Secure Firmware Update Over the Air in the Internet of Things Focusing on Flexibility and Feasibility
Proposal for a Design

Silvie Schmidt[1], Mathias Tausig[1], Matthias Hudler[1], and Georg Simhandl[2]
[1]*Competence Centre for IT-Security, FH Campus Wien, University of Applied Sciences, Vienna, Austria;* [2]*Adaptivia GmbH, Vienna, Austria*
{silvia.schmidt, mathias.tausig, matthias.hudler}@fh-campuswien.ac.at; georg.simhandl@adaptivia.com

Keywords: Firmware Update, Secure Firmware Update Over the Air, FOTA, Internet of Things, IoT, Embedded Systems

Abstract: The rapid development of the IoT challenges various fields of security. One of the most delicate issues in this area is the process of updating the firmware of a device connected to the IoT. This article describes the development of a design for such an update process focusing on flexibility and feasibility - without neglecting security. This is achieved by our proposal due to an enhanced bootloader which “decides” the kind of update process used based on the user’s settings. Our proposal consists of four update-process-packages, each offering different feature sets of security.

1 INTRODUCTION

The Internet of Things (IoT) is becoming more and more a part of our everyday life. Due to the rapid development of the IoT and the demand for more features by users, the process of updating the firmware of the “Things in the Internet” gained in importance regarding its security.

In our setting the firmware is the RIOT-OS[1] application running on our board. Updating(Zimmer et al., 2015) this firmware means correcting bugs, adding new features, patching security, etc. If we think about security, we mainly think about protection against unauthorized access to the device and against threats from malware, and the Internet (Kleidermacher and Kleidermacher, 2012). Subsequently, we defined various **preconditions** for our test setting: there is no physical access to the devices and as an evaluation board we chose Atmel SAMR21-Xplained Pro Evaluation Board (Cortex-M0+, 256kB Flash,32kB RAM).

Firmware updates over the air (FOTA) arise various security issues, and this does not automatically target malicious attackers, i.e. a wrong firmware might be uploaded, the transmission of the new image failed, or the new firmware simply does not work as intended. Additionally, without any security checks malware can be uploaded instead of the firmware update. Often, a failed update process causes the device to become unusable at all. Regarding embedded devices used nowadays, this lack of security might have severe consequences since cars or even medical devices are part of the IoT.

[1]https://www.riot-os.org/

2 CONSIDERATIONS FOR A SECURE REMOTE FIRMWARE UPDATE PROCESS

The most sensitive issue concerning embedded systems security is the process of updating the firmware.

The firmware’s integrity has to be validated and may be decrypted. Without any security mechanism, the update would be written over the old one without any revision. There are several strategies to update firmware concerning security, system stability, and transmission reliability.

Memory Partitioning (Atmel, 2013) is a technique to ensure the continuous availability of a working firmware. This strategy requires double memory size which is a drawback for devices with small memory size or applications with big code size.

Figure 2 shows our concept for an optimal Memory Partitioning containing explicit areas for the update and the backup. All tests highlighted in figure 1 are performed in the update area - except the operability, which is executed at the firmware’s area. Consequently, the update package is copied into the backup area if all checks were successful. If the update process fails, the backup is installed.

2.1 Requirements of a Secure Remote Firmware Update

The firmware integrity has become the most important security issue for embedded devices because its protection solves various security issues regarding the update process. Therefore, signing the firmware update is easy to implement and subsequently, often the only security mechanism used. This is also caused by concerns about decreasing performance due to enhanced security. Nevertheless, signing the firmware update should never be the only security feature provided (Cui et al., 2013). Firstly, we identified several requirements regarding security issues derived from the process of updating firmware. Nevertheless, we are aware that the requirements may differ concerning the device and/or its application.

The major requirements are:

- *Authentication:* The device may only accept software from a specific source.
- *Version Control:* Only the version intended for the device shall be accepted; this also prevents the installation of outdated software.
- *Package Integrity and Complete & Error-Free Transmission:* Tampered or incomplete firmware may not be accepted by the device. After its transmission the update package has to be checked for errors and it has to be transfered completely.
- *Operability Check:* The new firmware has to be checked if it is working as intended.
- *Reduced User Interaction:* The user should not have to interact extensively since this is a "good" source of errors.

Our assumption is, that if stronger security is required efficiency decreases. Therefore, a balance has to be found. What is crucial for my device/application? Moreover, one has to decide if system reliability is worth the cost of a larger memory. Summing up, a flexible solution has to be found for a secure firmware update.

3 SECFOTA - PROPOSAL FOR A SECURE FOTA

The **delivery status** of the embedded device (client) contains the bootloader + public keys + secret keys (including fall-back keys) and the initial application/firmware.

3.1 Update Process

After evaluating the issues discussed in the previous sections, we designed a secure remote firmware update process. Due to our assumption regarding the impact of enhanced security on performance and memory costs, we provide four packages, which we call SecFOTA-packages. The encryption and the signing of the new firmware is executed by the server.

The bootloader calculates which SecFOTA-package to use, based on the size of the firmware and the memory's size - and of course by the requirements of the user and application. The package with the best possible options is prioritized. The features provided by the SecFOTA-packages are:

- **Integrity and Authentication** through digital signature

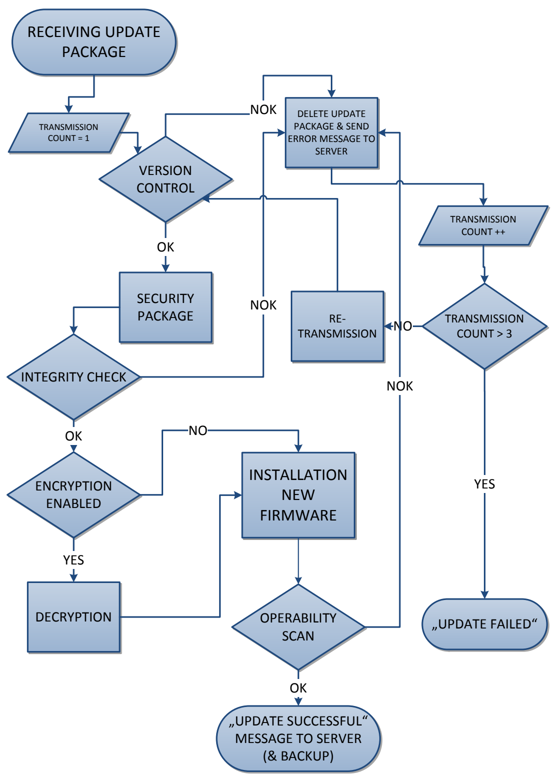

Figure 1: Concept for SecFOTA - Client-Side

- **Fault Tolerance** via backup
- **Confidentiality** by the means of encrypting the transmitted update of the firmware

Table 1 shows the features provided by each package. A memory table containing all necessary addresses is managed by the bootloader in any case. The

Red Pkg	SIG, EE, B
Blue Pkg	SIG, EE, NoB
Yellow Pkg	SIG, ED, B
White Pkg	SIG, ED, NoB

Table 1: SecFOTA-packages providing various features: encryption enabled (EE), encryption disabled (ED), backup (B), no backup (NoB), signature (SIG).

update process on the client's side is illustrated in figure 1.

3.2 SecFOTA-Packages in Detail

The level of security - by the means of integrity, authentication, fault tolerance, and confidentiality - is represented by various SecFOTA-packages; it is a corporately decision by the user and the bootloader, which package to apply.

In our concept the **update area** is a pre-defined address area in the flash memory, where the verification of the update package's signature is executed. All solutions contain a **memory table**, which is managed by the bootloader. It holds the addresses for the

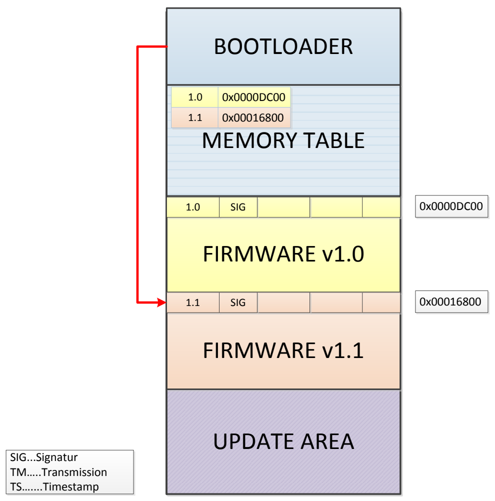

Figure 2: Red Package Sketch

various areas, the version number, and all important informations regarding the applicable security package.

- **White Package:** representing low security, but good performance regarding runtime.
- **Yellow Package:** is the same as the White Package, but it keeps a backup in the Flash memory. Consequently, this solution saves memory in the size of the firmware.
- **Blue Package:** offers memory-saving strong security.
- **Red Package:** stands for strong security, system stability, and transmission reliability. The draft for the main aspects of this solution is illustrated in figure 2.

3.3 Techniques

The following techniques are considered to implement the ideas discussed in 3.1 and 3.2:

- **Signature:** ECDSA is our first choice; due to its performance advantages Elliptic Curve Cryptography (ECC) is predestined for constrained devices.
- **Encryption:** Targeting a generic proposal we have to consider a bare-metal C-implementation of AES.

4 CONCLUSION AND FUTURE WORK

A balance between efficiency in time and memory consumption on one side, and the level of security on the other has to be determined. Finding a preferable generic solution is the goal of this work; at least, an evaluation if such a solution is feasible. Therefore, our future work will contain mainly performance tests of cryptographic algorithms on the board.

4.1 Open Questions

- Is the bootloader interchangeable? Which security issues arise?
- What tests can be performed to check the operability/functionality of the firmware?
- Are timestamps useful for re-registration-check of the client? Is it necessary in the IoT?
- If there is no backup kept on the device, is there a possibility to solve this problem from the server-side? Is the device unusable forever? Is it reasonable to keep a small application with the network stack on the device?
- Is Secure Boot an option as a root of trust? Is it decisive for SecFOTA?

ACKNOWLEDGEMENTS

Zoran Stojkovic, z.stojkovic@embed-it.com, Embed-IT GmbH.

vienna business agency

A service offered by the City of Vienna

Supported by Vienna Business Agency.

REFERENCES

Atmel (2013). Atmel: At02333 - application note on safe and secure bootloader implementation for sam3/4.

Cui, A., Costello, M., and Stolfo, S. J. (2013). When firmware modifications attack: A case study of embedded exploitation. In *NDSS*. The Internet Society.

Kleidermacher, D. and Kleidermacher, M. (2012). *Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development*. Elsevier.

Zimmer, V., Sun, J., Jones, M., and Reinauer, S. (2015). *Embedded Firmware Solutions: Development Best Practices for the Internet of Things*. Apress, Berkely, CA, USA, 1st edition.