Secure Software Updates for IoT Devices

By Alan Grau, Icon Labs

alan.grau@iconlabs.com www.iconlabs.com

Secure software updates are a critical capability for the success and growth of the IoT. The model of installing embedded devices into the field that are static and never updated is not sustainable. IoT devices must support robust, secure, and scalable software and firmware update mechanisms.

Challenges for IoT Devices

Existing software update mechanisms cannot simply be re-used to solve the challenges of software updates for the IoT. Each of the existing solutions targets a specific class of device (Operating System and hardware) with a defined use (general purpose computer, cell phone, etc.). The IoT consists of a wide range of device types and an ever wider range of uses cases. A software update solution for desktop PCs won't meet the requirements of updating a wearable-IoT device, and the solution used for updating cell phone software won't meet the requirements of the connect car. *A secure software update standard for IoT devices will need to address the range of hardware, operating systems, connectivity capabilities, and operational use cases of a very broad set of devices.*

Secure software updates

A scalable, secure software update standard must define mechanisms (technology) and policies to address the following issues:

- Software signing and validation (including the role of certificates, keys or other authentication mechanisms)
- Transport protocols used for distribution of the software
- Encryption of the software while being distributed, if encryption is used
- Scheduling and distribution of the software updates. For example, a staged rollout of software updates may be desirable so that any issues discovered in the field that escaped lab testing can be found and reported before the majority of the devices are updated.
- Who has control over software updates? For many devices, the user should be allowed some discretion over when updates occur so as not to interrupt their use of the device, or the device needs to be aware of how the update will impact its operation (we don't want cars updating their software while driving down the highway). This needs to be balanced with the need to ensure that updates occur.
- Distribution via an IoT gateway. For IoT devices that connect via a gateway, the role of the gateway in caching and distributing software updates needs to be defined.

- Authentication of the software update server by the IoT device
- Immutable Device Identity
- Secure storage/anti-tamper storage

Some issues have both a policy and a technology component. For example, a policy could specify if the software update package is encrypted for distribution. The encryption algorithm and key distribution mechanisms are technology components that would also have to be defined.

Scalable solutions for IoT devices

Given the limited resources of many IoT devices, a secure software update standard will have to provide sufficient flexibility to scale from small MCU based systems, to larger devices with significant resources. One option is to define tiered security profiles with increasing security requirements. These profiles would define things such as the encryption, signing and hash algorithms used for that profile, how certificates are managed and used for each profile, etc. This allows engineers to understand the trade-offs they are making in terms security. They will understand, and be able to communicate to downstream stakeholders the security trade-offs they have made, based upon the profile they have implemented.

Use of tiered security protocols allows the creation of a standard that addresses very low end devices with minimal resources along with more capable devices with greater computing power. This model allows practical trade-offs between ideal security and what can realistically be implemented in an IoT device, including:

- Use X.509 certificates and RSA signatures
- Support decryption of the software update package
- Support updates of the full software image
- Support decompression of compressed software images

The table below provides a sample of some of the considerations that will need to be addressed when creating security profiles for secure firmware updates. This is intended only to illustrate a framework that can be used as a starting point. Additional effort will be required to define the details, clarify the requirements, and ensure that the appropriate profiles are specified.

Tier	Device profile	Secure update capabilities	Security trade-offs
Tier 1	MCU class device, limited resources running a very small RTOS or no RTOS	Support transfer of encrypted images with a fixed encryption key and hashing of the image for integrity	Requires a shared secret key and only provides weak validation of the image. If the key is leaked there is no way to securely reprogram devices with a new key.
Tier 2	MPU class device running a full featured RTOS but without HW security	Add support for RSA signed images using X.509 certificates.	Much stronger as a strong validation of the image can be performed.

	acceleration		
Tier 3	MPU class device with a security co-processor or HW crypto acceleration.	Add support for certificates stored in a HW TPM or secure HOW storage.	Adds a layer of HW security to the overall solution.

Software update scheduling algorithm

Detailed consideration of implementation issues will be critical for ensuring that any standard developed is widely adopted. The algorithm for distributing software updates is one such consideration.

A staged scheduling algorithm for distributing software updates mitigates against widespread distribution of an update with a software bug that is discovered in the field that was not discovered during lab testing. For systems with very large numbers of devices, or in which the update process is expensive in terms of processing resources, bandwidth or impact upon device usage. A sample staged rollout schedule is provided in the table below.

Day	Percentage of devices updated	Number of devices updated (assumes 1million total devices)
Day 1	.1%	1,000
Day 2	1%	10,000
Day 3	10%	100,000
Day 4	20%	200,000
Day 5	20%	200,000
Day 6	20%	200,000
Day 7	20%	200,000
Day 8	Remaining devices	89,000

Other considerations when creating a rollout schedule include impact on network bandwidth, if the update includes critical security features, and any device specific requirements for maintaining consistency of versions.

The standard should not attempt to define the staged rollout schedule or algorithm, but should include provisions for allowing end users to define a schedule that takes into consideration their needs.

Summary

Secure software updates are a critical feature for IoT device adoption. Ad-hoc solutions won’t scale or solve the interoperability requirements of the IoT. The historic approach of never upgrading devices in the field is not sustainable. Creating a standard that is sufficiently scalable and flexible to meet the needs of IoT devices will require careful consideration of a broad set of use cases, device types, and implementation issues. While not an easy task, creating a standard for secure software updates for IoT devices will help accelerate the adoption of the IoT and solve a critical secure requirement for the IoT.