Implementing secured software updates for IoT devices

Josef Haid

www.infineon.com/optiga-trust-x

Contents

1. Introduction

Devices in the Internet of Things (IoT) are used in dynamic environments. The ability to install new software on devices is essential. Software updates are used to add new features, fix bugs, and resolve known security weaknesses.

Although users know that regular device updates are important, they still find them to be inconvenient, disruptive, and sometimes even annoying. In response to this, vendors of IoT devices have worked on highly automated software update processes to (1) reduce customer interaction and (2) minimize downtime of devices and their connected systems.

From a security perspective, these automated processes represent possible entry points for attackers. If they are not properly protected, devices may be left open to manipulation, typically through the installation of malicious code on a device. The malicious code will then do whatever the attacker wants it to, which can include:

- Taking control of critical infrastructures, industrial automation systems or cars, e.g. Stuxnet in 2010 and attacks against computer systems in cars
- Incorporating the device in a botnet for a large-scale DDOS attacks, e.g. the DynDDOS attack in September 2016
- Gathering confidential and/or private data

Security is also a benefit for customers. Their devices are less likely to be misused as a vector to launch other attacks on their own networks or on other Internet hosts (as part of a botnet).

This paper describes the architecture behind secured software update processes and what needs to be taken into account when implementing an appropriate architecture.

2. The secured software update process – A high-level view

IoT device vendors regularly develop updates to add new functionality and fix known bugs. A software update processes is required to distribute and install the new software. Complex operating systems such as Linux support a package-orientated approach. Less complex IoT devices perform a complete software replacement. This is the method covered in this paper.

A basic overview of a secured update process is shown in the figure below and described in the following paragraphs.

The updates are typically distributed using a broadcast model. This means that a single software update data set, known informally as a "software update", is generated (A) and distributed to the target devices (B).

The software update data set is then downloaded onto the devices (1). The download can be initiated either manually or automatically depending on the architecture of the software update service. Afterwards, the system checks that unauthorized and/or manipulated software images have not been installed on the device, and that the software data set is from an authorized issuer. This is done by checking the authenticity and integrity of the downloaded software update data set. After a successful check, the software image is installed in the final position in the device's memory (3). Finally, the device activates the new software image. This is typically done by a bootloader after resetting the device.

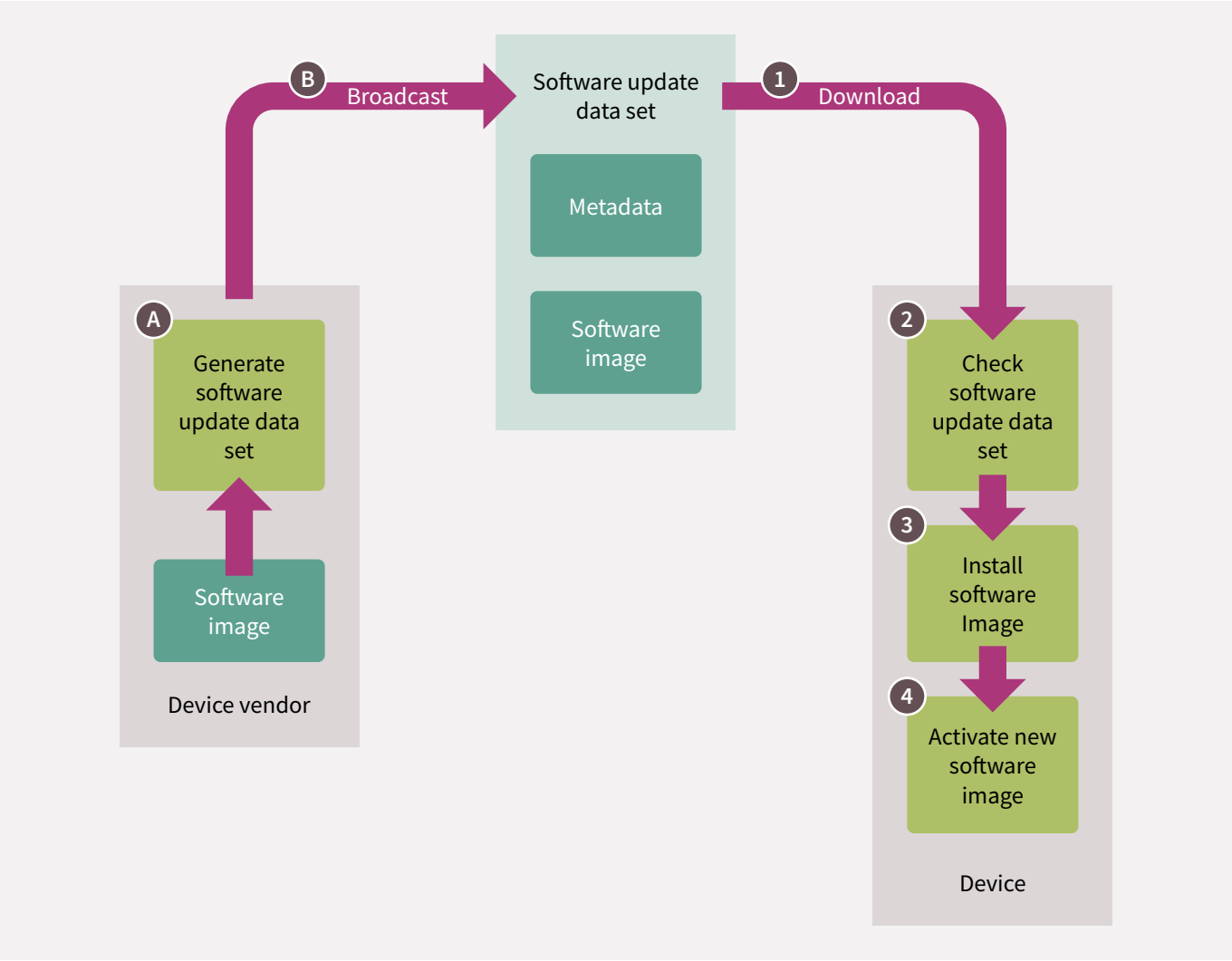


Figure 1: Software update process

3. Security aspects of the update process

With many of today's IoT devices, the user has to provide a password before the software image can be installed. The software itself may then have an additional layer of protection for integrity, for example, a cryptographic checksum (hash). What is missing in many cases is clear proof that the software update image was released by an authorized party, for example the vendor of the IoT device.

The password protection scheme can be replaced or enhanced by cryptographic mechanisms enabling authenticity and integrity checks, policy enforcement processing as well as optional confidentiality protection:

- The entity that issues the software update image can be identified, confirming that image has been issued by an entity that is authorized to update the firmware or software on the device.
- Policy enforcement rules are set by the authorized issuing entity, in other words, the sequence of permitted updates is clearly described. In most cases, for instance, only updates with a higher version number than the current version can be installed, so that software is protected from being rolled back to an older version.
- The intellectual property (IP) of the software image can also be protected to secure the device vendor's business model and prevent the software image from being reverse engineered. Typically, this requires a master key plus a function to derive the local key on the IoT device itself.

The information stored in the software update data set plays an important role and is described in the next chapter.

4. The software update data set

The software update data set, often referred to simply as the “software update”, is a set of data that contains the software code and information required for proper installation.

More specifically, it contains the following two fields of data:

- Metadata describing the software update image and conditions
- The software update image, in other words the actual code that is loaded onto the IoT device

Both fields are described in more detail below.

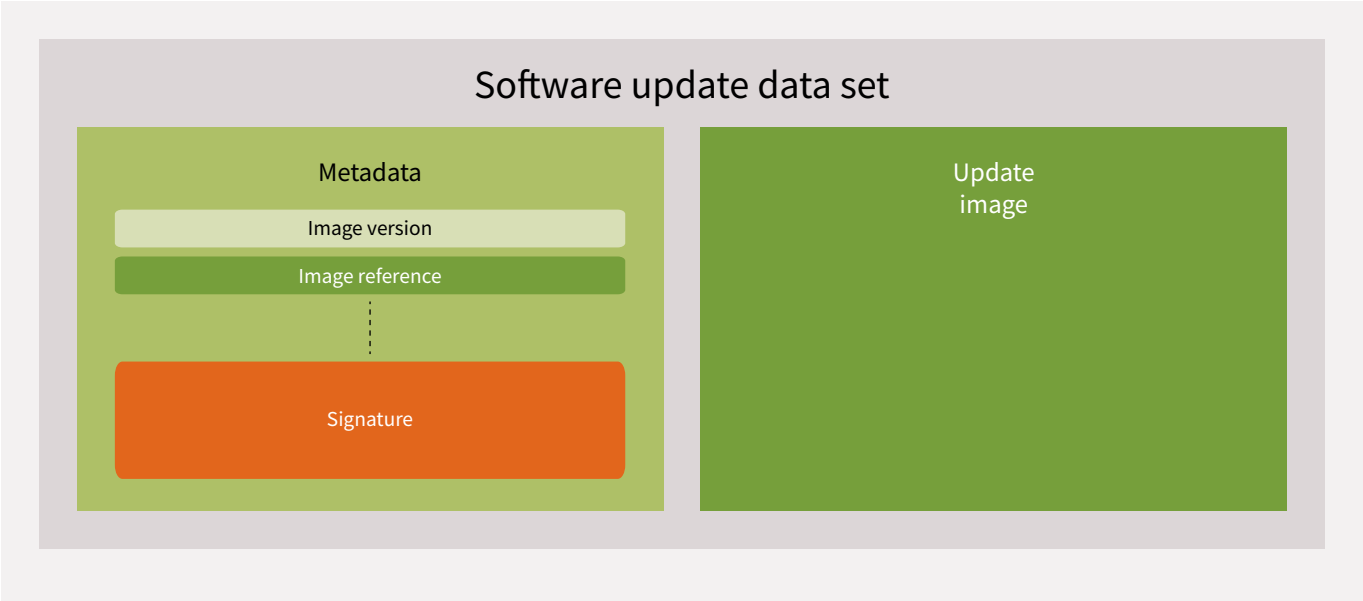


Figure 2: The software update data set contains software code and metadata

4.1 Metadata

An IoT device has to determine who issued the software update data set and the conditions under which the update can be installed. In this context, metadata is crucial as it describes the conditions and policy rules for installation.

The metadata of a software update data set contains a wide range of data, including:

- Image version: The version number is generally used to identify the software installed on the device. From a security perspective, the version number is also required for policy enforcement. This includes preventing a device from rolling back to an older software version (which might have a known security weakness).
- Image hash: Hashing is a cryptographic checksum calculation that is run on the software update image. The IoT device checks the integrity of the software update image by calculating the hash of the received image and comparing it with the hash value in the metadata.
- Image reference data is used to bind the metadata to the software update image. Using a specific (random) number in both the software update image and metadata is one way of addressing this topic.
- The metadata also contains information on how the IoT device should deal with the software update image, for example if it has to replace all or only part of the software installed on the device.

Without further security measures, these data fields typically can be manipulated by an attacker. To equip the metadata with capabilities for securing the update process, the following cryptographic steps are required:

1. The metadata itself must be integrity-protected. This is typically done via a cryptographic checksum called a hash value.
2. The metadata should be electronically signed, for example with an elliptic curve signature that relies on a private key issued by the update root certification authority used by the device vendor.
3. The signature is then verified using the corresponding public key stored securely on the IoT device, ideally using hardware-based security storage features. If this verification fails, the update process will be aborted.

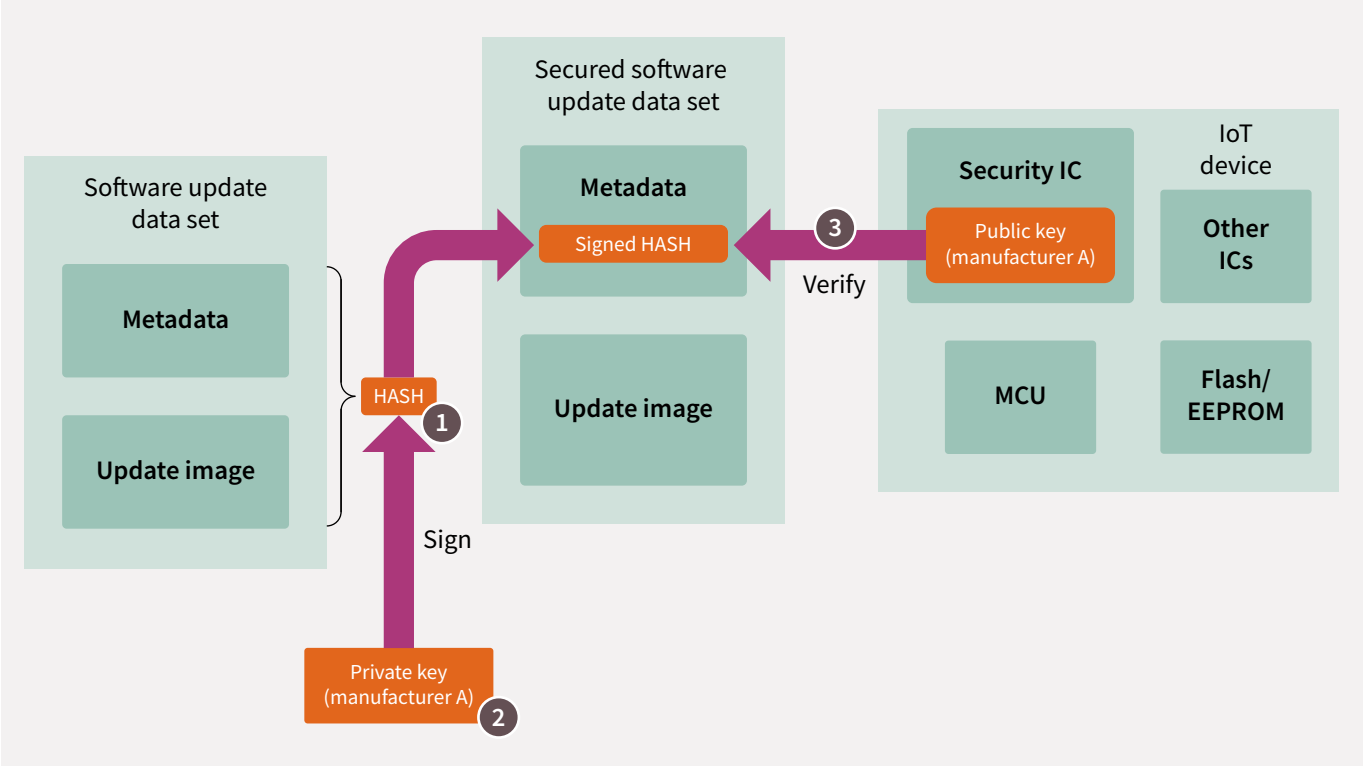


Figure 3: Verifying the authenticity of a secured software update data set

4.2 Software update image

The software update image contains the compiled code that partly or fully replaces the code already installed on the device. To stop the code from being changed by an attacker, its integrity is protected by calculating a hash value and storing it in the metadata (see also the explanation of metadata).

The secured software update image can also be kept confidential. Confidentiality protection is often implemented when software update images are transferred over public networks. In this context, confidentiality prevents reverse engineering and protects against:

- Theft of know-how and IP
- The update being analyzed to identify potential vulnerabilities

To enable confidentiality, the IoT device must know the decryption key. This can be done, for example, by deriving the update key from an update master key assigned to the IoT device during manufacturing and combining this with metadata. The key is derived by cryptographically combining a secret key with further – device-specific – information. This enables a higher security level than can be achieved by using the same secured software image decryption key over the device's entire lifecycle.

5. Implementing secured software update processes

This chapter gives detailed insights into how a secured software update process is implemented. It focuses in particular on how to implement the required functionality in a secured way.

This places the spotlight clearly on hardware-based security. It enables reliable implementation of secured software update processes while at the same time facilitating the design process.

5.1 Hardware-based security

Hardware-based security enhances conventional attempts to secure devices exclusively with software. Unfortunately, software has several inherent and significant weaknesses. Software is written code, and code can be read and analyzed. And once it has been analyzed, it can be modified to the requirements of an attacker. And if a device is re-programmed with modified software, the authentication process and system integrity can be broken.

Another potential and severe weakness of software-based solutions is the inappropriate storage of secret keys during various process and production steps. Typically, in software-based protection systems, attackers can easily identify secret keys that are built into the software or otherwise stored in readable form.

However, software can be protected by hardware. Secured hardware protects the processing and storage of code using encryption, fault and manipulation detection, and secured code and data storage. Software thus becomes trustworthy when combined with secured hardware.

The next section shows how hardware-based security can be used intelligently to implement a comprehensive secured software update process.

5.2 Implementing a secured software update on an IoT device

The main components of an IoT device include the microcontroller (MCU), the non-volatile memory (optionally embedded in the MCU) and the security IC that implements the hardware-based security capabilities. Sensors and actuators complete the IoT design.

The challenge is to map the secured software update process to the device's architecture. Figure 4 below depicts a common way of doing this. It shows how the software update data set is processed across the various components.

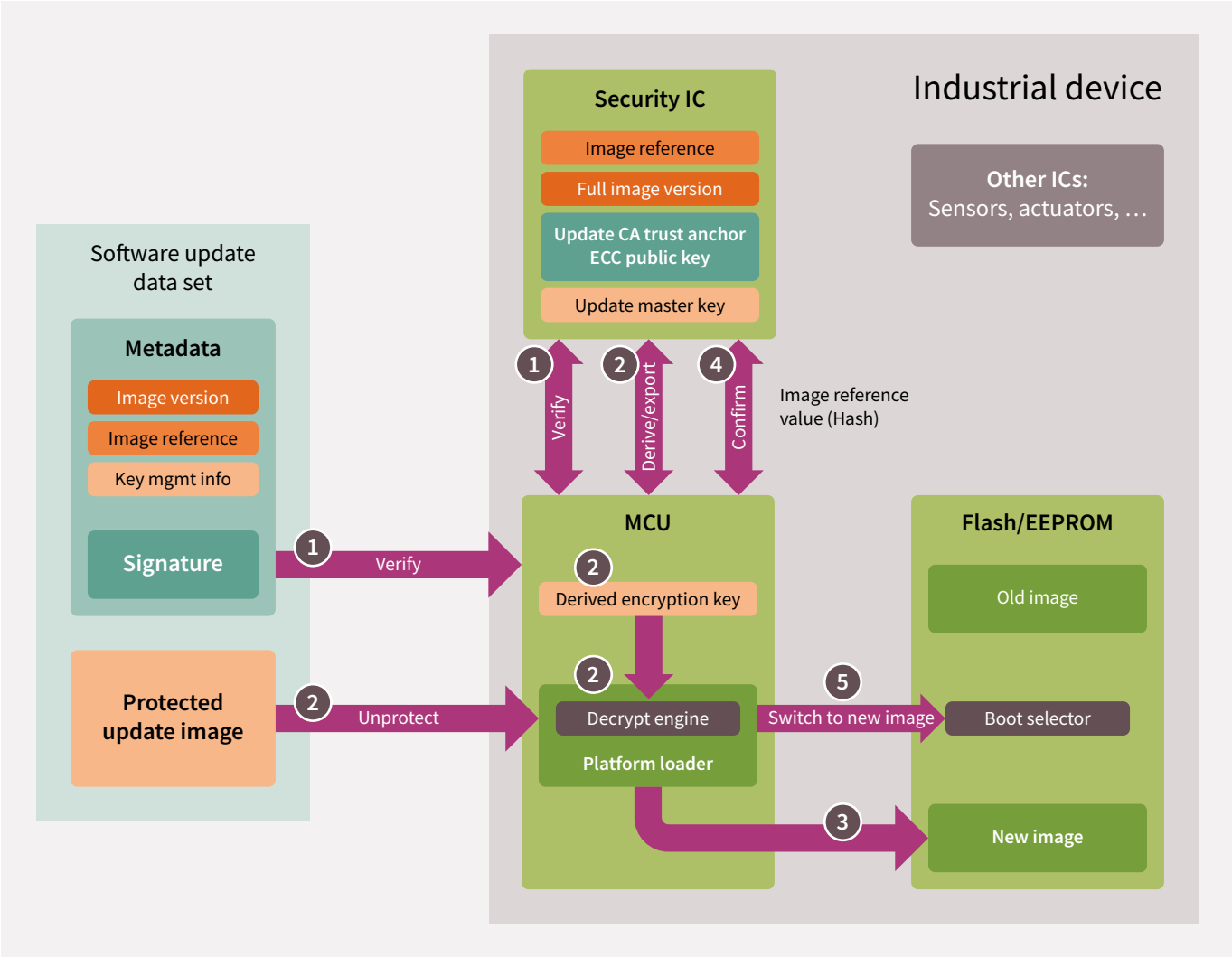


Figure 4: Implementing a secured software update on an IoT device

Step (1) Secured storage of a public key and secured verification of a metadata signature
The metadata is signed with the private key of the issuing entity. This enables the device to check the authenticity of the software update data set. A possible attack scenario would involve the attacker exchanging the public verification key of the IoT device with a view to installing a software update image. A second threat would be improper implementation of the verification algorithm. This would make it easier for an attacker to manipulate this operation.

Hardware-based security is an effective way of countering these attacks. These tamper-resistant solutions protect the integrity of keys and enable them to be stored securely. Hardware-based security solutions also offer functions for on-chip signature verification using public key cryptography on the security IC.

Step (2) Secured storage and handling of the update master key for decrypting the software image
The software update image can also be optionally encrypted to add an additional layer of protection. In this case, the decryption key is derived from a common key installed on all devices of the same type known as the update master key. This key is an interesting target for attackers and must therefore be securely stored and protected against attempts to read it by attackers.

A hardware-based security approach is the most effective way of storing an update master key as it protects the key in a secured and tamper-resistant environment. Additionally, the key can be derived on the security IC. This reduces the risk of attack even further by restricting handling of the update master key to the protected security IC instead of exposing it to the application processor.

The actual decryption of the file is then typically done by a platform loader hosted on the microprocessor (MCU) for performance reasons.

Step (3): Copying the software update image to the target location
At the end of this step, the software update image is copied to the final address in the device's flash memory. If the software update image is encrypted, it has to be decrypted with a derived key before it can be copied.

Step (4) Checking the integrity of a software image and activating a new software image
Before a new software image is activated, a final integrity check is performed. This can be done by running a hash sum calculation on the copied software image and comparing it to the value stored in metadata.

This secured check can be carried out as an encapsulated function on a security IC. This makes it easier to implement and reduces the security architecture required for the device. The result of this comparison triggers the switch to the new software image.

Step (5) Activating the new software image
Once integrity has been verified, the platform loader activates the new software image. It is worth mentioning here that the platform loader itself is an important part of the security concept. As it is immutable, the platform loader's code is typically located in the ROM or one-time-programmable (OTP) non-volatile memory. The platform loader runs on the application processor and takes the final decision to switch from the old to the new software image.

Once this step has been carried out, the secured software update process is finished and the device operates with the new software.

5.3 Provisioning keys for IoT devices

The secured software update process is based on a public key infrastructure. The private key is handled in a trusted environment set up by the IoT device manufacturer. The public key is stored on the device itself, where it is used to authorize secured software update data sets. The integrity of this public key must be protected during the device's entire lifecycle from production through shipment to operation in the field.

This is especially challenging in manufacturing environments as these are often operated by third-parties. Security levels are usually low here and so they are not suitable environments for handling sensitive keys.

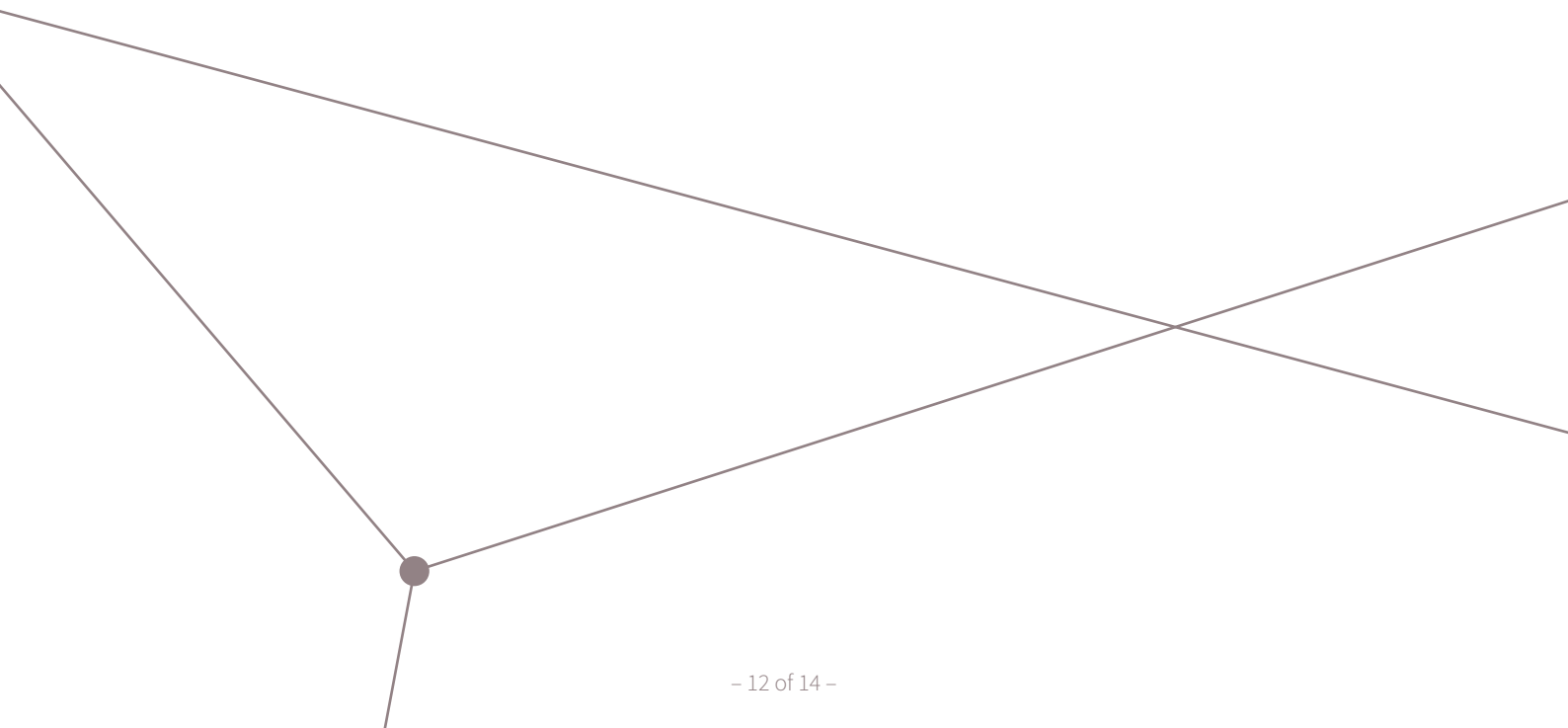
Hardware-based security is an effective means of provisioning IoT device keys and protecting the confidentiality and integrity of those keys. With hardware-based security, keys are programmed into the security IC in a secured environment, for example during chip production or in a secured environment set up by the IoT manufacturer. Once this has been done, the security ICs (with the configured keys) can be shipped to any manufacturer without having to disclose the keys.

5.4 Signing authorized software updates

Software update data sets are distributed and the update can be installed by all devices of the same type. The IoT devices check the authenticity of the software update data sets by verifying the signature of the metadata.
For this to happen, the IoT device manufacturer must operate a service for signing the metadata of the secured software update data set with a private key.
A dedicated step is required if the software update image is distributed in encrypted format. In this case, the manufacturer's service has to encrypt the software image using a key derived from the update master key plus additional image-specific data from the metadata.

In this case, both keys – the private key and the update master key – must be protected. This is typically done through a combination of several measures:

- Physical barriers, for example a separate room with restricted access.
- A dedicated network infrastructure, often decoupled from the internet and company networks.
- Storing the keys securely, in dedicated tamper-resistant security ICs, also known as hardware security modules (HSMs). In this case, the keys do not normally leave the HSM. Instead, they provide the appropriate measures for signing the metadata and deriving the key for encrypting the software update image.



5.5 Using Infineon's OPTIGA™ Trust X to implement secured software update functionality

There is one crucial difference between security and other features. With regular features, the update process "only" has to make sure that the functionality has been correctly implemented. With security features, however, the algorithms must be correctly implemented and secured against attacks. Looking ahead, this is set to become an increasingly challenging task for many device manufacturers as attack models are becoming more sophisticated over time. Hardware-based security offers strong, tamper-proof protection for today's and tomorrow's IoT devices. Software, code and data are stored and processed in secured areas. Encrypted memory and processing, fault and manipulation detection as well as secured code and data storage combine to enable this high level of protection.

OPTIGA™ Trust X was designed to meet the security challenges facing manufacturers of IoT devices. It is a turnkey, hardware-based solution offering enhanced security for connected devices. This includes secured firmware update functionality.

Building on more than 30 years of experience in security, OPTIGA™ Trust X was developed to reduce integration effort – making it ideal for customers who lack security expertise but nonetheless need fast time-to-market. It is a premium security solution offering high performance and low power consumption. This device supports non-rich operating systems and comes in compact packages. In addition, OPTIGA™ Trust X helps IoT manufacturers master design challenges such as restricted space for hardware extensions, cost sensitivity and, in many cases, the power efficiency demands of battery-powered devices.

For more information, please refer to
www.infineon.com/OPTIGA-Trust-X

6. Conclusion

The risk of IoT devices being attacked, modified or misused is high. As such, secured software updates are one of the most critical features of connected devices. This paper provides a comprehensive summary of theoretical approaches and recipes for practical implementation. It also shows that hardware-based security can make it much easier to design secured software update processes.

Infineon's OPTIGA™ Trust and TPM family offers tamper-resistant hardware solutions including a range of ready-to-use security functions tailored specifically to securing IoT devices.

Find out more:
www.infineon.com/OPTIGA