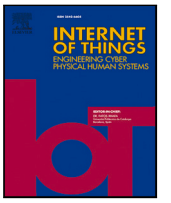
Internet of Things 18 (2022) 100508

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Review article

Secure firmware Over-The-Air updates for IoT: Survey, challenges, and discussions

Saad El Jaouhari [a,*], Eric Bouvet [b]

[a] ISEP Ecole d'ingenieurs du numerique, LISITE, 28 Rue Notre Dame des Champs, 75006 Paris, France
[b] Orange Labs, 4 Rue du Clos Courtel, 35510 Cesson Sévigné, France

ARTICLE INFO

Keywords:
IoT devices
Firmware
Updates
Over-The-Air
Security
Privacy
State of the art
Survey
Challenges
Threats
Trust chain

ABSTRACT

The Internet of Things (IoT) market has shown strong growth in recent years, where many manufacturers of IoT devices and IoT-related service providers are competing. Time to market has become essential to be competitive. The faster a competitor develops and integrates his product, the more likely he is to dominate the market. This competition could lead to critical security issues due to the lack of testing or the short development time. Moreover, lots of IoT devices present some vulnerabilities that can be exploited by attackers. They are also constantly subject to Zero-days, which require quick intervention to maintain the security of the environments in which they are deployed in. For these purposes, the quick update of the firmware image of these IoT devices is an effective way to counter most of these attacks. This document starts by defining the firmware update mechanisms for IoT, and in particular the ones done Over-The-Air. Then presents a state-of-the-art of the currently proposed solutions, with the particularity of surveying from the literature, the standardization bodies and from some well-known industrial solutions. It also proposes a new classification of the different types of System on Chip (SoC) present in the marketed IoT devices together with an analysis of the different challenges and threats related to the OTA update. The objective is to open up the horizon for future research directions.

1. Introduction

IoT devices (i.e. smart things, connected objects, smart objects, etc.) are rapidly expanding and invading our daily lives whether at home, on the street, or at work. These digital equipment are being developed by many manufacturers that often prefer performance and profitability over robustness and safety. The software ecosystem associated with these connected objects is as varied as there are microcontroller (MCU) manufacturers for low-capacitance objects or embedded systems or System on Chip (SoC) for more complex and energy-intensive objects. These digital objects are ubiquitous, and they are increasingly integrating a communication channel in order to communicate with the outside world. Consequently, new threats are now possible either on the functioning of the objects themselves or via attacks targeting the data they carry in transit. The widespread distribution of these IoTs also favors massively distributed attacks toward Internet infrastructures (e.g., Botnet DDoS). In order to be present in the network, these IoT devices use an association procedure that allows them to access network gateways (access point gateway). More or less secure, this association allows the IoT device to communicate with other objects and equipment connected to the local network, or even more if there is a link to other networks (e.g. Internet or mesh network). The level of trust that can be associated with a network of connected IoT devices will depend on the level of security implemented in each element.

* Corresponding author.
E-mail addresses: saad.el-jaouhari@isep.fr (S. El Jaouhari), eric.bouvet@orange.com (E. Bouvet).

https://doi.org/10.1016/j.iot.2022.100508
Received 16 July 2021; Received in revised form 10 February 2022; Accepted 12 February 2022
Available online 26 February 2022

Moreover, in order to have a better resilient and reliable IoT solution, it is important to be able to react quickly to the different vulnerabilities that appear in the IoT devices by patching them before infecting the rest of the network. Depending on the capabilities and software environments of these IoT devices, new technologies make it possible to more or less securely update the firmware image and the other necessary updates to correct the security flaws. In practice, there are multiple ways of updating an IoT device, since different manufacturers follow different approaches depending on their tools, infrastructures, and strategies. These ways fall in one of these two categories: (1) the wired updates (e.g., via a USB port) or (2) the wireless Over the Air (OTA) updates. Also, the update can be either full or partial (e.g., only a piece of the firmware or the OS), it may include signature and integrity verification and it may require an attestation of delivery and installation. Furthermore, the lack of a consensus reference is pushing different groups to improve this safety through requirements, recommendations, and standards such as ENISA (European Union Agency for Network and Information Security) [1], which has recently published a list of recommended security measures to increase the level of security and confidence of connected objects, and the IETF (Internet Engineering Task Force) [2], which is working on a standard for updating connected objects.

The main objective of this paper is to converge to a solution that allows the update of heterogeneous IoT devices in an efficient and secure way, while taking into account their constraints and their limited capabilities. From our point of view, building a trust chain from the manufacturer/provider to the end-devices in the local network of the IoT device (which can be managed by the Internet Service Provider (ISP) for instance) is a possible solution for a secure Firmware Over-The-Air (FoTA) update process. This establishment of the trust chain is mainly based on the evaluation of the security features and capabilities of each element in the supply chain. The problematic part is the one related to the IoT devices due to their constraints (i.e., limited interactions, sleep mode, battery-powered, etc.), their huge number, and their heterogeneity (i.e., different SoC from different manufacturers and different providers, different communication protocols, etc.). This work is an initialization step toward investigating implementing and testing this solution in future works.

On the other hand, for ISP, IoT service management is a part of the new technological innovation linked to the emerging IoT, and it starts with managing the connectivity. Once basic connectivity is offered, ISPs are well-positioned to add features in different areas. They can extend their role even more by also having a suitable device management system with adaptive security, scalable and efficient routing among other features, specially tailored for the IoT environment. Hence, toward establishing a strong IoT service management position, ISP needs to deploy additional functionalities into their platforms. For instance, more related to the topic of this paper, and related to device management, operators can provide device security and authentication, remote detection of troubleshooting problems, and FoTA updates for IoT devices. These features become more crucial if the concerned IoT devices are also part of the products offered by the ISP. Moreover, in order to implement such features, they can take advantage of the already deployed hardware (e.g., routers, gateways, etc.) in order to connect, manage and interact with different IoT devices in different environments. For instance, in the smart home environment, it can be the router, or a mobile application that will play the role of the gateway in order to manage the smart home's IoT devices.

1.1. Key contributions in a glance

In this survey, we provide a critical review of most existing papers. Then, we compare them, in Table 2, based on multiple factors, as explained in Section 5. Moreover, we would like to highlight the key differences between our survey and other existing works:

- The paper provides a literature review of the existing works from the different digital libraries, from the standardization bodies, and also from industrial solutions, which are not treated in most of the surveys.
- It proposes a new classification of the different types of IoT devices based on their hardware and the embedded security components. We argue that the current classification does not follow the current advancement in microcontrollers capabilities in IoT devices.
- It highlights the main challenges to deal with toward an objective of a generic and secure approach for FoTA updates for IoT devices.
- Finally, it provides the ISP's point of view regarding IoT service management since it is a part of their new technological innovation linked to the emerging IoT. It discusses mainly the extra responsibilities regarding the management of such IoT devices and the challenges that need to be solved to provide secure services to the users.

1.2. Structure of the paper

To sum up, his paper starts with a quick background presentation of some important notions, in Section 2, together with a new classification of IoT devices based on their hardware capabilities. Then, in Section 3, the firmware update-related threats are listed, followed by the main challenges facing the generic and secure aspects of the FoTA updates for IoT in Section 4. Next, a study of the different solutions proposed in the literature, standards, and industries is presented and compared in Section 5. The paper, then, presents the ISP point of view in Section 6, and finishes with a conclusion and future works.

2. Background

This section provides an overview of some important related topics to have a clear view of the different challenges related to the FoTA update for IoT devices.

Class	RAM	Flash	Presence of some security features ?	Examples MCU
Class 0, C0	<< 10 KB	<< 100 KB	Yes	PIC24FJXXXGB20X; SAML11; STM32WLE5J8
			No	PIC12LF1552; PIC16(L)F145X; SAMD09;
Class 1, C1	~ 10 KB	~ 100 KB	Yes	ESP32-S2; STM32L081CB
			No	PIC32MX1
Class 2, C2	~ 50 KB	~ 200 KB	Yes	nRF52810; nRF52811; STM32L082CZ
			No	ESP8266EX; ESP8285; PIC32MX7
Class 3, C3	[50 KB,128 KB]	[200 KB, 512 KB]	Yes	nRF52833; nRF52832; STM32F215RE; nRF51822; PIC32MZEF
			No	STM32F205RC; STM32F205RE
Class 4, C4	[128 KB, 512 KiB]	[512 Kib, 1 MB]	Yes	nRF9160; nRF5340; nRF52840; PIC32MZDA
			No	STM32F205RG
Class 5, C5	> 512 KB	> 1 MB	Yes	ESP32-D2WD;
			No	ESP32-U4WDH;

Fig. 1. IoT SoCs updated classification.

2.1. Constrained IoT devices

In the market, there are millions of IoT devices ranging from small motion detector sensors to huge connected devices such as autonomous cars. To have a generic FoTA approach, a classification of the different types of constrained devices needs to be defined. The RFC7228 [3] classify the different IoT devices into three main classes. However, we argue that with the current advancements of the microcontrollers manufacturing and the new microcontrollers (MCU for microcontroller unit) that have more and more computation power, the current classification needs to be updated. Moreover, the current IoT devices may even have more than one chip inside a very tiny device. Such devices, may for instance contain SoC for the software (i.e., bootloader and firmware), a microcontroller for WiFi network and microcontroller for BLE network, and so on. Furthermore, among these devices, some offer security components, while others do not, which makes it even harder to classify these devices. Thus, in this document, we propose a classification based on the most powerful chip inside the constrained device. Based on it, we can at least determine if the device can perform any cryptographic computations, thus, able to provide basic security. In this document, an extension of the RFC7228 classification is provided together with examples of some popular SoCs such as ESP32S2 [4] provided by ESPRESSIF, nRF52810 [5] provided by NORDIC Semiconductor and STM32WLE5J8 [6] provided by ST Microelectronic, as presented in Fig. 1. Moreover, a special focus was given the security aspect in order to show that nowadays even class 0 (C0) devices can have security mechanisms and cryptographic embedded hardware. The proposal is an arguable choice and will be a base for the rest of this paper.

2.2. Secure element

To achieve the full potential of the IoT, establishing a trusted environment is the key. In such an environment, each IoT device is clearly identified and authenticated to remove any malicious node/element, and where the data are transported securely via end-to-end mechanisms. Security must be designed starting from the devices up to all points in the ecosystem. For this purpose, it is recommended to deploy only IoT devices that embed a Secure Element (SE) [7]. A SE is a tamper-proof chip that can be directly embedded in the IoT device [8,9]. It ensures that the data is stored in a safe place and that information is given to only authorized applications and devices. Such element provides mainly protected storage for the device master secrets (e.g., cryptographic keys securely provisioned at manufacturing time such as the public key of the firmware provider to verify its signature), secure access to the credentials, and in the best case end-to-end security to the remote endpoint (mainly for firmware updates). However, this secure element is not implemented in most IoT devices mainly for the overhead financial cost of adding another chip, and also for the additional resources that are not available in the very constrained devices.

2.3. Root of trust

Ideally, a secure FoTA update procedure starts from a secure non-compromised IoT device deployed under certain strict conditions. Such conditions can be for instance that the IoT device is operating as expected; the firmware is up to date; and that the firmware image has not been tampered with in any way during the transmission phase. For this purpose, the Root of Trust (RoT) is based on a hardware-validated boot process to ensure that the device can only be started using code from an immutable and trusted source [10]. Hence, it guarantees the authenticity and the integrity of received supposed trusted software at boot time. Moreover, since the anchor for the boot process is embedded in the hardware (i.e., cryptographic keys), it cannot be updated or modified in any way, especially when combined with a cryptographically secured boot process. The latter closes the easily accessible gaps for hackers to exploit. The main foundations of the RoT are: (1) the multiple security measures that protect the Integrated Circuit (IC) against physical and logical attacks, (2) the highly-secure storage for user-specific credentials such as cryptographic keys and certificates, ideally using a Secure Element (SE), (3) a truly random source, that allows the unique identification of the device in the network, and eventually the mutually-authenticated, end-to-end and trusted connections with remote servers or cloud service

Table 1
Pros and Cons of the OTA modes.

Mode	Strengths	Weaknesses
Push	• Efficient for small-sized updates. • Allows automatic scheduling for pushing updates when there is a zero-day for instance or a new version. • Several IoT devices (e.g. from the same brand and version) could be updated in a controlled manner.	• IoT devices must be registered into the update server and must be uniquely identified. • For security concerns, a secure channel must be established, which is not simple for constrained IoT devices. • The scheduling of the updates is not also easy to keep the availability of all the devices and additionally real-time can be an issue for the last-scheduled devices. • The firmware server needs to securely store and remember all the IoT devices. • Spike of update transmission and load in the firmware server, which is caused by in case of a tight scheduling of FoTA update for each IoT device (or with the corresponding gateways) with the corresponding secure channels.
Pull	• A good alternative when a gateway is deployed. • Each gateway can schedule the locally managed network.	• Pull is not done in real-time, which can be an issue for urgent updates. • A significant scale of pull may apply a significant resource consumption on the firmware server side.

providers. This can be achieved for instance using the credentials stored on the SE and the unique identifiers. For IoT, SEs can either take the form of (1) fully functional SoCs, with basic computing capabilities (which is suitable for critical applications such as payments, industrial devices, smart grid control, etc.) or (2) it can be based on a Field-Programmable Gate Array (FPGA) solution, which is a general-purpose chip on which the logic can be configured after manufacture. Moreover, the FPGA solutions offer the possibility to use a low-power, low-cost, secure element on very basic IoT devices [11,12].

For the first form of SEs, ARM as an example, provides their own RoT architecture based on their TrustZone solution embedded on chips (in particular for the Cortex-M) [13]. From their point of view, the central elements that need to be protected are the keys/certificates usually provisioned during manufacturing time. Then, onion layers of security measures up to the OS and the applications are formed to secure the IoT device. The keys are mainly protected by the TrustZone CryptoCell, which is considered as the initial layer of the RoT. On this basis, the secure boot flow starts with the initial boot block (IBB) (that is immutable and is a trusted entity owned by the vendor or the manufacturer). Then, all the software images beyond the IBB are digitally signed and secured at each level/ring while empowering the least privilege principle. Typically, the baseline-boot verifies the signature of the firmware boot that in turn verifies the OS code and so on up to the application codes to establish the trust chain for the secure boot.

As for the second, Microsemi as an example, provides an FPGA-based solution called SmartFusion2 SoC FPGAs [14,15]. They argue that it minimizes power and offers small form factors without compromising on functionality and cost. It also deploy security solutions that prevents tampering, counterfeiting and installation of malicious code.

2.4. Modes of FoTA updates

FoTA updates for IoT devices are deployed mainly in two modes [16], each one has its strengths and weaknesses as it is explained in Table 1:

1. Pull mode: is practical for capable devices, such as gateways, that can manage other devices and has the support of a wide range of communication protocol stacks. Moreover, they are most likely always powered and directly connected to the Internet. It is a client-initiated mode, where the client queries the update server for new updates periodically, and when an update is available the client downloads, verifies and then installs the new firmware. In this case, the remote firmware server sends either a URI of the firmware image repository via a data structure or directly the new firmware image binary.
2. Push mode: is mainly applicable for resource-constrained IoT devices with limited protocol stack support. It can also be seen as a server-initiated mode, where the update server pushes the updates to the client when there is a new patch available. The same procedure of downloading, verifying and installing the new image is then launched.

Some implementations combine the two modes to have more flexibility for updating different types of IoT devices with different requirements and availabilities. This new mode is called “Hybrid”. Moreover, the firmware can be either fully updated or only partially updated. In the IoT environment, partial updates should be possible in particular for the IoT devices that do not have enough capabilities. It depends mainly on the frequency the IoT devices are connected for updates. This option would decrease bandwidth consumption and on-device processing time since less code is downloaded and processed, reducing the total update time and making it easier to update even the isolated rarely awake devices if properly scheduled. Currently, partial update are usually present in classes 4 and 5 as in Fig. 1.

2.5. Mobile gateway for IoT devices

In various cases, the gateway takes the form of a mobile application that manages a set of IoT devices, mostly from the same manufacturer and provider. It allows making the gateway more portable and easily accessible to the users, using a user interface. This mobile application is connected to two worlds, on the one hand to the IoT world by interacting with the IoT devices and on the other hand to the Internet to allow for instance the FoTA update.

It also, however, presents a security risk if not properly implemented. Among these risks we mention:

- *Privacy risks*: in addition to the FoTA update, the application may manage and process some IoT data, that can be sensitive data [17]. Thus, it needs to be compliant with the General Data Protection Regulation (GDPR) laws. If the data are not processed locally in the application, it needs to give guarantees that the data going through are protected, by encrypting them for instance.
- *Security related risks*: due to the complex world of mobile applications, there are more and more vulnerabilities and zero-days since it can be an easy way to gain access to the mobile of the user and hence to his private information [17]. Moreover, if the runtime environment is not properly implemented, by isolating the different applications and limiting their interactions, it becomes possible to corrupt the IoT application by other malicious applications.

In order to protect the IoT environment from the risks related to the IoT applications and to improve the overall security of the system, several countermeasures can be implemented such as:

- Focusing on the application and device security from day one, following the security and privacy by design concepts.
- Updating the applications regularly.
- Implementing device authentication to provide secure access to the application by only the authenticated users and applications.
- Integrating safety features in IoT mobile apps, to react in case of abnormality.
- Doing extensive research on threats and vulnerabilities especially the new ones.

For these purposes, it is always better to keep the whole system updated (IoT devices and also related applications) and to remain ready to face any attempt to break security.

3. FoTA update related threats

This section exposes the different types of attacks that may threaten the FoTA update procedure, based on reflections, research papers, and well-known attacks published online. These attacks are divided into three categories depending on the target (i.e. the IoT devices, the gateways and application, or the suppliers).

3.1. Target: IoT devices

Since the IoT devices are the weakest node in the update process, the attackers will mainly try to exploit their constraints in order to infiltrate the local network of the user for malicious purposes. In this section, the attacks targeting these devices are highlighted.

The "**Rollback attack**": the hacker re-sends a valid but old firmware version to the devices [2]. "**Mismatched firmware**": the attacker sends this time a valid firmware but for a different type of IoT device, which will lead to the dysfunction of the device. Thus, the devices become unavailable [18]. Another attack targets the very constrained devices that are most of the time in sleep mode, this attack can be called "**Offline update attack**". Since some IoT devices are most of the time offline, they may have missed a couple of firmware updates. In this case, the attacker sends a newer version than the one installed in the devices but still not the latest version. This newer version may still have some unpatched vulnerabilities, that can be exploited [18]. "**Repeated update requests attack**": can be seen as a kind of DoS attack where the malicious peer sends as many as possible new FoTA update requests either with valid or invalid firmware. It leads the IoT device to check the integrity and the authenticity of the firmware each time. This attack provokes the unavailability of the IoT device and extra power consumption, which is problematic for the battery-powered ones. "**Device cloning attack**": the attacker replicates some IoT devices so that only the cloned one will be updated and not the original devices. In this case, the attacker will simply remove the cloned devices from the network and will try to exploit the unpatched devices. "**Non-ephemeral keys attack**": in this case, the attacker tries to deduce the asymmetric key pair from all the exchanges that have been done between the IoT device and the other entities, in particular since the IoT device will rarely renegotiate a new pair of keys. The attacker can then forge a signature on a rogue update [18]. "**Unchanged default password**": is not a particular problem for IoT but for all the devices where the default password was never changed, and where it can be the same for all devices of the same model manufactured by the same manufacturer. Hence, an attacker who possesses a similar device model can suppose that the target has also the same password. Furthermore, a weak password can be easy to guess. In both cases, an attacker may aim to get unauthorized access to the device using the obtained password. "**Firmware reverse engineering**": is another type of process an attacker can perform, by reverse-engineering the binaries into assembly in order to analyze the functionality and to get access to secret data [2,19].

3.2. Target: Intermediaries

In this case, the attacker targets the intermediaries between the IoT device and the final firmware supplier. These intermediaries can be gateways or applications since in most cases they are the one communicating with the suppliers on behalf of the constrained

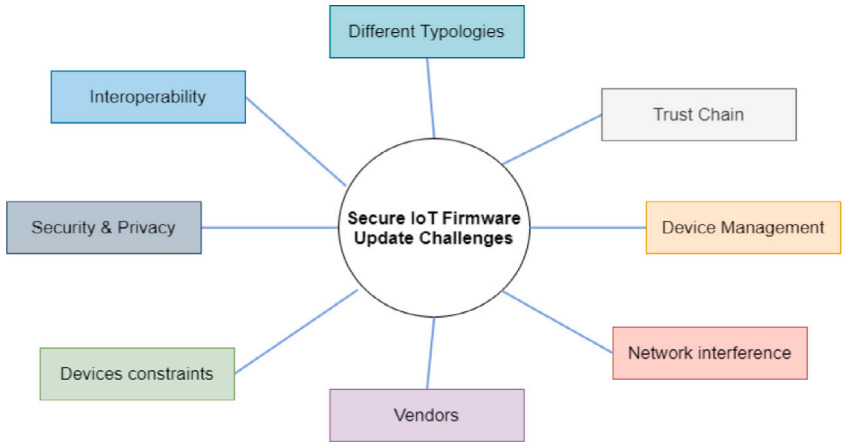


Fig. 2. FoTA update related challenges for IoT devices.

devices. The most common attack is the "**Man-In-The-Middle (MITM)**" where the attacker attempts to spoof the exchanges between the IoT device and the corresponding gateway or application to establish a MITM, in particular during the key establishment process. Upon a successful attempt, the attacker may force the IoT device to accept any malicious FoTA update, since from the IoT device point of view it is coming from a trusted source. The attacker may also attempt to "**Control the mobile application**" by exploiting some vulnerabilities that can be found on Android or iOS systems.

3.3. Target: suppliers and manufacturers

Some attackers may go even higher, by attempting to target the firmware suppliers and manufacturers. They may either try to perform a "**Man-In-The-Middle**" between the suppliers and the gateways/IoT devices, and they may also spoof the supplier's identity, and then "**Tampering**" the firmware with a malicious one. This attack may succeed if there is no integrity verification and no origin authentication. Furthermore, a "**Rogue supplier**", which can be defined as a former supplier, which has been compromised and gone rogue, is a source of threats. In this case, this rogue supplier has access to the users' and IoT devices' details, and even to the signature keys of the firmware which has been sent in previous exchanges with the manufacturer. In this case, the attacker can send as much malicious firmware as he wants with a valid signature and to a large database of targets [18].

4. Challenges

The FoTA update for IoT devices, and in particular for the most constrained ones, is an arduous issue. Several challenges need to be solved to guarantee the robustness and the completeness of a universal generic solution compatible with most IoT devices. In this section, we highlight these challenges to give a clear idea about the complexity of the issue, as shown in Fig. 2. Furthermore, a properly designed, implemented and deployed solution that takes into account the different challenges, can be even extended to manage not only the FoTA updates but also all other aspects in IoT environment such as data management and security.

4.1. The constraints of the IoT devices

This is the most blocking challenge in the IoT world. Most of the IoT devices (e.g., small sensors and actuators) do not have enough capabilities to securely conduct the FoTA update process, in particular for the integrity and authenticity checks. This challenge is more arduous for classes C0 and C1. However, with the advancement in the SoC production, it becomes possible to have some cryptographic hardware in the SoC of the IoT devices, allowing hence the execution of some basic security checks. We argue that the classes C2 and higher are the ones that may have enough capabilities to perform the security checks. However, it is still up to the manufacturer to embedded some secure elements in the device, since, as shown in Fig. 1, even C5 devices may not have some secure elements due to the slightly high cost of adding cryptographic hardware. However, solutions including delegating the security tasks to a trusted intermediary (e.g. a gateway) are currently one of the most adopted approaches. Moreover, other issues need to be managed to be also compliant with all types of IoT devices, such as the battery-powered devices that sleep most of the time and the management of task interruption of the IoT devices (in particular for the ones that continuously sends sensitive data to a remote entity and that cannot be easily interrupted) in addition to the availability issues.

4.2. The security and privacy

As it was highlighted all over this paper, having a secure update is the main objective. For the IoT device, the two most important properties that need to be guaranteed are the **integrity** of the received firmware at all the levels (i.e., gateway, the application layer

of the IoT device and at the bootloader level) and the **authenticity** of the received firmware to guarantee that the received firmware is coming from the right origin and not from a malicious peer. The latter can be guaranteed by verifying the signature in the supposed signed firmware. Moreover, to have a robust update system, **end-to-end** security needs to be established from the supplier to the end devices, while passing by intermediaries. This can be done by first having an IoT device capable of securing the boot and the exchanges (or through a trusted intermediary) and by deploying an adaptive **key distribution system** to share the keys between all the entities all along the supply chain.

Furthermore, the **privacy**, which is currently underestimated yet mandatory under the new GDPR law in Europe, needs also to be taken into account, in particular when dealing with sensitive data of the users such as health-related data. For instance, it is important to protect the privacy of the user's data stored in the IoT flash memory, where sometimes sensitive data such as private login/password resides in. These sensitive data are subject to multiple attacks such as via malicious code during the update process, or physical attacks.

4.3. Interoperability

The heterogeneous nature of the IoT makes it hard to conceive a single and standardized solution for all the IoT devices. The heterogeneity of the IoT comes first from the different types of IoT devices deployed all over the network. These devices, as mentioned in Fig. 1, may belong to different classes. They can interact using different communication protocols at different network stack levels, some of them are proprietary and lots of them are standardized but still siloed (e.g., BLE, Zigbee, WiFi, Lora, Zwave, CoAP, etc.). Furthermore, they may belong to different architectures, that can be either proprietary or standardized such as the case for OneM2M, LwM2M, IoT-A, Web of Things, etc., and that are not interoperable, which also creates silos of devices. Moreover, another interoperability issue is encountered with the different applications (most of the time mobile ones) used to interact with the IoT devices are obviously not interoperable for either manufacturer reasons or security reasons since they have to be executed in a separated sandbox to protect the mobile and the other applications from being attacked by a malicious application.

4.4. Different network topologies

To establish the right networking technology for a specific application, it is important to first understand the network architecture, or the network topology, that is supported by each technology standard. The communication technologies being used today in IoT can be categorized into three basic network topologies: point-to-point (P2P), star, and mesh [20,21].

- A P2P network establishes a direct connection (unidirectional or bidirectional) between two network nodes. Communication can only take place between these two involved nodes, or devices. An example of this type of network is Bluetooth Low Energy (BLE).
- Star topology: each node is connected to an intelligent central hub/gateway with a point-to-point connection, the other nodes are indirectly connected to each other with the help of the hub. SigFox supports start network topology.
- Mesh topology: is a type of networking where all the nodes cooperate to distribute data in a network. This topology is typically used for use cases such as home automation, smart HVAC control, and smart buildings. The communication protocols that use this topology include 6LoWPAN, ZigBee, Z-Wave, and Thread.

Updating the firmware for the P2P topology is much easier than in the other ones. Since in star and mesh topologies, a trust chain must be established from the source to the destination.

4.5. Device management

Being able to efficiently manage the various IoT devices in a particular network is an important step toward a complete and robust FoTA update process for IoT devices. The device management system needs to first be able to efficiently discover and identify every device in the network to interact with the latter when there is a new update available. The system needs to be able to communicate with heterogeneous IoT devices by implementing different communication stacks. Moreover, it has to periodically update the status of the IoT devices, and in particular, the firmware version installed in the devices, using a status tracker. Last but not the least, it needs to recover some proof of installation from the IoT devices to guarantee that the latest firmware image has been properly installed. The device management system will then be able to update the different underlying devices when a new firmware image is available from a supplier. It can also have some rooting capabilities, in order for instance to monitor the traffic, apply some filtering or isolate the network in case of detection of a malicious or infected IoT device.

4.6. Vendor

In some cases, manufacturer's willingness to patch their vulnerable IoT devices can be a blocking point, mainly for economical reasons, since updating the IoT devices means deploying a scalable infrastructure for distributing the firmware images to the different IoT devices. Moreover, these devices may cost very few dollars (i.e. an average price of 0.44$ in 2018 according to Microsoft's "2019 Manufacturing Trends Report" [22]), that they do not see interest in updating them, even though the cheapest ones are the main source of threats.

4.7. Establishing the trust chain

From our point of view, establishing a trust chain from the supplier to the IoT device, while going through multiple intermediaries, is a possible solution for a secure FoTA update over-the-air. In the best case, the establishment of the trust chain can be easily done if there is mutual authentication in addition to end-to-end security mechanisms. In this case, each point in the network can trust the authenticated node and can securely exchange data with it. The secure element for this intent is an important component, toward the establishment of the trust chain, since it brings different cryptographic functions in order to secure the IoT device. However, due to the different challenges mentioned earlier, not all the nodes are capable of implementing such mechanisms. In this matter, the delegation of the computation-consuming operations to a trusted third party can be a solution with the purpose of setting up the trust chain.

This third party can be for instance a gateway in the smart home or the home router, provided by an ISP, that manages all the devices from the constrained ones to the most capable ones. Ideally, it has the capability to identify precisely and to discover the new devices introduced to the local network. Moreover, in order to evaluate each element in the network, it can first audit the different IoT devices before introducing them in the local network. This audit concerns mainly the origin of the device, the different vulnerabilities linked to both the device itself and to the firmware version. The latter leads to a trust evaluation, where a trust score is attributed to each element in the supply chain from the IoT device to the final destination (i.e., the firmware supplier in the FoTA update case). This score will then be a basis for the creation of the trust chain. Finally, in order to keep the network secured, an anomaly detection system needs to be deployed in the network, by monitoring the behaviors of the different IoT devices. Bad behavior must be identified and IoT nodes need to be either updated or isolated from the network in order to stop the spread of the attack if it is the case.

5. State of the art

In this section, a survey of the multiple works done around the FoTA update for IoT devices is provided. It covers the literature review of papers published in digital libraries, works presented by the standardization bodies, and finally some specifications provided by well-known industrial in the SoC world.

5.1. State of the art literature

The authors of [23] propose a Secure Code Update By Attestation in sensor networks (SCUBA), which can be used to repair a compromised sensor through firmware updates. SCUBA utilizes an authentication mechanism and software-based attestation to identify memory regions infected by malware and transmits the repair update to replace these regions. However, the attestation technique based on self-checksumming code heavily relies on consistent timing characteristics of the measurement process and the use of an optimal checksum function. Due to these assumptions, SCUBA is not a suitable approach for IoT settings. In [24] The Update Framework (TUF) helps developers maintain the security of software update systems, providing protection even against attackers that compromise the repository or signing keys. TUF provides a flexible framework and specification that developers can adopt into any software update system. However, this system is more adequate for more powerful devices, and also not suitable for IoT devices. Furthermore, in [25] the authors propose Securing Software Updates for Automobiles (Uptane). It is based on TUF for automotive systems by adding a director repository. This repository allows an original equipment manufacturer (OEM) to have more control of software images deployed in individual Engine Control Units. [26] is another work regarding the secure FoTA update for automotive Electronic Control Units. The solutions employ the separation of entities (e.g., OEM, Apps provider) and the verification of the system's integrity and authenticity, together with the employment of firmware versioning and entitlements for each vehicle and its corresponding electronic control units and dependency resolution on behalf of the vehicle. In [27], the authors present a FoTA update system based on the Lightweight mesh network protocol providing a low-power mesh protocol, route discovery, and establishment. The solution is based on Lightweight Mesh (LWMesh) network protocol over peer-to-peer mesh (P2PMesh) architecture. However, in addition to the proprietary aspect of the LWMesh, the proposal misses the security aspects.

The work in [28] presents a mechanism that focuses on securely getting an update to the device through an untrusted network and verifying that the update is authentic. The goal is to make recommendations that would cause minimal increases in processing and memory requirements so that they could be used by IoT devices running on constrained platforms and on constrained networks. Moreover, the authors propose to apply defense-in-depth strategy to incorporate two security controls: (1) secure transport mechanism (CoAP-DTLS) (2) Integrity control (via ECDSA). However, the proposal provides just some general information with no proof nor implementation only some recommendations. The authors in [29] describe a solution for updating home routers since they believe that it is playing the key element in services at home architecture. They propose an over-the-air firmware update system for home routers and gateways cooperating with a network management system (NMS) and operations support system (OSS) in Chunghwa Telecom (CHT). Yet, the solution is not adaptable for the IoT device itself. [16] presents a FoTA procedure for an IoT device ecosystem and defines a new secure object called Firmware Object Signing and Encryption (FOSE). The object is encoded using a secure representation/format such as JOSE (JSON Object signing Encryption). The main objective is to solve two problems: retransmission (in case of network loss or break) and security (by encrypting the payload) for FoTA. This paper also proposes a simple procedure for over-the-air updates. However, the proposal does not provide implementation nor evaluation (e.g., the overhead of the encryption, etc.) of the work. Moreover, it addresses only the case where there is an intermediary application between the end-device and the device manager.

In the work [30], the authors propose the use of PUFs (Physical Unclonable Functions) inside the BLE hardware of IoT devices to improves the security of firmware updates, and particularly for key generation, because the devices can be uniquely identified at low cost. In particular, using the entropy generated by their SRAM (Static Random Access Memory), the devices can store all the secret keys obfuscated instead of being stored in clear, and they can derive new fresh keys from the previous ones. Experimental results with BLE chips of IoT devices confirm that the obfuscated data do not reveal sensitive information and that the keys needed by the proposed protocol are successfully reconstructed by only the original device. In addition, the nonces generated by the SRAM device are adequate to be used in Key Derivation Functions according to NIST. The CC2541 BLE hardware from Texas Instruments is used. Moreover, the proposed trustworthy firmware update contemplates a registration phase during the manufacturing process, in a secure environment, in which the first symmetric keys are established between the IoT devices and the server. However, the proposal addresses only the devices that use SRAM PUFs in their Bluetooth Low Energy (BLE) system on chips, which is not generic to other hardware and protocols. In [31], the authors present a security framework that includes both JTAG security and secure FoTA update. The passwords for FoTA updates are derived from random values stored in One-Time Programmable (OTP) memory and also unique for a given device. OTA passwords are safe because they are stored in the primary bootloader program which is stored in secure memory and only JTAG can access it. The unique password for every device and restricted JTAG access makes the system more secure in the field. The proposed framework consists of two sections: JTAG protection section and firmware update section. The JTAG section comprises of conventional JTAG 1149.1 circuit with lock/unlock mechanism. The lock/unlock mechanism consists of True Random Number Generator (TRNG), OTP memory, comparator and logical AND gate. Moreover, to implement and validate the proposed security framework, the authors have chosen openMSP430 microcontroller [32], AES-CBC, RSA and bootcore. However, the solution does not address the firmware update in the holistic approach and focuses only on the SoC.

The authors, in [33], designed an architecture for a secure software update of realistic embedded devices (ASSURED), in order to provide a secure and scalable update framework for IoT. They mention that ASSURED includes all stakeholders in a typical IoT update ecosystem while providing end-to-end security between manufacturers and devices. To demonstrate its feasibility and practicality, ASSURED is instantiated and experimentally evaluated on two commodity hardware platforms HYDRA and ARM TrustZone-M. The proposal is an enhancement of the TUF architecture, and adopts the same terminology as proposed in Software Updates for IoT (SUIT) Working Group, to propose end-to-end security, update authorization from the controller, attestation of update installation, the protection of the code and secret keys on the device, and the minimal burden for the device. However, the authors did not take into account the scalability issues when dealing with hundreds of IoT devices, for instance to guarantee the attestation of update installation. Moreover, they did not take into account the heterogeneity of the IoT devices and the local connectivity issues (e.g., when there is no security over BLE or RF, etc.), and they do not take into account the case where the IoT device is very constrained and without the possibility of securing the device. In [34] the authors exposed first state-of-the-art research in secure firmware updates and then they provided four-element update theoretical model for firmware update. In this model, the process starts with the packing element where the firmware is developed and complied by an author or manufacturer in addition to the key generation and signatures. Then, the delivery element, to deliver the firmware to the devices, which can be over encrypted or unencrypted channels. Next, is the authentication element, which involves checking signatures for authenticity verification. And finally, the attestation element, in order to ensure that not only the correct firmware has been loaded but also that the device itself is not in a malignant state. However, we argue that several elements for a secure FoTA updates are missing to name a few the end-to-end security and secure bootloading. In addition to the absence of a concrete implementation and evaluation.

The paper in [35] first survey open standards and open source libraries that provide useful building blocks for secure firmware updates for the constrained IoT devices. Then, they proposed a solution to create a secure, standards-compliant firmware update solution that uses state-of-the-art security for the constrained IoT devices with less than 32 kB of RAM and 128 kB of flash memory, by implementing the new IETF standard called SUIT. The work has been implemented using a single OS (i.e., RIOT) on several constrained IoT devices such as Atmel SAMR21, STM32F103REY and Nordic nrf52840. However, the authors did not address the end-to-end security, and they did not test their solution on other OSs, which can show different behavior and update patterns. Finally, they focused only on a single communication protocol (i.e., 6LowPAN). [36] proposes a unified scheme to manage the IoT devices in user environments, this scheme is a user-centric one where the user needs to provide consent to update the IoT devices. The scheme uses also a status tracker to verify the status (e.g., monitoring, current version, etc.) of all the IoT devices and vulnerability scanners to look into the most known databases for new vulnerabilities related to these listed devices. However, the solution does not specify how to provide end-to-end secure FoTA update. In [18], the authors propose a design and prototype of architecture for onboarding and secure software update of low-level IoT devices. It uses authenticated key establishment via the Chain-of-Custody concept to onboard the IoT device and the gateway, and a known key-locking mechanism that uses a public key of the manufacturer embedded in a current software image to verify the signature of the next software update. Moreover, the model demonstrates that EC-based public-key algorithms can be used with constrained devices such as ATMega2560 8-bit microcontroller with 16MHz clock. However, the proposed schema involves the initialization and addition configurations, which may be complicated. Moreover, it supposes that there will be a single application for controlling all the IoT devices and that all the communications go through a gateway which is not the case currently.

In [37], the authors propose UpKit a portable and lightweight software update framework for constrained IoT devices encompassing all phases of the update process: (1) the generation phase, where the firmware and the corresponding manifests are generated and signed by the vendor server, (2) the propagation phase, where the update server recovers and double signs the signed firmware and manifest and then sends them OTA to a gateway via a pull method (eventually possible with push), the latter relay them to the update agent present in the IoT devices, (3) the verification phase, which verifies the freshness, the authenticity and the integrity of the data before sending them to the bootloader, to reject invalid software at an early stage, and then reboots, (4)

finally the loading phase, which consists in preparing the image to be executed (i.e., moving it to the right memory address). This paper [38] describes the different key steps in an FoTA update process together with a quantification of the energy overhead per deployment phase and the energy impact. The authors resume the update process in two distinct phases, each with multiple steps. The first one is "the software module management phase" which is performed offline to minimize the impact on the deployment network. They claim that using the combination of a compatibility matrix and a digital twin network, it is possible to identify version incompatibilities, bugs, and performance issues even before the update is executed. The second phase is "the secure software rollout", which elaborates on the eventual rollout of software modules to devices, quantifying the energy overhead per step (during encryption, authentication, integrity verification, data dissemination, firmware installation, and during the activation step). Their results show that besides the obvious dissemination cost, other phases such as security also introduce a significant overhead.

In [39], the authors present an example of using the ARM TrustZone technology to have a secure FoTA update. It employs the RSA algorithm to verify the image signature in addition to an application running in the secure OP-TEE to verify the authenticity and the corruption. However, the solution is not generic and not tailored for IoT devices with very constrained capabilities. The work in [40] presents a firmware update OTA prototype for mesh networks. In this case, one node (the transmitter) can update the firmware of the nearby sensors (the receivers). The authors also built a custom bootloader to support both the FoTA updates and the normal booting process. Moreover, to secure the exchanges, they use a hardware-based AES-CCM module for encryption and authentication. However, there are no authenticity and integrity checks and there is no protection against a malicious node in the network. [41], describes a set of challenges that face the update procedure in IoT environment. It also analyzes the SUIT work regarding this topic. It also provides an analysis of some Blockchain-based solutions.

Table 2 compares the previous works by taking into account the following criteria:

- The "**Security at the firmware generation**" column, concerns the consideration of securing the new firmware image and the corresponding metadata (which will be referred to as *update package*). It concerns mainly the digital signature of the firmware and optionally the distribution or the pre-sharing of the public key with the IoT devices, even though it is the responsibility of the IoT device providers. The presence of at least one of these considerations is expressed with a 'Yes'. In this case, some keywords of the used cryptographic algorithms are provided if specified in the research paper.
- The "**Security during the firmware propagation**" column, shows if the solution assures that the transport of the FoTA update package is done over a secure channel. In this case, some keywords of the used protocol are provided if specified in the research paper.
- The "**Firmware Verification**" column, indicates if the proposed solution assures the presence of firmware verification, which includes mainly the authenticity and integrity verification of the received update package.
- The "**Secure Bootloading**" column, indicates if the proposed system can determine if the underlying IoT devices can securely receive, validate and install the update package.
- The "**Multi-hardware**" column, provides the hardware on which this solution was tested on. If tested on a single piece of hardware, then the system is considered as "Not generic" since it is specific to this hardware, otherwise, if tested on multiple hardware, it can be considered as more or less "Generic" (the more supported by hardware the more it can be Generic).
- The "**Multi-OS**" column, provides the OSs on which this solution was tested on. If tested on a single OS, then the system is considered as "Not generic" since it is specific to this OS, otherwise, if tested on multiple OSs, it can be considered as more or less Generic (the more supported by OSs the more it can be Generic).
- The "**End-to-End Security**" column, indicates by a 'Yes' or 'No' if the proposed solution provides end-to-end security.
- The "**Class 0 to 2 Compatibility**" column, indicates if the solution was tested on very constrained IoT devices. Since some works are tested on less-constrained IoT devices, belonging to class 4 and higher, such as raspberries pi 3 and 4. They are also considered as IoT devices, but nowadays have enough capabilities to perform a secure update by themselves. In this case, some keywords of used microcontrollers are provided if specified in the research paper.

5.1.1. FoTA updates for IoT devices via Blockchain

The Blockchain is a decentralized way of data storage that replicates the data into multiple systems to overcome the danger of a single point of failure, which the conventional server-based architectures is exposed to. The following works exploit the Blockchain capabilities to guarantee the security of the FoTA updates for IoT devices.

In [42], the authors present a Blockchain-based solution for managing firmware updates in IoT. Its objective is to verify and distribute the firmware binaries securely to the IoT device deployed by the device vendor. The main purpose is to solve the single point of failure issues that occurs in typical centralized architecture for firmware update. The system is based on the Ethereum Blockchain platform, which comprises the creation of a smart contract, the distribution and verification of the firmware binaries. The verification process applies the Proof-of-Work (PoW) consensus algorithm and the concept of hash chain to check the smart contract and the integrity of the firmware binaries. However, in the end, all the gateways and eventually IoT devices will pull the firmware from the specified URI which may create again the single point of failure. A similar work has been presented in [43] based on the Blockchain Ethereum. They also propose a kind of delegation, to provide indirect firmware updates, using a broker and a replication of the contract mechanism. However, the purpose of using this indirect method was not explained and the work was not implemented nor tested. The paper in [44] proposes a solution for both complete firmware updates and delta updates using private Blockchain. Basically, in order to update, the information regarding the firmware available on the IoT device is compared against the information stored in the Blockchain server, in addition to integrity checks using hash functions. These hashes are also stored in

Table 2
State of the art comparison.

Ref	Security at the firmware generation	Security during the firmware propagation	Firmware verification	Secure bootloading	Multi-hardware	Multi-OS	End-to-End Security	Class 0 to 2 Compatibility
[18]	Yes: Ed25519 and RSA	Yes: Ed25519, RSA, and AES	Yes	Yes	Not generic	Not generic	Yes	Yes: ATMega2560
[23]	Yes	Yes: Guy-Fawkes protocol	Yes	No	Not generic	Not generic	No	Yes: TI MSP430
[24]	Yes	Yes: TLS	Yes	No	Not generic	Not generic	No	No
[25]	Yes	Yes: TLS	Yes	No	Not generic	Not generic	No	No
[26]	Yes: ED25519	No	Yes	No	Not generic	Not generic	Yes	No
[27]	No	No	No	No	Not generic: specific for Atmel devices	Not generic	No	Yes: SAM-R21 Atmel boards
[28]	No	Yes	Yes	No	Not generic	Not generic	No	No
[29]	No	No	Yes	No	Not generic	Not generic	No	No
[16]	Yes	Yes: CoAP-DTLS	Yes	No	Not generic: not implemented	Not generic: not implemented	No	No: not implemented
[30]	Yes	Yes: AES and MAC	Yes	No	Not generic	Not generic	Yes: symmetric key	Yes: TI CC2541 keyfob
[31]	Yes: RSA	Yes: AES	Yes	Yes	Not generic	Not generic	No	Yes: TI MSP430
[33]	Yes: ED25519	Yes: AES	Yes	Yes	Generic: I.MX6-SabreLite and ARM V2M-MPS2	Not generic	Yes	No
[34]	Yes	No	Yes	No	Not generic: not implemented	Not generic: not implemented	No	No: not implemented
[35]	Yes: Ed25519	No	Yes	Yes	Generic: Nordic nrf52840, Atmel SAMR21 and STM32F103REY	Generic	No	Yes: Atmel SAMR21
[36]	No	No	No	No	Generic: ARM and MIPS related CPUs	Not generic	No	No
[37]	Yes: ECDSA	No	Yes	Yes	Generic: nRF52840, TI CC2650 and TI CC2538	Generic: Contiki-NG, RIOT, and Zephyr	No	Yes: TI CC2650
[39]	Yes: RSA	Yes	Yes	Yes	Not generic	Not generic	Yes	No
[40]	No	Yes: AES-CCM	No	Yes	Not generic	Not generic	No	Yes: TI CC2650

the Blockchain server. In the work [45], the authors present an extension of the basic firmware update scheme provided by the Open Connectivity Foundation (OCF) [46] by adding a Blockchain solution. The objective is to guarantee the integrity, the availability, to avoid the single point of failure and man-in-the-middle issues. The authors propose to use the smart contracts in order to store the firmware binary in the Blockchain, after a verification step of the new firmware and the inclusion of the smart contract in the Blockchain network. They propose two ways of updating the IoT gateways, which act on behalf of the IoT devices (OCF Device). The first one is a direct update from the manufacturer, and the second one is in a peer-to-peer (p2p) fashion from the neighbor gateway nodes. The second p2p firmware update happens between devices of the same type and from the same manufacturer. [47]

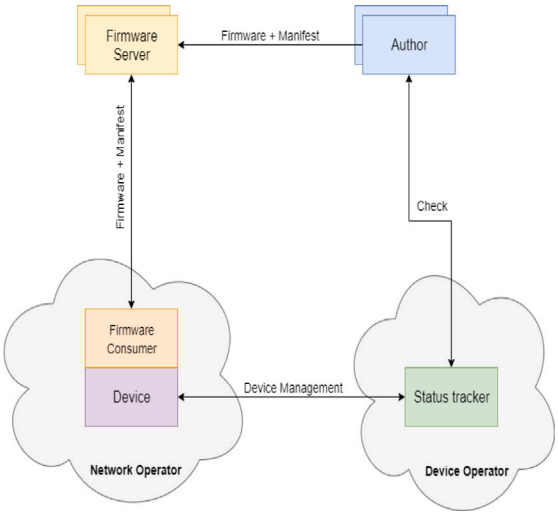


Fig. 3. SUIT general architecture [2].

is a work that enhances the Lora Alliance FUOTA procedure [48], as it will be explained later in 5.2.4, by introducing the Blockchain smart contract to securely update the firmware of the IoT devices using the Lora protocol. The smart contract is used to verify the authenticity and the integrity of the firmware (precisely the header of the firmware image which contains parameters such as the nodeID, the firmware version, the checksum, etc.). The authors also concluded that there is a need to use more gateways that will collaborate to increase the reliability and the performance of the firmware update procedure.

5.2. State of the art of the standardization bodies' works

In this section, a special focus is given to the works presented by the different standardization bodies.

5.2.1. IETF

The RFC8240 [49] summarizes the Internet of Things Software Update (IoTSU) Workshop in 2016, whose main goal is to discuss the challenges, requirements and solutions related to this topic. It also provides multiple open challenges and relevant questions toward having a future generic platform for updating the billions of heterogeneous IoT devices. Some of these requirements are:

- *Authentication*: new firmware images must be accepted only from the authenticated sources, usually done via the public key.
- *Version Control*: only the newest and the freshly generated FoTA updates must be accepted, and additionally it needs to be intended for the particular device.
- *Integrity*: checking that the firmware was not tempered or changed during the OTA update process.
- *Reduced users interaction*: since the user is a good source of errors and also may not be expert enough to perform complex operations. However, their consent must be taken into account to respect their privacy.
- *Platform and OS independent*: it must be generic enough to work in most of the well-known hardware and OSs.
- *Scalability*: to be able to update the exponentially evolving number of IoT devices efficiently.

Also, the IETF Draft [2] aims at standardizing the FoTA update process for IoT devices. It lists the different requirements and describes an architecture for a FoTA update mechanism suitable for IoT devices. The architecture is agnostic to the transport of the firmware images and associated metadata. Such a process has to ensure mainly image confidentiality, authenticity and integrity. The document tries also to unify the terminology used for the different entities and elements involved in such an update process. It provides also examples of typical message flow for FoTA updates starting from the publishing of a new firmware version.

The Fig. 3 shows the proposed SUIT architecture, which encompasses the firmware image creation by an author, and it upload to a firmware server. The firmware image and the manifest are then distributed to the device, either in a push or pull mode using the firmware consumer in the IoT device. The device operator follows all the processes using the status tracker, which allows the device operator to know and control what IoT devices have received a new firmware for update and which of them are still pending one [2].

The manifest encodes the information that the devices need to make the necessary FoTA update decisions. It is a data structure that contains the following information among others [2,50,51]:

- Information about the device(s) the firmware image is intended to be applied to.
- Information about when the firmware update has to be applied.

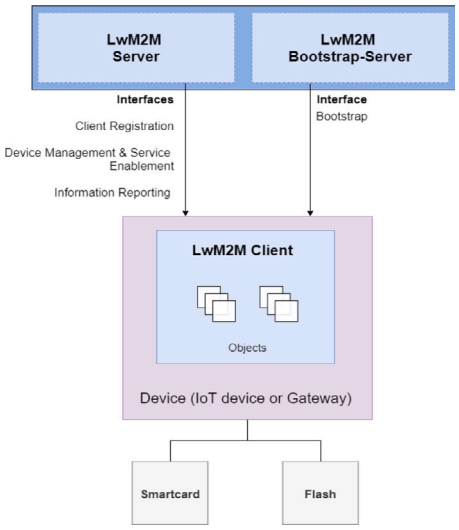


Fig. 4. The overall architecture of the LwM2M Enabler [52].

- Information about when the manifest was created.
- Dependencies on other manifests.
- Pointers to the firmware image and information about the format.
- Information about where to store the firmware image.
- Cryptographic information, such as digital signatures or message authentication codes (MACs).

5.2.2. OMA LwM2M

Lightweight M2M (LwM2M) is a device management protocol designed to be a common standard for managing lightweight and low power devices on a variety of networks, necessary in order to realize the potential of IoT, in particular for the machine-to-machine (M2M) environment. It is specified by Open Mobile Alliance (OMA) SpecWorks Device Management Working Group, and the architecture is based on Constrained Application Protocol (CoAP) and the underlying security (i.e., mainly DTLS).

The Fig. 4, represents the overall architecture of the LwM2M Enabler for IoT devices [52]. It describes four interfaces between the main entities in the LwM2M IoT environment (i.e., the LwM2M server and the IoT devices or the corresponding gateway). These four interfaces are:

- *Bootstrap*: allows multiple operations to initialize the different objects for the LwM2M client (i.e., a client can be an IoT device or a gateway) to register with one or more LwM2M Server.
- *Client Registration*: defines the different exchanges to register the LwM2M client with the LwM2M server.
- *Device Management and Service Enablement*: specifies the different operations that allow the interaction with the resources and objects exposed by the LwM2M client (e.g., the "Discove" operation, which is used to discover attributes and to discover which resources are implemented in a certain Object).
- *Information Reporting*: specifies the set of possible operations to observe the LwM2M client and to notify the LwM2M server.

The full specification is available in [52]. However, LwM2M focuses only on the CoAP protocol and the related security. It specifies that the LwM2M client needs to have a Smartcard, and that the bootstrap information, which also contains the different keys and certification, are transported via a secure channel from the LwM2M client to the Smartcard for secure provisioning.

5.2.3. OCF

Open Connectivity Foundation (OCF), is an IoT standardization body, whose objective is to ensure interoperability and to provide a global and generic IoT platform for consumers and industries, via standardized communication platform and open-source implementation. They also provide solutions regarding the firmware updates of the IoT devices (i.e., namely OCF devices) from the external update server (e.g., the manufacturer). OCF specifies three triggers that can be performed by the OCF client: (1) checking the availability of a new software (2) downloading and checking the integrity of the software package (3) installing the verified software package, as specified in [46]. However, the provided interactions between OCF and the external server are limited to only checking and retrieving if a new software update is available. Thus, missing some important security checks and mechanisms to guarantee a smooth and secure firmware update.

5.2.4. Lora Alliance

In [48], Lora Alliance summarizes the components and the process to update the firmware of an IoT device on top of the LoraWAN protocol. In this case, the update process is performed at the application layer, and is called FUOTA (Firmware Update Over-The-Air). The architecture together with the sequence of actions to achieve the FUOTA is described in [48]. In this process, the main entities are:

- Firmware Update Server (FUS): responsible for generating the new firmware image or delta-image. The information related to the image is embedded in the image header such as the target hardware version, the current firmware version, image CRC, etc.
- Application Server (AS): among the other modules, the Firmware Management (FM) and the File Distribution Server (FDS) are the ones responsible for the firmware update process. The FUS interfaces mainly with the FM of the AS. The FDS is specifically tasked to deliver the firmware image to the end-device. The FDS knows all the identifiers of the end-devices that need to be updated. It also communicates with the NS to negotiate parameters of the transmission windows for the LoRaWAN classes, fragmentation, list of end-devices, etc. Then, the FDS sends the fragmented firmware image file to the Dev via the NS (via unicast or broadcast).
- End-Device (Dev): among the other modules, at the App Stack the FM and the File Distribution Client (FDC), and at the Secure Bootloader, the Firmware Update Agent (FUA) are the ones linked to the firmware update process. The FDC is the counter-part of the FDS on the end-device side. The Dev downloads and reconstructs the fragments of the binary firmware image. It checks the signature of the image for integrity, authenticates the FUS using the public key stored in the end-device, and checks the compatibility of the hardware and versions. Finally, it proceeds to the secure boot. The end-device can optionally send the new firmware version currently running to the FUS.
- Network Server (NS): manages the network part of the LoraWAN, including the multicast case.

5.3. State of the art industry

It is also important to report that there are huge advancements from industrial companies to provide a secure FoTA update for IoT devices, in particular after the multiple attacks that impacted some huge accounts and that required quick and urgent interventions to stop them, to mention “Mirai attack” [53]. In addition to the solutions provided by the semiconductors and microcontroller constructors, there are also the solutions provided by some of the huge companies in FAANG (also GAFAM) (Facebook, Amazon, Apple, Netflix and Google) and in BATX (Baidu, Alibaba, Tencent and Xiaomi). This section highlights some of these solutions to have a global view of the technical advancements in the IoT’s FoTA update domain.

5.3.1. Google Cloud IoT Core and Mender

Cloud IoT Core [54] is a Cloud-based product provided by Google, which mainly allows a secure device connection and management from a few to millions of IoT devices. It provides a solution for collecting, processing, analyzing, and visualizing IoT data in real-time to support improved operational efficiency. Among other functionalities, it allows the state monitoring of each connected device. Upon a configuration update from Cloud IoT Core, an update from another external source such as a FoTA update from the manufacturer, the device state update in the platform is triggered.

In a Google Cloud blog [55], there is also a solution that combines the Cloud IoT solution with Mender, which is an open-source project that aims at providing robust system-level updates as well as application updates for IoT devices. It adopts a client–server architecture, where the main components are the management server (Mender Server), which stores and controls the deployment of the OTA software updates via REST APIs, and the IoT device (Mender Client), which reports to the server periodically to check for new updates. The download of the new update is done via HTTPS. At the IoT device level, a dual A/B roots partition is required in order to migrate from the old installed version to the new one in a smooth way. In the reference design, a U-Boot is used as a bootloader. The solution uses HTTPS polling so no ports are open on the device in addition to the code signing to guarantee confidence and integrity [56].

5.3.2. AWS IoT device management

Is a solution provided by Amazon [57] to register, organize, monitor and remotely manage IoT devices. Among the multiple functionalities it has, the solution provides means to query the states of the managed IoT devices and to send FoTA update, mainly for FreeRTOS devices. The solution allows to digitally sign the new firmware before its deployment to either a single or to a fleet of IoT devices. It also enables the authenticity and integrity verification of the newly received firmware image after its deployment on the IoT device and also monitor this deployment, to avoid any failure cases.

5.3.3. Nordic BLE Device Firmware Update

For its Bluetooth-based devices, Nordic semiconductors provide a Device Firmware Update (DFU) [58] service for updating IoT devices OTA over Bluetooth low energy. In this process, two devices are required. The first one is the DFU target, which is the devices concerned by the firmware update, and the second is the DFU controller, which is the device that transfers the firmware image. The latter can be for instance a mobile phone running an application or an nRF5 development kit. The full procedure is explained in [58].

5.3.4. Texas Instruments OTA Downloads

Over the Air Download (OAD) [59] is an ecosystem provided by Texas Instruments (TI) that allows the update of the firmware image running on BLE devices wirelessly. It is also based on two main components: (1) the OAD target, corresponding to the IoT

device whose firmware is being upgraded over the air and runs the OAD service (which is basically a GATT server) and (2) the OAD Downloader, which receives the firmware image and then transfers it OTA to the target together with some OAD metadata.

5.3.5. ESP32 OTA updates

ESP provides a set of tools that simplify the FoTA updates using the so called ESP-IDF (Espressif IoT Development Framework) [60]. Its main advantage is that it allows the update of the firmware running on an ESP32 chip using only functions included in the esp-idf framework. Hence, it simplifies the task and avoids the use of any external tools or platforms. The framework offers a set of native functions to implement the OTA update in [60]. It can be used by including the corresponding header "esp_ota_ops.h", present also in the [60]. Furthermore, for more security, there exists a set of APIs called "ESP HTTPS OTA" [61] that allows the use of HTTPS to update the ESP chip using firmware present in a remote location. The APIs allow to automatically identify an OTA partition in the flash memory which is not in use and then to save the newly updated firmware in that partition. Finally, it configures the chip to boot from that partition.

5.3.6. ARM mbedOS OTA updates

"Mbed OS is an open-source operating system for platforms using Arm microcontrollers designed specifically for Internet of Things (IoT) devices: low-powered, constrained devices that need to connect to the internet. Mbed OS provides an abstraction layer for the microcontrollers it runs on, so that developers can focus on writing C/C++ applications that call functionality available on a range of hardware. Mbed OS applications can be reused on any Mbed-compatible platform" [62].

In the newest version of Mbed OS (currently at v5.15), **Pelion Device Management** [63] is used to interact and manage the connected devices that implement Mbed OS, in a simple, flexible and secure way. Among its services, it enables the provisioning and connection of IoT end nodes with cost-effective, and provides secure and reliable software updates from remote locations OTA, ensuring, hence, a long product lifetime. Moreover, ARM provides a secure firmware update solution via a security model that describes the responsibilities and relationships within the system. Security in firmware updates uses public-key cryptography to verify an update validity, ensuring the authenticity of the firmware (i.e., manifest and image). Moreover, an authenticity certificate, signed by a Certificate Authority (CA) or simply self-signed, is stored and used for verifying firmware images. In this case, the trust Anchor is a public key, pre-installed on the device (at the manufacturing time) [64].

5.3.7. Matter, formally (Connected Home over IP (CHIP))

The three American giants (Google, Amazon and Apple) have joined forces with the Connectivity Standards Alliance (formerly Zigbee Alliance) to increase the interoperability of connected objects in the home using voice services, with a special focus on security. The new working group is called Matter [65], formally "Project Connected Home over IP" [66]. The main goal of this project is, on one side to improve the user experience, by allowing different and incompatible IoT devices to communicate with each other, and on the other side to simplify the development for manufacturers to build IoT devices secure, reliable and compatible with the smart home and voice services such as Amazon's Alexa, Apple's Siri, Google's Assistant, and others.

This new protocol, also known as Connected Home over IP, will be IP-based and will operate in the 2.4 GHz frequency band, and possibly on 5 GHz. It will only work over IPv6, so IoT devices with IPv4 addresses will not be compatible. A first specification planned for the end of 2020, first for WiFI, up to and including 802.11ax (aka Wi-Fi 6), that is 802.11a/b/g/n/ac/ax, will make the Zigbee network compatible with Wi-Fi, Thread and low energy Bluetooth standards, before subsequently integrating Ethernet or cellular networks. The already well-developed and mature IP-based system is expected to bring better connectivity and security (e.g., end-to-end security and privacy).

However, in this concern, several questions raise such as its effect on the non-IP compatible devices and also ones that use only IPv4, since they will not be addressed in this new project, the silos of devices it will create and the privacy issues. Nevertheless, the solution can simplify the specification and the deployment of a secure and generic FoTA updates over IP based devices, by taking advantage of the already mature solutions used in IP based solutions, and by adapting them to the constrained environment of IoT.

5.4. Orthogonal solutions to the IoT FoTA updates

In addition to the FoTA update works, it is interesting to highlight some enablers that can help in facilitating the update procedure, and related to the device management topic. For instance, in [67], the authors propose a solution called CIDRE that can recover IoT devices within a short amount of time, even if attackers have taken root control of every device in a large deployment. The recovery requires minimal manual intervention. After the administrator has identified the compromise and produced an updated firmware image, he/she can instruct CIDER to force the devices to reset and install the patched firmware on the devices. It is based on the notion of Dominance by enabling the owner or administrator of a large IoT deployment to install and run a firmware version of his or her choice by force even if the device firmware is under the control of an attacker and actively resists recovery. This feature can be also an orthogonal solution for quick recovery combined with a secure FOTA solution. And in [68], the authors present an engineering project which is composed of two missions that aim at exploring two complementary technologies: (1) the LWM2M protocol via its client and server Eclipse Leshan implementations, to implement an archetype of an LwM2M-based IoT device, and an operational firmware upgrade functionality and (2) the HawkBit project to deploy a LwM2M-enabled firmware upgrade architecture. The work can be seen as an orthogonal architecture to deal with the device management issues, in addition to a more complete and secure firmware update over-the-air solution for the IoT devices. Moreover, additional work needs to be done to manage also the more constrained devices, since it was not studied in the work.

6. ISP'S point of view

The main role in IoT is the provisioning of communication and connectivity service, in particular in the context of the smart home, which is the main application domain selected in this paper. For this aim, the ISP will be faced will multiple challenges, not only from the previous ones, but also new ones related to the ability to manage millions of IoT devices efficiently and securely, the scalability, the availability, in particular for critical systems such as e-Health and industry 4.0, and most importantly the cost evaluation to deploy and maintain such management infrastructure. Moreover, significant investments in both the end-to-end communication establishment and on the data management are inevitable to support the projected growths in data traffic, generated not only by IoT. Furthermore, additional challenges are added when a radio communication protocol such as NB-IoT and the future 5G is used (e.g. video streaming). In this paper, a special focus is given to the ability of the ISP to control and manage the update process in the IoT environment.

In many cases, the constrained IoT devices rely on a more capable intermediary, which can be a gateway or a mobile application in a smartphone. The intermediary in one side must be able to communicate with the IoT devices using a short range and low-power low-bandwidth communication protocol (e.g., BLE, Zigbee) and with the supplier or the manufacturer using a secure protocol (e.g., HTTPS or CoAP with DTLS), and on the other side, it needs to guarantee the previously mentioned security and privacy requirements for FoTA updates.

The ISP interferes mainly at the gateway level. However, there are multiple use cases where the operator might have different levels of control to guarantee the smooth execution of the FoTA update process:

- The use case where the ISP is also an IoT device provider: means that in addition to the management of the traditional network, it has also the responsibility of managing all of its IoT devices including the connectivity, the control of collected data, the respect of the privacy of the user, and last but not least, the ability to efficiently and successfully update the firmware images of these devices. In this case, the manufacturer/supplier is the ISP, the gateway also is a trusted element and it might have enough capabilities to ensure end-to-end security from the supplier to the IoT devices. Finally, the IoT device hardware and software must conform to the local laws (i.e. GDPR in Europe), by ensuring their compliance with the security and privacy requirements. This is the ideal case where the trust chain from the supplier to the IoT device is established by default. The ISP has control over all the supply chain and can enforce security policies at all levels to protect the system.
- The use case where the ISP manages heterogeneous IoT devices (also from other suppliers) in the user's environment, such as the smart home: in addition to the previous tasks, the ISP might also deploy some additional security mechanisms to protect the local environment of the user and to protect their devices against some malicious infected devices that can be present in the same network. In these circumstances, the establishment of the trust chain become harder since the ISP needs to identify the other IoT devices, which may be constrained devices, exported from other countries, and eventually not conform to the laws of the local country. Moreover, in addition to the identification process, it needs to make sure it communicates with benign entities (suppliers) that are whitelisted. In the worst case, a network separation is necessary to protect the other benign devices. As for the update process, it has to make sure that all the IoT devices have the latest firmware version. For the ISP devices it is simple as mentioned before, however, for the other IoT devices, it needs to make sure that they are correctly updated and this includes: (a) the IoT devices are not malicious (solutions including remote attestation can be tested [69]), (b) checking if the IoT devices can securely and correctly update itself, (c) do not interact with malicious entities, which may harm the local network. In this case, the operator still has full control only over its related infrastructure, yet, since most of the IoT traffic is routed through the ISP equipment, it can apply some additional control over the local network via ACLs for instance.

7. Conclusion and future works

In this paper, we surveyed the state of the art from diverse sources (i.e., literature, standards and industrial solutions) to follow the advancement in the FoTA update domain for IoT devices. We also highlighted the main security threats and the main challenges that face this domain which may open research horizons for future works for both academics and industries. Most of these challenges are still unsolved due to the nature of the IoT environment. However, there is a significant advancement to have an interoperable solution. Furthermore, in future works, a first logical step is to propose an architecture involving interactions between different actors of the FoTA update process, then to design and implement multiple scenarios where an FoTA update is needed. These scenarios should include various types of IoT devices with varying capabilities to achieve a generic solution. Moreover, it should take into account the security capabilities of each device and make decisions regarding how the FoTA update must be conducted. A very constrained IoT device with low capability may rely on the home gateway to securely update its firmware. Furthermore, once these scenarios are completed, a trust chain based architecture can be conceived where it divides and organizes the different IoT devices in the smart home based on a level of trust. Also, an important challenge that should be also addressed in our future works, is the overhead that will add the deployment of these additional security measures in the update process, and its impact on the resources of the IoT devices, in particular the memory footprint.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] European Union Agency for Cybersecurity (ENISA), Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures, November, 2017, http://dx.doi.org/10.2824/03228, URL https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot.

[2] B. Moran, H. Tschofenig, D. Brown, M. Meriac, A Firmware Update Architecture for Internet of Things, Internet-Draft draft-ietf-suit-architecture-08, Internet Engineering Task Force, 2019, Work in Progress, URL https://datatracker.ietf.org/doc/html/draft-ietf-suit-architecture-08.

[3] C. Bormann, M. Ersue, A. Keranen, Terminology for Constrained-Node Networks, RFC 7228, IETF, 2014, URL http://tools.ietf.org/rfc/rfc7228.

[4] ESPRESSIF, Esp32-S2 a secure and powerful wi-fi MCU with numerous I/O capabilities, 2021, URL https://www.espressif.com/en/products/socs/esp32-s2.

[5] N. Semiconductor, Bluetooth 5.2 SoC supporting bluetooth low energy, 2021, URL https://www.nordicsemi.com/products/nrf52810.

[6] S. Microelectronic, Sub-GHz wireless microcontrollers, 2021, URL https://www.st.com/en/microcontrollers-microprocessors/stm32wle5j8.html.

[7] GLOBALPLATFORM, Introduction to secure elements, 2018, URL https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Secure-Element-15May2018.pdf.

[8] NXP, EdgeLock SE050: Plug and trust secure element family - enhanced IoT security with maximum flexibility, 2018, URL https://www.nxp.com/products/security-and-authentication/authentication/edgelock-se050-plug-trust-secure-element-family-enhanced-iot-security-with-maximum-flexibility:SE050.

[9] S. Microelectronic, Stsafe-A1SX secure element for sigfox LPWAN network, 2021, URL https://www.st.com/en/secure-mcus/stsafe-a1sx.html.

[10] S. Zhao, Q. Zhang, G. Hu, Y. Qin, D. Feng, Providing root of trust for ARM TrustZone using on-chip SRAM, in: Proceedings of the 4th International Workshop on Trustworthy Embedded Devices, in: TrustED '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 25–36, http://dx.doi.org/10.1145/2666141.2666145.

[11] F. Siddiqui, M. Hagan, S. Sezer, Embedded policing and policy enforcement approach for future secure IoT technologies, in: Living in the Internet of Things: Cybersecurity of the IoT - 2018, 2018, pp. 1–10, http://dx.doi.org/10.1049/cp.2018.0010.

[12] S.Y. Kouty, Multilayer secure hardware network stack using FPGA, in: 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI, 2020, pp. 439–444, http://dx.doi.org/10.1109/ISRITI51436.2020.9315502.

[13] ARM, Trustzone for cortex-m, 2021, URL https://www.arm.com/why-arm/technologies/trustzone-for-cortex-m.

[14] Microsemi, SmartFusion2 SoC, 2021, URL https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2.

[15] M. Elnawawy, A. Farhan, A.A. Nabulsi, A. Al-Ali, A. Sagahyroon, Role of FPGA in internet of things applications, in: 2019 IEEE International Symposium on Signal Processing and Information Technology, ISSPIT, 2019, pp. 1–6, http://dx.doi.org/10.1109/ISSPIT47144.2019.9001747.

[16] K. Doddapaneni, R. Lakkundi, S. Rao, S.G. Kulkarni, B. Bhat, Secure FoTA object for IoT, in: 2017 IEEE 42nd Conference on Local Computer Networks Workshops, LCN Workshops, 2017, pp. 154–159, http://dx.doi.org/10.1109/LCN.Workshops.2017.78.

[17] A.K. Jain, D. Shanbhag, Addressing security and privacy risks in mobile applications, IT Prof. 14 (5) (2012) 28–33, http://dx.doi.org/10.1109/MITP.2012.72.

[18] H. Gupta, P.C. van Oorschot, Onboarding and software update architecture for IoT devices, in: 2019 17th International Conference on Privacy, Security and Trust, PST, 2019, pp. 1–11, http://dx.doi.org/10.1109/PST47121.2019.8949023.

[19] J. Park, A. Tyagi, Using power clues to hack IoT devices: The power side channel provides for instruction-level disassembly., IEEE Consumer Electron. Mag. 6 (3) (2017) 92–102, http://dx.doi.org/10.1109/MCE.2017.2684982.

[20] S. Al-Sarawi, M. Anbar, K. Alieyan, M. Alzubaidi, Internet of things (IoT) communication protocols: Review, in: 2017 8th International Conference on Information Technology, ICIT, 2017, pp. 685–690, http://dx.doi.org/10.1109/ICITECH.2017.8079928.

[21] M.J. McGrath, C.N. Scanaill, Sensor network topologies and design considerations, in: Sensor Technologies: Healthcare, Wellness, and Environmental Applications, A Press, Berkeley, CA, 2013, pp. 79–95, http://dx.doi.org/10.1007/978-1-4302-6014-1_4.

[22] Microsoft, 2019 Manufacturing trends report, 2019, Microsoft Dynamics 356, URL https://info.microsoft.com/rs/157-GQE-382/images/EN-US-CNTNT-Report-2019-Manufacturing-Trends.pdf.

[23] D. Perito, G. Tsudik, Secure code update for embedded devices via proofs of secure erasure, in: D. Gritzalis, B. Preneel, M. Theoharidou (Eds.), Computer Security – ESORICS 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 643–662.

[24] J. Samuel, N. Mathewson, J. Cappos, R. Dingledine, Survivable key compromise in software update systems, in: E. Al-Shaer, A.D. Keromytis, V. Shmatikov (Eds.), Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010, ACM, 2010, pp. 61–72, http://dx.doi.org/10.1145/1866307.1866315.

[25] T. Karthik, Kuppusamy, D. McCoy, Uptane : Securing software updates for automobiles, in: 14th Embedded Security in Cars (Escar16) Conference in Munich, Germany, 2016.

[26] D. Mbakoyiannis, O. Tomoutzoglou, G. Kornaros, Secure over-the-air firmware updating for automotive electronic control units, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, in: SAC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 174–181, http://dx.doi.org/10.1145/3297280.3297299.

[27] H. Chandra, E. Anggadjaja, P.S. Wijaya, E. Gunawan, Internet of things: Over-the-air (OTA) firmware update in lightweight mesh network protocol for smart urban development, in: 2016 22nd Asia-Pacific Conference on Communications, APCC, 2016, pp. 115–118, http://dx.doi.org/10.1109/APCC.2016.7581459.

[28] M. Goldberg, A secure update mechanism for internet of things devices, in: Cyber-Assurance for the Internet of Things, John Wiley and Sons, Ltd, 2016, pp. 129–136, http://dx.doi.org/10.1002/9781119193784.ch3, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119193784.ch3, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119193784.ch3.

[29] C. Teng, J. Gong, Y. Wang, C. Chuang, M. Chen, Firmware over the air for home cybersecurity in the internet of things, in: 2017 19th Asia-Pacific Network Operations and Management Symposium, APNOMS, 2017, pp. 123–128, http://dx.doi.org/10.1109/APNOMS.2017.8094190.

[30] M.A. Prada-Delgado, A. Vázquez-Reyes, I. Baturone, Trustworthy firmware update for internet-of-thing devices using physical unclonable functions, in: 2017 Global Internet of Things Summit, GIoTS, 2017, pp. 1–5, http://dx.doi.org/10.1109/GIOTS.2017.8016282.

[31] S.K. Kumar, S. Sahoo, K. Kiran, A.K. Swain, K.K. Mahapatra, A novel holistic security framework for in-field firmware updates, in: 2018 IEEE International Symposium on Smart Electronic Systems, ISES Formerly INiS, 2018, pp. 261–264, http://dx.doi.org/10.1109/iSES.2018.00063.

[32] OpenCores, The reference community for Free and Open Source gateware IP cores, URL https://opencores.org/.

[33] N. Asokan, T. Nyman, N. Rattanavipanon, A. Sadeghi, G. Tsudik, ASSURED: Architecture for secure software update of realistic embedded devices, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11) (2018) 2290–2300, http://dx.doi.org/10.1109/TCAD.2018.2858422.

[34] A. Kolehmainen, Secure firmware updates for IoT: A survey, in: 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), 2018, pp. 112–117, http://dx.doi.org/10.1109/Cybermatics_2018.2018.00051.

[35] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, E. Baccelli, Secure firmware updates for constrained IoT devices using open standards: A reality check, IEEE Access 7 (2019) 71907–71920, http://dx.doi.org/10.1109/ACCESS.2019.2919760.

[36] B. Hong, J. Huang, T. Ban, R. Isawa, S. Cheng, D. Inoue, K. Nakao, Measurement study towards a unified firmware updating scheme for legacy IoT devices, in: 2019 14th Asia Joint Conference on Information Security, AsiaJCIS, 2019, pp. 9–15, http://dx.doi.org/10.1109/AsiaJCIS.2019.00-11.

[37] A. Langiu, C.A. Boano, M. Schuß, K. Römer, Upkit: An open-source, portable, and lightweight update framework for constrained IoT devices, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, 2019, pp. 2101–2112, http://dx.doi.org/10.1109/ICDCS.2019.00207.

[38] J. Bauwens, P. Ruckebusch, S. Giannoulis, I. Moerman, E.D. Poorter, Over-the-air software updates in the internet of things: An overview of key principles, IEEE Commun. Mag. 58 (2) (2020) 35–41.

[39] R. Dhobi, S. Gajjar, D. Parmar, T. Vaghela, Secure firmware update over the air using TrustZone, in: 2019 Innovations in Power and Advanced Computing Technologies, Vol. 1, I-PACT, 2019, pp. 1–4.
[40] K. Kerliu, A. Ross, G. Tao, Z. Yun, Z. Shi, S. Han, S. Zhou, Secure over-the-air firmware updates for sensor networks, in: 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops, MASSW, 2019, pp. 97–100.
[41] J.L. Hernández-Ramos, G. Baldini, S.N. Matheu, A. Skarmeta, Updating IoT devices: challenges and potential approaches, in: 2020 Global Internet of Things Summit, GIoTS, 2020, pp. 1–5.
[42] A. Pillai, M. Sindhu, K.V. Lakshmy, Securing firmware in internet of things using blockchain, in: 2019 5th International Conference on Advanced Computing Communication Systems, ICACCS, 2019, pp. 329–334, http://dx.doi.org/10.1109/ICACCS.2019.8728389.
[43] A. Yohan, N. Lo, An over-the-blockchain firmware update framework for IoT devices, in: 2018 IEEE Conference on Dependable and Secure Computing, DSC, 2018, pp. 1–8, http://dx.doi.org/10.1109/DESEC.2018.8625164.
[44] S. Dhakal, F. Jaafar, P. Zavarsky, Private blockchain network for IoT device firmware integrity verification and update, in: 2019 IEEE 19th International Symposium on High Assurance Systems Engineering, HASE, 2019, pp. 164–170, http://dx.doi.org/10.1109/HASE.2019.00033.
[45] E.N. Witanto, Y.E. Oktian, S. Kumi, S. Lee, Blockchain-based OCF firmware update, in: 2019 International Conference on Information and Communication Technology Convergence, ICTC, 2019, pp. 1248–1253, http://dx.doi.org/10.1109/ICTC46691.2019.8939910.
[46] O.C. Foundation, OCF Security specification v2.1.0, November, 2019.
[47] A. Anastasiou, P. Christodoulou, K. Christodoulou, V. Vassiliou, Z. Zinonos, IoT device firmware update over lora: The blockchain solution, in: 2020 16th International Conference on Distributed Computing in Sensor Systems, DCOSS, 2020, pp. 404–411.
[48] L. Alliance, FUOTA process summary technical recommendation TR002 v1.0.0, 2019, URL https://lora-alliance.org/sites/default/files/2019-04/tr002-fuota_process_summary-v1.0.0.pdf.
[49] H. Tschofenig, S. Farrell, Report from the Internet of Things Software Update (IoTSU) Workshop 2016, RFC 8240, IETF, 2017, URL http://tools.ietf.org/rfc/rfc8240.txt.
[50] B. Moran, H. Tschofenig, H. Birkholz, K. Zandberg, A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest, Internet-Draft draft-ietf-suit-manifest-04, Internet Engineering Task Force, 2020, Work in Progress, URL https://datatracker.ietf.org/doc/html/draft-ietf-suit-manifest-04.
[51] B. Moran, H. Tschofenig, H. Birkholz, An Information Model for Firmware Updates in IoT Devices, Internet-Draft draft-ietf-suit-information-model-07, Internet Engineering Task Force, 2020, Work in Progress, URL https://datatracker.ietf.org/doc/html/draft-ietf-suit-information-model-07.
[52] A.O. Mobile, "Lightweight machine to machine technical specification: Core", 2019.
[53] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, Understanding the mirai botnet, in: 26th USENIX Security Symposium, USENIX Security 17, USENIX Association, Vancouver, BC, 2017, pp. 1093–1110, URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.
[54] Google, Cloud IoT Core, URL https://cloud.google.com/iot-core.
[55] R. Nguyen, Mender and Cloud IoT facilitate robust device update management, URL https://cloud.google.com/blog/products/iot-devices/mender-and-cloud-iot-facilitate-robust-device-update-management.
[56] Mender, Mender a robust and secure way to update all your software, 2019, URL https://mender.io/.
[57] Amazon, AWS IoT Device Management, URL https://aws.amazon.com/iot-device-management/.
[58] N. Semiconductor, Device Firmware Update process, URL https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.0.0%2Flib_bootloader_dfu_process.html.
[59] N. Semiconductor, Over the Air Download (OAD), Bluetooth Low Energy Software Developer's Guide 1.01.00, URL http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_30_00_06/docs/blestack/html/cc1350/oad/oad.html.
[60] Espressif, Espressif IoT Development Framework. Official development framework for ESP32, URL https://github.com/espressif/esp-idf/tree/master/components/app_update.
[61] Espressif, ESP HTTPS OTA, URL https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/esp_https_ota.html.
[62] ARM, An introduction to Arm Mbed OS 5, URL https://os.mbed.com/docs/mbed-os/v5.15/introduction/index.html.
[63] ARM, The full iot stack, URL https://www.pelion.com/.
[64] ARM, Security in firmware update, URL https://www.pelion.com/docs/device-management/current/updating-firmware/security.html.
[65] matter, Matter is the foundation for connected things, 2021, URL https://buildwithmatter.com/.
[66] matter, Connected home over IP, 2021, URL https://github.com/project-chip/connectedhomeip.
[67] M. Xu, M. Huber, Z. Sun, P. England, M. Peinado, S. Lee, A. Marochko, D. Mattoon, R. Spiger, S. Thom, Dominance as a new trusted computing primitive for the internet of things, in: 2019 IEEE Symposium on Security and Privacy, SP, 2019, pp. 1415–1430, http://dx.doi.org/10.1109/SP.2019.00084.
[68] I.F. Trentin, S. Berlemont, D.A.C. Barone, Lightweight M2M protocol: Archetyping an IoT device, and deploying an upgrade architecture, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops, 2018, pp. 403–408, http://dx.doi.org/10.1109/PERCOMW.2018.8480313.
[69] H. Birkholz, D. Thaler, M. Richardson, N. Smith, W. Pan, Remote Attestation Procedures Architecture, Internet-Draft draft-ietf-rats-architecture-02, Internet Engineering Task Force, 2020, Work in Progress, URL https://datatracker.ietf.org/doc/html/draft-ietf-rats-architecture-02.