Sivabalaji Sivasankarapandian, International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 15, Issue 5, May 2025, pp 154-157

RESEARCH ARTICLE OPEN ACCESS

Remote Firmware Attestation and Firmware Upgrade

Sivabalaji Sivasankarapandian*
Email: ssivabalaji04@gmail.com

ABSTRACT
This paper outlines the necessity of trusted computing and focuses on secure firmware updates over the air. It discusses the secure firmware upgrade process using ECDSA signature validation.
Keywords – RoT, TCG, TPM, TCM, ECDSA, HASH, MAC, SHA-3, SHA-2, Keccak

Date of Submission: 10-05-2025 Date of acceptance: 21-05-2025

I. INTRODUCTION

To ensure that the target embedded device operates only with authorized firmware and configuration data, we must implement a method to verify both authenticity and integrity. This involves ensuring that the data is trusted and remains unaltered. Utilizing a cryptographic digital signature—similar to placing a seal or a handwritten signature at the end of a letter—provides this assurance of integrity. As IoT devices become more prevalent in our lives, the ongoing attempts by malicious actors to gain control of these devices also increase, making the adoption of embedded system security essential for device protection. Consider the threat posed when a hacker tries to modify the firmware or operational configuration data of an IoT device. While the authenticity and integrity of the firmware and data are generally protected during the manufacturing process, once deployed in the field, these devices can be vulnerable to hacker access and may require periodic updates to firmware or configuration data. Such access or updates create opportunities for intruders to alter the device's behavior or, worse, gain complete control over it, leading to potentially disastrous consequences. One example of this type of attack is malware injection, which involves inserting malicious code into the firmware update source. If an attacker manages to install unauthorized firmware, it can result in several dangerous outcomes:

1. Data Exposure: For instance, in the medical sector, malware injection could lead devices—such as portable health monitors—to unintentionally transmit private medical information. In a broader context, malicious firmware could make encryption keys accessible to the public.

2.Malfunctioning Devices: A notable incident is the Oldsmar water treatment facility attack, in which a hacker accessed the software system and increased the sodium hydroxide content from 100 ppm to 11,100 ppm, potentially causing severe harm to individuals ingesting contaminated water.

3. Unpredictable Behavior: This can manifest as erratic device operations that could pose serious risks to human safety. By recognizing these threats, we can better understand the importance of securing embedded systems in IoT devices.

II. TRUSTED COMPUTING

Trusted computing aims to enhance security in commodity computing systems. The Trusted Computing Group (TCG) creates standards and specifications for this purpose. Key features of trusted computing include:

- Secure boot ensures the system starts in a trusted configuration.
- Curtained memory provides strong isolation, preventing unauthorized access by other processes.
- Storage Island allows software to securely encrypt secrets.
- Secure I/O protects against key-stroke loggers and screen scrapers.

- Integrity measurement computes hashes of executable code and system state data.

- Remote attestation enables a device to present reliable evidence about its firmware and software to remote parties.

III. ATTESTATIONS

Software-Based Attestation.

Pioneer is an early software-based attestation method that does not require specialized hardware. It calculates a checksum of device memory using a function with side effects, ensuring any emulation results in detectable timing overheads. This method has been adapted for embedded devices but faces challenges and potential attacks, including TOC/TOU vulnerabilities. Software-only solutions depend on strong assumptions about adversarial capabilities and direct communication between verifier and prover, limiting their effectiveness for network-based attestation.

Static Root of Trust.

Secure Boot is a hardware mechanism that ensures system integrity at boot time. It uses an immutable bootloader in ROM with a public key to verify code signatures, executing only valid ones. Trusted Platform Modules (TPMs) are widely used in commercial systems for security. TPMs rely on two features: Platform Configuration Registers (PCRs) accessed via a fixed API and PCRs reset only on boot, extended using cryptographic hashes. TPMs can sign PCRs representing the software state at load time. The BIOS serves as the root of trust by performing the initial extension during boot.

Dynamic Root of Trust.

A dynamic root of trust enables attestation based on the current software/Firmware state, requiring features in extended trusted computing specs and support from CPUs and chipsets (e.g., Intel TXT and AMD SVM).

REMOTE ATTESTATION

Remote attestation is essential for trusted computing. Using TPM or TCM security chips, users can verify the identity and integrity of a computing platform's configuration. These chips check hardware, firmware, software, and virtual machine states during communication with a remote verifier, enhancing network security. Remote attestation extends trust from the terminal to the network and meets various security needs: (1) It determines if the platform supports trusted computing capabilities. (2) It detects the integrity of software input/output and platform configuration. (3) It verifies if the current configuration satisfies security policies. It's widely used in PCs, servers, embedded devices, mobile Internet, and cloud computing. It's also crucial for attesting platform identity based on TPM/TCM.

This section will describe RA and how it can be achieved in various settings. It will cover singular attestation, which is attestation of a single Verifier and a single Prover, and swarm attestation. Furthermore, extensions to RA will be described, i.e. code updates, erasure, and reset, as well as shu ed measurements and control ow attestation. A common way to implement RA is a challenge-response protocol. Such a protocol can be realised in four steps as can be seen in Figure 1. 1. Initially the Verifer generates a challenge e.g. a random bitstring. 2. The Verifer sends the challenge to the Prover. 3. The Prover calculates a proof of its local state and forwards it to the Verifer. Lastly the verifier validates the response to the challenge.

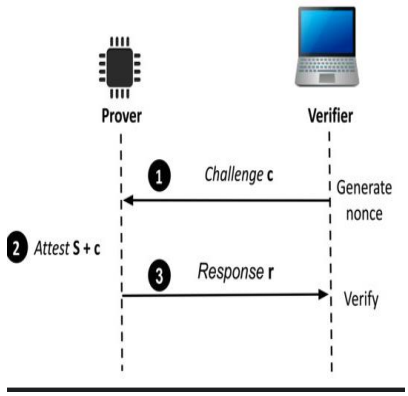

Figure 1 : Remote Attestation

Swarm attestation

The goal of swarm attestation, also known as collective attestation, is to attest a network of devices in a way that is faster than attesting each device individually. This property will be referred to as efficiency. This section will introduce the concept of scalable RA protocols, that work for a swarm of

devices, and describe how swarm attestation can be accomplished. Swarm attestation is a type of RA that, instead of describing how attestation of a single device is conducted, describes how attestation of many devices is orchestrated. These protocols are usually not dependent on the underlying RA paradigm i.e. software, hardware, hybrid.

IV. SECURE BOOT AND SECURE FIRMWARE UPDATE

An asymmetric (public-key) cryptography, mathematically related key pairs—consisting of a public key and a private key—are used for algorithm computations. As the term implies, the public key can be shared with anyone without compromising security. In contrast, the private key is critically confidential and must never be disclosed. The core principle of secure downloads based on asymmetric cryptography is that the firmware developer uses the private key to sign the firmware, while the embedded device stores and utilizes the public key for verification. Unlike symmetric-key cryptography, a key advantage of asymmetric cryptography is that the confidential element (i.e., the private key used for signing) is never stored on the embedded device. Therefore, when using Elliptic Curve Digital Signature Algorithm (ECDSA), attackers cannot retrieve the private key used for signing firmware or data, even if they employ sophisticated invasive techniques. All they can access from the device is the public key, and it is mathematically infeasible to derive the private key from the public key with ECDSA. This feature is a fundamental benefit of asymmetric cryptography.

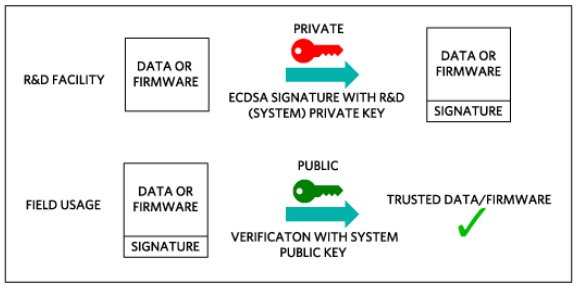

Figure 2 : Firmware Sign & Signature Verification

Figure 2 illustrates the process of implementing secure boot and secure download methods utilizing asymmetric ECDSA (Elliptic Curve Digital Signature Algorithm). This approach is designed to ensure a high level of trust in the integrity and authenticity of the firmware and configuration data, provided that the key length is sufficiently robust, typically set at a minimum of 256 bits for optimal security. The solution comprises two critical components. First, within a research and development (R&D) facility, where firmware or configuration data is meticulously crafted and produced, an ECDSA key pair is generated. This key pair consists of a private key, which must be kept confidential, and a corresponding public key, which can be distributed freely. Second, any firmware or configuration data that requires protection is digitally signed within this secure development environment using the system's private key. This signing process ensures that the data remains unaltered during transmission and verifies the identity of the signer, thereby safeguarding the system against unauthorized modifications and enhancing overall system security.

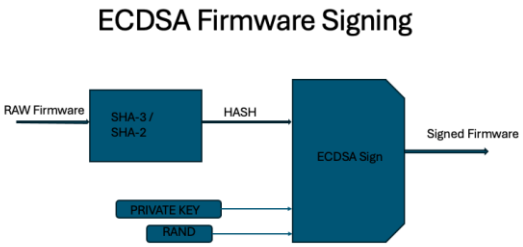

Figure 3: Firmware Sign using ECDSA

As illustrated in Figure 3, the FIPS 180 SHA-256 algorithm is integrated into the crypto-data path, which allows the ECDSA signature to be computed based on the SHA-256 hashed value of the firmware image or data file. In practice, this signature is generated and appended to the firmware or data file at the R&D facility, as shown in Figure 1. It is the signature of the SHA-256 hash that enables resources within the end application to verify both the authenticity and integrity of the firmware or data file.

For field usage, the end-application processor will utilize either internal or external resources to first perform a SHA-256 hash of the firmware or data file. Then, using this computed value along with the accessible system public key, it can verify the validity of the appended ECDSA

signature, as shown in Figure 4. If this verification check is successful, it ensures that the firmware or data file is both authentic and unmodified.

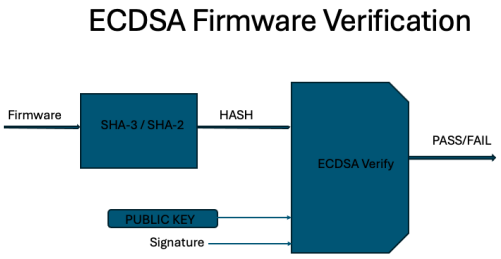

Figure 4: Firmware Signature Verification using ECDSA

Several secure boot options for firmware exist, including hash algorithms like SHA3, SHA2, and Keccak, as well as MAC and signature algorithms such as ECDSA and RSA. While RSA is typically not used for IoT devices because of its higher memory and power requirements, various algorithms are available for Data Center Servers.

V. CONCLUSION

This paper discussed remote firmware attestation and illustrated remote firmware updates using ECDSA signature validation. It also covered Root of Trust and its variations during secure boot.

REFERENCES

[1] The Elliptic Curve Digital Signature Algorithm (ECDSA) Don Johnson and Alfred Menezes and Scott Vanstone Certicom Research, Canada Dept. of Combinatorics & Optimization, University of Waterloo, Canada

[2] S. Lamba and M. Sharma, "An Efficient Elliptic Curve Digital Signature Algorithm (ECDSA)," 2013 International Conference on Machine Intelligence and Research Advancement, Katra, India, 2013, pp. 179-183

[3] P.O. L. Keleman, D. Matić, M. Popović and I. Kaštelan, "Secure firmware update in embedded systems," 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin), Berlin, Germany, 2019, pp. 16-19

[4] A. Kolehmainen, "Secure Firmware Updates for IoT: A Survey," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018, pp. 112-117

[5] Huang, J., Zhang, HJ., He, S. *et al.* A remote attestation mechanism using group signature for the perception layer in centralized networking. *J Wireless Com Network* **2022**, 11 (2022). https://doi.org/10.1186/s13638-022-02092-9