

Securing Embedded Firmware Update over the Air with Distributed Encryption

D. Vizár, M. I. Ben Salah, Q. Liang

Secure firmware update over the air (FUOTA) and secure boot are fundamental for every Internet of Things (IoT) device fleet, for ensuring authenticity, integrity and secure remote deployment of firmware used on an embedded device. Also, to protect the IP within, the update packages are routinely encrypted. Using threshold encryption mitigates the risk of an attacker getting a perpetual access to firmware updates after a successful attack.

For applications with many connected embedded devices (a.k.a. IoT applications), the features of secure boot and secure FUOTA are essential. The former ensures that only unmodified firmware intended by the application owner can boot on the devices, which may be subject to remote attacks exploiting vulnerabilities in the communication stack, for example. The latter then enables the firmware in the device to be legitimately and securely changed remotely by the application owner only, e.g., to patch security vulnerabilities, fix bugs or to enable new features. To ensure the integrity and authenticity of the firmware, digital signatures are typically used.

In addition, firmware must routinely remain confidential, to protect sensitive IP contained within, or to mitigate an effective identification of software vulnerabilities by attackers. The typical solution is to encrypt the firmware package in transport, either using the same decryption key for all the devices, or by using a personalized decryption key for every device. The former approach, while simpler to deploy, suffers from a weakness, whereby it suffices for an attacker to successfully extract the decryption key from a single device to ensure a perpetual access to all subsequent firmware updates. Indeed, possessing the universal decryption key makes the attacker equivalent with a genuine device. The approach with personalized keys has the advantage of being able to revoke a compromised device, such that no more updates are encrypted with its keys but at the cost of increased overall complexity, necessitating a secure creation and distribution of personalized update packages. Upon a closer inspection, the security benefits are not always certain, as an intelligent attacker may faithfully simulate the behavior of the attacked device, making it difficult to detect that a device needs to be revoked in the first place.

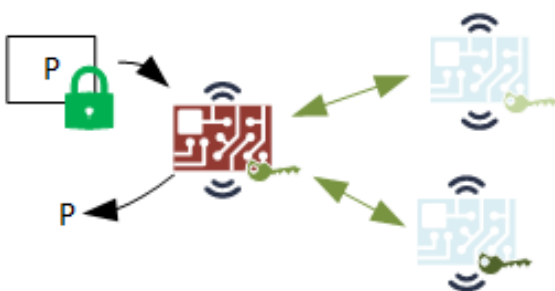


Figure 1: A device (red) performing distributed decryption with help of two other devices, recovering the payload P . Each has a key share, a threshold t (here $t=3$) of key shares is needed for each decryption.

This problem may be alleviated with threshold encryption, which allows a secret key to be split into n shares (n is a parameter), such that each share is given to one device and for every decryption, t devices must interact as depicted in Figure 1. The

threshold t allows for a trade-off between security and efficiency; the higher t , the more shares must an attacker acquire to be able to decrypt but also the more devices are involved in a decryption. CSEM developed an embedded implementation of the threshold encryption scheme DiSE [1] for the Nordic nRF52840 SoC [2,3].

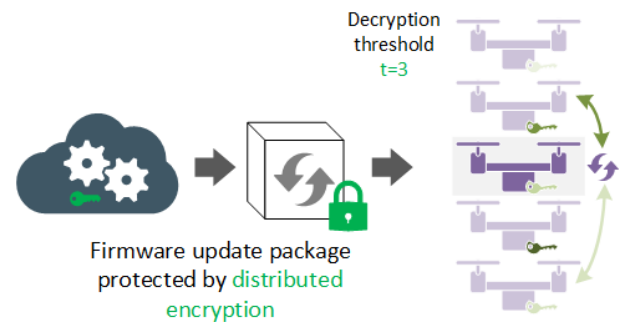


Figure 2: A firmware package secured by threshold encryption is sent from cloud application to an end-device (e.g., a drone), which decrypts the package with help of other nearby devices and updates.

CSEM designed an architecture using the above scheme to mitigate the issues created by leaked device keys. Figure 2 illustrates the process of firmware update, in which the payload of a firmware package is encrypted using threshold encryption prior to distribution, then decrypted by the target device using distributed decryption. In this architecture, a single, well-secured encrypting node may use a master key for efficiency, or a distributed architecture with multiple encryption nodes is possible for extra robustness to attacks on the cloud. It has been implemented by integrating DiSE with the MCUboot secure boot framework and the CSEM real-time operating system $\mu 111$ and demonstrated using AWS cloud as the firmware dispatch backend, sending the update through an AWS-native ESP32 running FreeRTOS as a gateway to an nRF52840 devkit, which successfully updated with help of another nRF52840, performing the distributed decryption over BLE. In addition, the challenge of providing the device bootloader with auxiliary decryption information from other devices without integrating the communication stack into it has been resolved with a dedicated authenticated protocol between the peer devices, where the firmware requests from the peers encrypted auxiliary information on behalf of the bootloader and places the obtained data into a dedicated region of internal flash, such that only the bootloader can decrypt these.

The FUOTA with threshold encryption developed at CSEM can help manage and mitigate risks in many risk-sensitive verticals, such as drone fleets, Industry 4.

[1] S. Agrawal, P. Mohassel, P. Mukherjee, P. Rindal, "DiSE: distributed symmetric-key encryption", ACM CSS (2018) 1993.

[2] D. Vizár "Distributed encryption for robust data confidentiality and integrity for IoT", CSEM Scientific and Technical Report (2021) 48.

[3] R. Müller, "Robust IoT security with threshold cryptography", bachelor's thesis, HEIG VD (2021).