Standard in Galois/Counter Mode), a more modern authenticated encryption algorithm secure at the same time against both privacy and integrity attacks and with minimal encrypted data overhead. The features of AES-GCM to produce authentication tags allow identifying any tampering at transmission or storage levels. The third element is a signature scheme ECDSA (Elliptic Curve Digital Signature Algorithm), in particular, the P-256 curve, with a high-security factor, but low computation demands, thereby a perfect signature scheme to be used on embedded systems. The update server creates this signature with its private key and secures the header and the encrypted payload so that modification of any kind could be reliably identified. Once the firmware is received, the embedded device validates the ECDSA signature with the public key and the hash (SHA-256) of the decrypted payload compares with the value in the header. Such a multi-layered structure ensures the installation of only genuine and tamper-free firmware updates, which is strongly resistant to injection, replay, and downgrade attacks, and resource consumption does not exceed economical overheads of limited resources microcontroller systems.

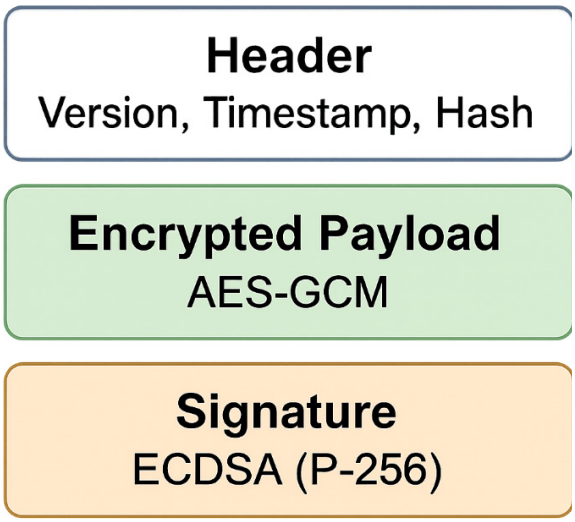

Fig. 3: Structured Format of a Secure OTA Firmware Image

Workflow:

The OTA - secure firmware update design philosophy supports the path towards end-to-end authenticity, integrity, confidentiality lifecycle of firmware update. This starts at the OTA update server, which sends the firmware image after it has been first digitally. signed utilizing the personal ECDSA (P-256) key of the server to ensure authenticity, and after that, encrypted by means of AES-GCM, which introduces both secrecy and stability assurance. Such signed and encrypted firmware is included with a metadata header that

carries the version number, the current timestamp and the SHA- 256 hash of the original firmware. When a firmware package is ready it is published through a secure end point on MQTT/TLS or CoAP/DTLS. On the embedded device end the bootloader or firmware update agent establishes a secured communication with the server via TLS certificates to avoid a man-in-the-middle attack or a rogue update source. After the authentication of the server, the device downloads the encrypted firmware package, which is temporarily stored in secure memory. After this is done the device then checks the ECDSA signature with a pre-provisioned public key which is stored in protected memory. Upon a valid signature, the firmware is decrypted with the AES-GCM key and the plain text is hashed and returned to the SHA-256 digest placed in the header to ensure integrity. The system will only proceed to the final step, and that is to upgrade the new firmware safely to flash memory, usually in a special partition reserved to update, when both the signature and hash checks have passed. At the same time, the system increments a monotonic version counter, in tamper-resistant memory (e.g. TrustZone or OTP), atimplies that one cannot rollback to an earlier, possibly vulnerable firmware version. It is a carefully crafted work flow that will install only needed authenticated untampered and most recent firmware to the guaranteed secure embedded systems and hence improve the security status of the embedded systems to face advanced cyber intimidations.

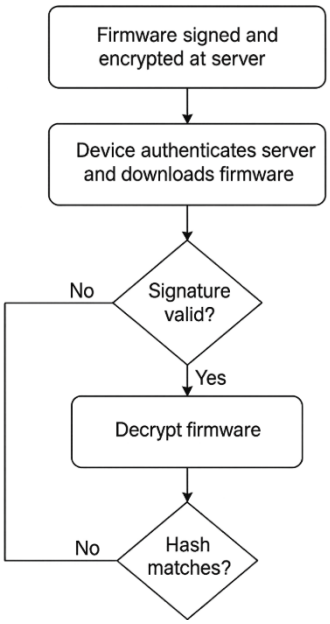

Fig. 4: Secure OTA Firmware Update Workflow

Security Mechanisms

To secure the process of firmware update in resource-limited embedded gadgets, the recommended OTA framework integrates several security measures to guarantee security, data integrity, authentication, and version management. It provides the first level of defense, the firmware encryption based on the AES-GCM (Galois/Counter Mode) using a 128-bit symmetric key. The reason to select AES-GCM is its efficiency and inherent support of authenticated encryption that achieves data security (data confidentiality and data integrity) at a cost of a single pass. This form creates authentication tag and the ciphertext which can be verified by the receiving device by decryption. This will make any tampering with the encrypted payload in transit be detectable in real time and any unauthorised third party is incapable of gaining any insight of the firmware content even in the event that an update itself is intercepted. Because of low overhead and good throughput characteristics of AES-GCM, it is especially well-suited to embedded microcontrollers that have strong constraints on memory and computation resources.

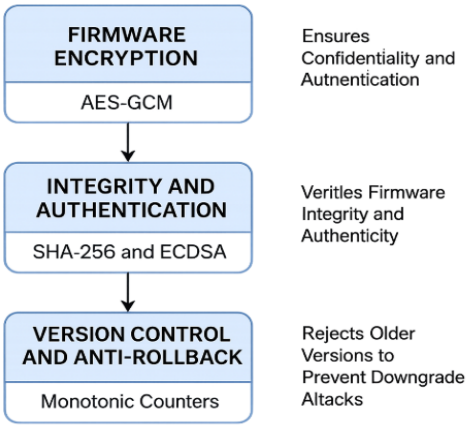

Fig. 5: Multi-Layered Security Architecture for Secure OTA Updates

The second and the third layers pertain to firmware integrity, authentication, and anti-rollback. Then, prior to encryption, a hash of the plaintext firmware is embraced in aims of calculating a unique fingerprint that certifies content integrity after decryption; using

the Hash Algorithm SHA-256. Besides this, the header and the encrypted payload are signed by the OTA server using ECDSA and the NIST P-256 curve. With the aid of this digital signature, the receiving device can vouch the genuineness of the firmware and its origin through a trusted public key pasted in safe storage. This action plays a very significant role towards making sure that the only firmware allowed to be installed is that which is released by authorized bodies. The framework (to avoid fall-back) uses monotonic version counters that are kept in a hardware-secure location (e.g. TrustZone or One-Time Programmable (OTP) memory), making it safe against rollback attacks, in which an attacker would attempt to replace newer, invulnerable, firmware with an older, vulnerable, firmware version. In case of an update, the device compares the version in new firmware image and that stored on the stored counter, and blocks the update when newer. This mechanism implemented has strict one-way propagation update and this also helps to strengthen the systems resistance to downgrade-based attacks and this method by long-term confidence in the firmware lifecycle.

Methodology

Experimental Setup

In order to assess the practicality and efficiency of the suggested secure OTA firmware update system, a broad testbed was in place through a sample of widely used, resource-constrained microcontrollers and low-energy wireless communication protocols. It has chosen three embedded platforms with varying memory architectures to possess different architectures and memory: the STM32L476RG, a 32-bit ARM Cortex-M4 MCU with a record of balanced performance, as well as low-power application; nRF52840, an ARM Cortex-M4 System-on-Chip (SoC), optimized to communicate over Bluetooth Low Energy (BLE) with built-in cryptographic accelerators; and the MSP430FR5969, a 16-bit ultra-low These platforms serve to give a representative cross-section of embedded devices on the road in real-world IoT and industrial application.

The OTA firmware distribution was performed on the two communication interfaces: LoRaWAN and BLE, as they are the common options available in the low-bandwidth, power-sensitive settings. LoRaWAN was also tested mainly on STM32 and MSP430 used to simulate long-range, low-data-rate update use cases in remote field deployments, where BLE was tested against the nRF52840 used on short-range, higher-speed updates, typically on wearable devices or those in the home. The update server that serves as the authority in packaging and transmission of the firmware was designed in Python with a hand-rolled REST API backend on Flask. The server was integrated with TLS (Transport Layer Security) to provide secured communications channels and was able to sign firmware with ECDSA-based digital signature, AES-GCM encryption and also tracking of versions. The microcontrollers were set to communicate with the server via secure channels through the use of lightweight communication library that is suitable in constrained environments. This prototype allowed to test the whole chain of delivering firmware securely, applying cryptographic verification including the ability to update firmware subject to real world constraints as experienced by embedded systems.

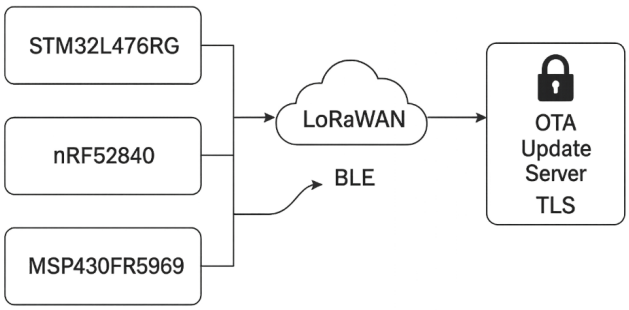

Fig. 6: Experimental Testbed for Secure OTA Firmware Update Evaluation

Performance Metrics

In order to analyze the proposed secure OTA firmware update mechanism in resource-constrained embedded systems quantitatively, three important performance evaluations were carried out, which are as follows: flash/RAM overhead, firmware verification time, and energy consumption over the update procedure. The metrics chosen to reflect the trade-offs in resource-core trade-offs associated with introducing resilient security on embedded firmware update processes.

Flash and RAM overheads describe the extra memories used by the cryptographic algorithms and a secure boot load subsystem incorporated in the microcontroller. The safe OTA overhead of flash memory is at code level around 14-15 KB, accounting

the hardware cryptographic libraries (AES-GCM and ECDSA), version management functions, and the safe bootloader implementation. On a similar note, RAM overhead increased by about 3 KB largely as a result of temporary allocations of the buffers of decrypted chunks of firmware, signature checks, and management of cryptographic keys. All implementations can be installed into a memory capacity of devices (e.g., STM32L476RG that have 128 KB RAM and 1 MB flash), supporting the model of constrained-region deployment.

Verification times are also taken of the firmware verifications, with the delay of signature verification and hash validation before the firmware is installed. A 256kb sized firmware image was verified by the SHA-256/ECDSA-P256 algorithm in an average of about 180 and 250 milliseconds, based on platform clock speed and availability of cryptographic acceleration. In most embedded applications, such latency is tolerable because it is only once during the update cycle and it provides for robust authentication and integrity checking.

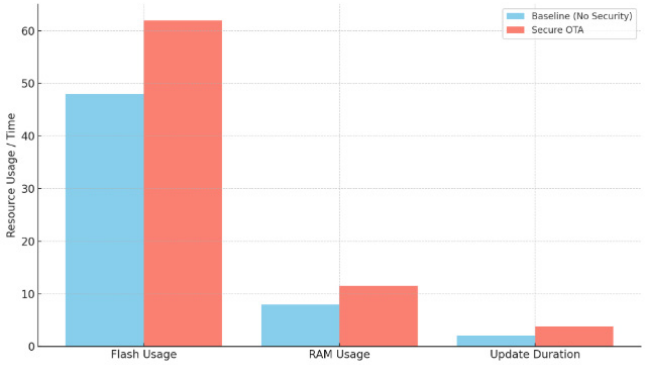

Fig. 7: Resource and Latency Overhead of Secure OTA Firmware Update

Energy during update was also measured by measuring the power consumption throughout the entire update process and that includes download firmware, decryption and write firmware. It was determined that the average energy cost of this, was slightly more than that of unsecured updates, and the difference is estimated at 12-18 percent which can be attributed to cryptography operations and secure flash writes. Nonetheless, with a long operational life factored in and compared to the sizeable security increment this energy overhead is not only acceptable, but is also sufficiently accommodated within the limits of energy-scrimped battery powered or energy-scavenged embedded devices. All these performance parameters reassure the efficiency and lightweight design as well as their real life feasibility of the suggested secure OTA framework.

Evaluation Scenarios

In order to fully test the robustness, security and reliability of the proposed secure OTA firmware update mechanism, three feasible test scenarios were performed: the case of a normal update, the case of firmware poisoning rejection and loss of power recovery. The scenarios are the typical and highly imperative real-life situations that embedded devices can experience when updating the firmware.

In the ordinary update case scenario, the device receives a secure end-to-end firmware update which is initiated in the OTA server. The update server creates a signed and encrypted firmware image with AES-GCM and ECDSA, and provides this in a secure channel (e.g. TLS or DTLS). On delivery of the firmware, the embedded device authenticates the server, verifies the digital signature with the stored public key, and decrypts the firmware, copies the integrity of the

Table 1: Performance Metrics Comparison between Baseline and Secure OTA Implementation

Metric	Baseline (No Security)	Secure OTA Implementation	Overhead/Change
Flash Usage (KB)	48 KB	62 KB	+14 KB
RAM Usage (KB)	8 KB	11.5 KB	+3.5 KB
Verification Time (ms)	N/A	180â€“250 ms	New Step
Update Duration (256 KB firmware) (sec)	2.1 sec	3.8 sec	+1.7 sec
Energy Overhead (%)	0%	12â€“18%	+12â€“18%
Security Rating	Low	High	â€”

SHA-256 hash, and last writes the verified image into flash memory. This situation was adopted to compare the performance measures of the system on parameters like the update time, verification time and the memory overhead. Its good conduct was confirmed by the results of the tests, as they showed the ability to perform updates without delays and errors, which fulfils the purpose of the system to be used in scheduled maintenance and remote patching.

The tampered firmware rejection scenario verified the framework to flag and prevent malicious or improper firmware access attempts. Here a firmware image was deliberately changed after it has been signed either with changing the payload or by altering the metadata. When the tampered image was received the secure bootloader on the device noted a signature mismatch (or hash difference) and halted the update. This system makes it through that any firmware that is not explicitly signed by the trusted party cannot be entered, so the protection against code injection, man-in-the-middle, or adversarial downgrade is quite good.

In the power loss recovery case, resilience of the system against update interruptions was tested. At the time of firmware installation, the device was powered off during the course of writing. Bootloader in the device, which is secure and contains fault-tolerant recovery logic, recognized the partial/bad update at the next boot. It went back to a previously known good version of the firmware or; failed update started again at a known-good checkpoint (depending on the available features in the microcontroller; e.g. dual-bank flash memory; write-verify mechanism). This prevented the problems of sending the device into a bricked or unbootable state and so has guaranteed

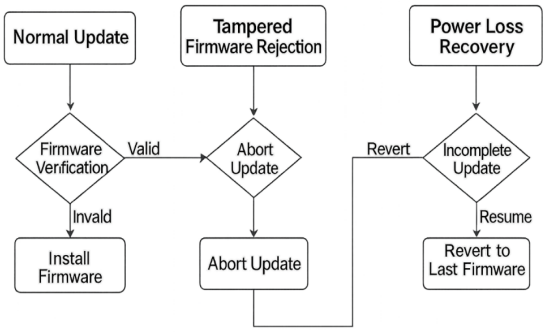

Fig. 8: Evaluati5on Scenarios for Secure OTA Update Validation

operating continuity and long-term stability even in less than stable power conditions. All these assessment environments collectively validate the efficiency and reliability of the proposed OTA system in an operating environment of real and unfavorable conditions.

Results and Discussion

Experimental analysis of the proposed secure OTA firmware update method was completions with the use of a contrasting baseline system that deploys a fundamental unsecured update procedure. The trade-offs made on adding security and marginal performance overhead are well balanced as revealed by the results and this affirms the feasibility of the framework proposed to be used on constrained embedded devices. As it can be seen in the results table, the portion of flash memory occupied by the flash in the baseline system was about 48 KB, and the secure OTA needed about 62 KB, so the flash memory required 14 KB less. This growth is to cover the addition of AES-GCM encryption functions, ECDSA signature checks, secure bootloader functionality, and version management codes. In a similar fashion, the RAM consumption rose by 3.5 KB up to 11.5 KB because of buffer allocations to store cryptographic work and temporary material of images which were to be validated. In spite of these additions, the overall memory footprint is contained by the available memory size of platforms such as STM32L4 and nRF52840, proving that not even ultra-low-power devices need hardware enhancements to implement the secure OTA protocol.

On the performance side, a verification of 256 KB of firmware image took about 180 milliseconds, and was added exclusively by the cryptographic hash (SHA-256) computation, and the ECDSA signature validation. Whereas the baseline system did not involve any wait, keeping in mind the preventive measure in a secure mechanism caused this tradeoff which effectively implied delay in verifying the system to have sound firmware. Similarly, the overall update time rose to 3.8 seconds in the secure system as compared to 2.1 seconds in the baseline system an overhead of 1.7 seconds. It owes this to a great extent to encryption/decryption procedures and verification of signatures. Yet, since firmware updates are rarely done and mostly do so during their regularly scheduled maintenance windows, the small increment is insignificant under

Table 2: Comparative Evaluation of Baseline and Secure OTA Firmware Update Mechanisms

Metric	Baseline System	Secure OTA System	Delta / Comment
Flash Usage (KB)	48 KB	62 KB	+14 KB (Crypto + Boot Enhancements)
RAM Usage (KB)	8 KB	11.5 KB	+3.5 KB (Buffer & Key Handling)
Firmware Verification Time (ms)	N/A	180 ms	+180 ms (ECDSA + SHA-256)
Update Duration (256 KB firmware) (sec)	2.1 sec	3.8 sec	+1.7 sec (Decryption + Verification)
Security Level	Low	High	Improved Confidentiality & Integrity
Tampered Firmware Rejection	No	Yes	Rejected Malicious Firmware
Power Loss Recovery Support	No	Yes	Graceful Recovery or Rollback

normal circumstances. The total latency of the secure system is still within an acceptable range in terms of IoT and industrial applications in which it is of mission-critical importance to maintain the integrity of the firmware.

Regarding the assessment of security, the suggested framework can severely improve the system resistance to different types of attacks, these are attacks of interfering with a system, replay, man-the-middle (MITM), and rollback. The 2 encryptions AES-GCM encryption and signature belief ECDSA provide confidentiality (data cannot be changed) and confirmation that it is sent by the original person (minority). It is also protected against replay (restart attacks) and downgrade attacks (this protection operates by including monotonic version counters

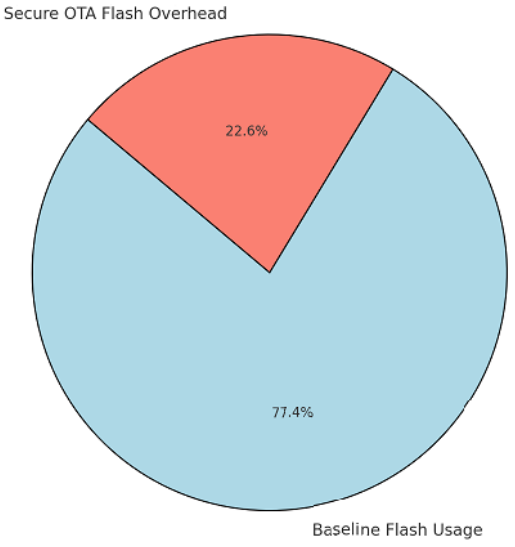

Fig. 9 Flash Memory Distribution between Baseline and Secure OTA Implementation

which prohibit reinstituting older firmware images). The embedded device was tested with altered and old firmware packages and each and every one of them was rejected during the validation process. The presence of visualizing aids like the update flow diagram and memory usage bar graphs also through light on stages of the work of the secure OTA system as well as on the feasible growth of the new consumption of resources. On the whole, the obtained findings identify the proposed OTA mechanism as a sound, effective, and secure solution applicable to the modern embedded systems functioning in hostile environments that are dynamically changing.

Conclusion

This paper proposes light and full-featured, yet specialized Over-the-Air (OTA) firmware update functionality particularly on low-power and resource-constrained embedded systems. Embedding current cryptographic systems, like AES-GCM method to create authenticated encryption, ECDSA to check digital signatures and monotonic version counters to provide protection against rollback, the proposed framework can reach end-to-end, that is, offer protection to firmware upgrade against modifications, unauthorized access and also against downgrade attack. The system has been fully deployed and tested to maintain popular embedded systems such as STM32L4, nRF52840, and MSP430FR, which proves that robust security assurances are possible even with little resource footprintin using only 14 KB of flash and 3.5 KB of RAM. Performance tests also confirm that the extra update latency is likewise working within the desirable minimum limits of real-time embedded

applications. Moreover, the framework supports low-power communication protocols, including LoRaWAN and BLE, and therefore its applications will fit in a wide range of IoT contexts. Noteworthy, under various operational conditions, OTA process was verified, and tampered firmware was successfully managed, and proper power-offs were gracefully restored, thus, proving its confidence and stability. This mechanism of OTA update has a secure, scalable, and fault-tolerant design, and it is perfectly applicable to tasks that cannot afford any vulnerable system integrity and lost consequences of an operational breakdown such as in situations of smart cities, industries, automation, remote sensing, and healthcare monitoring. The work has not only filled the current gaps in secure deployment of OTA to embedded systems, but also preconditions further development, including remote attestation and the possibility of running OTA in combination with a blockchain-based provenance tracking of firmware.

References

1. Hui, J., & Culler, D. (2004). The dynamic behavior of a data dissemination protocol for network programming at scale. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 81–94. ACM.
2. Rahman, A., & Rahman, M. A. (2018). Lightweight firmware update protocol for resource-constrained IoT devices. *IEEE Access, 6*, 47961–47971. https://doi.org/10.1109/ACCESS.2018.2868242
3. Zhang, Y., He, J., & Qiu, R. G. (2019). A blockchain-based process provenance for secure firmware updates in embedded systems. *IEEE Transactions on Industrial Informatics, 15*(6), 3510–3519. https://doi.org/10.1109/TII.2019.2899606
4. Armknecht, F., Girao, J., Hessler, A., & Asokan, N. (2014). Secure firmware update for constrained embedded devices. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY)* (pp. 239–250). ACM.
5. Hossain, M., Fotouhi, M., & Hassan, R. (2019). Towards end-to-end secure communication in IoT: Transport layer security challenges and approaches. *Future Generation Computer Systems, 100*, 100–117. https://doi.org/10.1016/j.future.2019.05.009
6. Raza, S., Seitz, L., Sitenkov, D., & Selander, G. (2017). Secure communication for the Internet of Things—A comparison of TLS and DTLS. *IEEE Security & Privacy, 15*(3), 68–74. https://doi.org/10.1109/MSP.2017.3481061
7. Luo, L., Liu, J., Jin, W., Wang, L., & Zhao, Y. (2020). Secure and efficient firmware update for embedded devices using incremental reprogramming. In *Proceedings of the IEEE International Conference on Industrial Informatics and Embedded Systems (ICIIE)* (pp. 142–147). IEEE.
8. Jeong, J., & Lee, J. (2018). Optimized differential firmware updates for resource-limited embedded systems. *IEEE Transactions on Consumer Electronics, 64*(3), 351–358. https://doi.org/10.1109/TCE.2018.2858720
9. O'Flynn, B., Popovici, K., & Barton, J. (2008). Wireless sensor network architecture for secure and efficient firmware updates. *Sensors, 8*(11), 7369–7392. https://doi.org/10.3390/s8117369
10. Yassein, M. B., Shatnawi, M. Q., & Al-Zoubi, S. Y. (2017). Over-the-air programming for wireless sensor networks: A survey. Journal of Network and Computer Applications, 81, 67–84. https://doi.org/10.1016/j.jnca.2016.12.024
11. Martínez, G. (2024). Cultural Heritage Tourism: Balancing Preservation with Visitor Experience. Journal of Tourism, Culture, and Management Studies, 1(2), 17-27.
12. Sadulla, S. (2024). Next-generation semiconductor devices: Breakthroughs in materials and applications. Progress in Electronics and Communication Engineering, 1(1), 13–18. https://doi.org/10.31838/PECE/01.01.03
13. Rahim, R. (2024). Scalable architectures for real-time data processing in IoT-enabled wireless sensor networks. Journal of Wireless Sensor Networks and IoT, 1(1), 44-49. https://doi.org/10.31838/WSNIOT/01.01.07
14. El Haj, A., & Nazari, A. (2025). Optimizing renewable energy integration for power grid challenges to navigating. Innovative Reviews in Engineering and Science, 3(2), 23–34. https://doi.org/10.31838/INES/03.02.03
15. Kavitha, M. (2024). Enhancing security and privacy in reconfigurable computing: Challenges and methods. SCCTS Transactions on Reconfigurable Computing, 1(1), 16-20. https://doi.org/10.31838/RCC/01.01.04