# IoT Device Security the Hard(ware) way

(Paper published at the 23rd European Conference on Pattern Languages of Programs in July 2018)

Markus Schuß, Johannes Iber, Jürgen Dobaj, Christian Kreiner,
Carlo Alberto Boano and Kay Römer

Institute of Technical Informatics, Graz University of Technology, Austria

## ABSTRACT

Numerous attacks on Internet of Things (IoT) devices have shown that security cannot be neglected, even when building devices with just a few kB of memory. While it is common sense to run regular software updates and use state-of-the-art security on embedded or general purpose systems, this is often not possible with IoT devices. While many of those devices have the facilities to perform over-the-air updates, their memory and processing capabilities limit the use of state-of-the-art cryptography. Additionally, these devices often lack the capabilities to secure the cryptographic keys, the foundation on which the device's security is built, which makes them even more vulnerable to attacks. In this work, we present a pattern that allows even constrained devices to utilize state-of-the-art cryptographic functions, providing the foundation for a secure Internet of Things. The identified pattern presents the following characteristics: (i) *confidentiality*, by offloading the cryptographic functions and key storage; (ii) *authenticity*, by signing messages with the securely stored key using hash as well as signature functions, often too complex for such constrained devices on their own; (iii) *integrity*, a key requirement for connected sensors. As an added benefit, a faster detection of corrupted or tampered updates can also increase the availability of the system. This pattern is primarily targeted at IoT device vendors, who wish to keep their devices secure, by implementing security in hardware.

## CCS CONCEPTS

• **Software and its engineering → Patterns**; *Designing software.*

## KEYWORDS

Internet of Things; System design.

## 1 INTRODUCTION

To date, thousands if not millions of deployed IoT devices lack the capability to communicate securely through the Internet. These devices are often based on constrained 8 or 16 bit micro-controllers with only a few kB of RAM and flash. Despite these severe constraints, these devices rely on the Internet protocol (IP) to connect to servers in the cloud, which alone is consuming a large portion of the available resources. Examples of applications using such devices are parking lot vacancy monitoring, remote controlled lighting, distributed air quality sensors and other smart city or home devices. A block diagram of a device used in such a system is shown in Figure 1, which includes not only the micro-controller and a radio for communication, but also the application-specific sensors and actuators required by the application. Historically, such devices were deployed using proprietary protocols on the user datagram protocol (UDP). With the advent of specialized IoT protocols, which typically favor easier development and information processing over a small memory footprint, the overhead has only increased. Furthermore despite new protocols typically support transport layer security (TLS), the implementation of this feature was often neglected due to the resource constraints of the devices. While one could argue that the replacement of these devices with more powerful versions is the easiest solution to this issue, the cost of replacing millions of devices would be high while not giving the customers any visible benefit. The increase in performance will not only increase the price per unit, but also its power consumption, which in turn decreases the time the device will be able to run on battery. In addition, the use of cryptography in software also increases the complexity of the solution and hence the attack surface for any malicious party while also increasing the likelihood of introducing a software bug. Lastly, this would set a precedent for the next time the used cryptographic primitives become outdated.

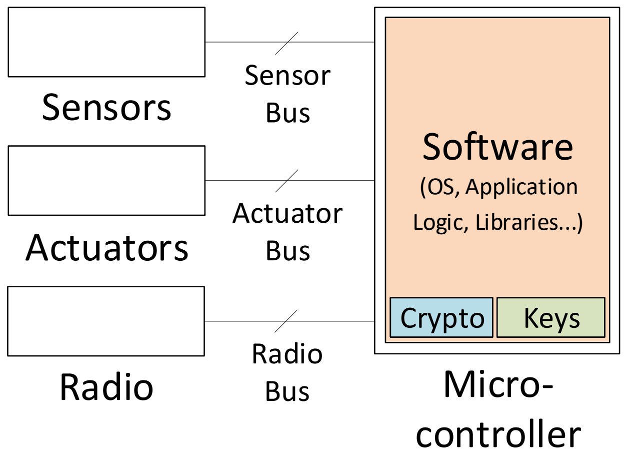

**Figure 1: Typical IoT device using cryptographic functions implemented in software. The devices peripherals (sensors, actuators and radio) do not directly affect the pattern.**

*EuroPLoP '18, July 4–8, 2018, Irsee, Germany*
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6387-7/18/07…$15.00
https://doi.org/10.1145/3282308.3282329

We identified a pattern for (i) adding external cryptographic co-processors to hardware allowing IoT device vendors to ensure that new devices can keep up with the current demands for cryptographic operations, and to (ii) enable their existing devices to cope with the demands of the security challenges of the next decades. This pattern can be applied during the design phase of new devices as well as serve as an upgrade for existing hardware. This pattern can be best observed in applications such as mobile phones and other cellular devices, as they rely on SIM cards to provide authentication to the network and can be used to increase security of applications such as mobile payments. Several new cryptographic features have been added in recent years, allowing the SIM card to serve as secure element for payment and bank transactions. These cards have a backwards-compatible interface, which allows older devices to use even the newest cards and - if the software supports it - all the new features that come with them. Therefore, a SIM card slot should always be considered for IoT devices, even those not relying on cellular technology for communications.

The target audience for this pattern are IoT device vendors that can include the ability to later upgrade the security capabilities of devices with long-term support without adding unnecessarily strong micro-controllers handling more demanding tasks later on. This reduces the cost for device and increases its lifetime.

## 2 HARDWARE IOT SECURITY PATTERN

### Context

The hardware IoT security pattern is placed in the context of the design of a new IoT device, capable of upgrading its cryptography functionality independently from the rest of the system. The latter is important as some micro-controller include cryptographic accelerators, however they cannot be upgraded.

### Problem

The pattern focuses on a way to ensure that the IoT device can always communicate with the vendor's server in a secure way, despite the constrained nature of the device. The main problem the pattern addresses is how to ensure that a IoT device can securely communicate through the Internet by allowing the upgrade the device's cryptographic functions independent from its micro-controller.

Consider an IoT device, designed with a cryptographic function e.g., triple-DES that has been found insecure since its initial deployment. As the hardware was designed for this task and only a limited amount of spare resources were included when designing the system, it is no longer capable to securely communicate. Upgrading the micro-controller to a more capable one would require porting the application and operating system to the new platform. In addition to the development cost, the new device requires testing and certification, which often makes this a highly undesirable undertaking. As these devices are often placed out of reach, e.g., in the ceiling of an office building or embedded in a parking lot, such an upgrade is rarely performed, hence, most of the time, the device remains in operation despite its insufficient security.

### Forces

- *(A) Offload cryptography*: Keep the business logic stable and separate from the security functionality, as the update of one should not affect the other. In addition, this helps with the certification of the device, as an update to the business logic should not affect the security (and vice versa).
- *(B) Costs*: Secure an existing IoT device with a limited amount of resources using state-of-the-art cryptographic operations and without increasing cost significantly or impacting the device's power consumption. As little changes as possible should be made to the hardware to keep the complexity down and provide the ability to perform a "in-field" upgrade.
- *(C) Security "ages"*: Cryptographic functions evolve faster than the typical lifetime of an IoT device. While these devices are intended to be serviced (e.g., battery replacement) every few years, their lifetime is typically intended in the area of a decade or more. Security researchers find flaws in the algorithm at a faster rate. Often, stronger cryptographic functions require more CPU resources and memory, which may not be available on the existing micro-controller.
- *(D) Software always has bugs*: Even if an algorithm is secure, the used implementation may not necessarily be secure as well. Certification of readily available software solutions is often lacking for this reason, and many vendors lack the ability to perform this certification in-house.
- *(E) Malicious attacks*: Side channels have been found in many implementations ranging from power analysis to timing attacks. Some of these attacks can be mitigated in software, which requires more elaborate algorithms and, consequently, drives resource consumption up. Mitigation against physical attacks often requires a specialized processor.
- *(F) Provisioning*: Each device needs to be assigned a private key during production. If this key is part of the firmware, the flashing process becomes more complicated and the costs increase. The manufacturer must be trusted as he has access to the firmware and the embedded private key.
- *(G) Time-to-market*: For security certification and auditing of a product, especially for high security applications, the overhead in time and know-how for developing and testing a completely secure system can be a challenge. The entire certification has to be redone from scratch if the underlying cryptographic functions should change.

### Solution

Use exchangeable cryptographic co-processors to secure IoT devices. Figure 2 shows a device with a slot for such a co-processor in the form of a SIM card. While new devices can be designed with a slot for a such a co-processor, existing devices will need to be upgraded with the addition of a cryptography module. This introduces less costs than replacing the entire hard- and software of all existing devices. In order to facilitate further upgrades, we identified the smart-card standard which has stood the test of time. The use of a SIM-card slot is the best suited for an IoT class device due to its small size and mechanical stability.

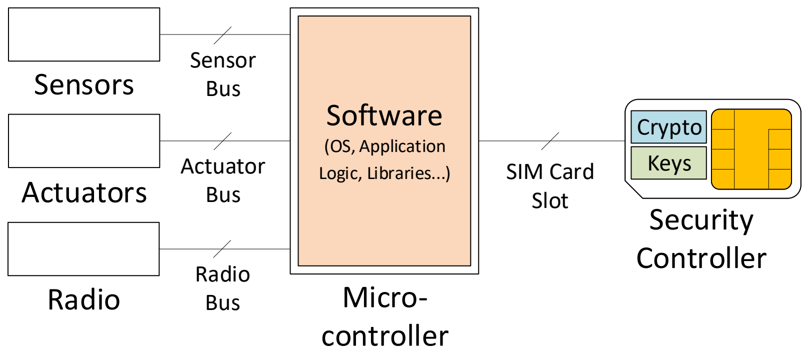

**Figure 2: IoT device implementing the hardware IoT security pattern. Cryptographic functions and key storage are moved to a dedicated, external security controller, freeing up resources on the the micro-controller.**

*Security as an upgradeable hardware component*: Newer devices should always be built with a slot for the security controller in mind. Even when using the best cryptographic functions available today, these functions will also eventually become obsolete. After the initial overhead of adding a slot for a cryptographic module, each upgrade can be performed along the regular maintenance e.g., replacing the battery. While the focus of the pattern is on new devices, it is possible to retrofit existing systems. Depending on the device, the upgrade can be as simple plugging in a connector or, in the worst case, may require soldering by a technician.

Besides using a SIM card containing a security controller, there are similar solutions available from different manufacturers. These devices fall into two main categories, (i) devices which implement the required functionality but use a proprietary software or hardware interface and (ii) devices which adhere to a given standard (this also includes the smart card standard used by SIM cards).

*Proprietary co-processors*: Chip vendors produce co-processors for cryptographic operations, usually relying on a simple hard- and software interface. These range from simple chips capable of only offloading a single type of operation to chips including all functions for an entire communication suite like transport layer security (TLS). These chips are optimized for IoT devices requiring little power and few physical connections to the micro-controller.

*Standardized trusted platform modules (TPM)*: Many vendors have released a versatile crpytographic coprocessors using the TPM standard, which is nowadays typically supported in the Linux operating system. While constrained IoT devices typically run a much simpler operating system, they can still be used, although without official support from the manufacturer. In addition to the offloading the cryptographic function, this chip also supports secure keystorage. Their main benefit is, that they are based on a well-known standard and a vendor lock-in is less likely than with proprietary co-processors. However, unlike the latter, trusted platform modules have a higher cost, more complex interface and typically higher power consumption and is therefore less useful for constrained IoT.

## Consequences

To date deployed IoT devices can be easily upgraded by a non technician by simply replacing the security controller, providing state-of-the-art cryptography, no matter the age of the device. While the cost of components increases, the time spend in development is decreased and the expected lifetime of the device much longer, lowering the total cost over time. The target of the pattern are vendors, however it is recommend to also consider the customer.

*Vendor:* Using a upgradeable co-processor for the cryptograpic functionality ensures that the device can always securely connect to the vendor's servers through the Internet. This is important, as the financial cost as well damage to the brand can be significant, especially considering laws such as the EU's General Data Protection Regulation (GPDR). Offloading the computational heavy cryptographic functions to a dedicated co-processor will also free up CPU time and memory in RAM and flash. With this new-found space, new features can be added to the aging hardware making an upgrade attractive for the end-user of a device.

*Customer:* The lifetime of existing IoT devices can be extended without having to buy and familiarize with a new one. While most customers would like to avoid any maintenance, this is not feasible in the real world (especially when considering battery powered devices). One can only hope that to those who deploy IoT devices in their home, the possible loss of privacy is sufficient reason to perform the maintenance.

Independently of the affected parties, the consequences of the hardware IoT device security pattern can be viewed as a list of pros and cons, which in turn can be attributed to the forces.

+ (A) Updates to the business logic do not affect the security of the device, as this part is handled by the security controller.
+ (B) The security controller requires less power to perform the cryptographic functions than the micro-controller would, extending the battery life.
+ (B) Easy upgrade for the vendor by utilizing existing maintenance intervals prolongs the time a device can be deployed.

+ (C) New cryptographic functions are feasible on old and constrained hardware without a major rewrite of the firmware (it still needs support for the security controller).
+ (E) The security controller is designed to withstand a large number of malicious attacks, including physical attacks which could not be mitigated in software.
+ (F) The security controller can be pre-programmed by the manufacturer using a device with a secure random number generator. Once the keys are stored in the security controller, they can not be read or cloned by a third party.
+ (G) The use of tested and certified security controllers reduces the time-to-market for products.
+ (G) The private key is no longer part of the firmware which speeds up production as all devices now use an identical firmware.
- (A) The software needs to be adapted to make use of the external security controller.
- (B) The additional slot for the security controller as well as the upgrade introduce costs.
- (B) For older devices, a technician is required to modify the existing hardware, which is often not easily accessible.
- (C) Newer algorithms can not be added via a software, but always require the replacement of the security controller.
- (D) There is typically no way to patch the security controller: if there is a bug, it needs to be replaced.

## Known Uses

*Mobile phones.* While the strategy of adding an additional cryptographic co-processor to existing hardware seems far fetched, this has actually happened in the past. SIM cards as typically used in mobile phones nowadays contain secure elements (i.e., a security controller). This feature is often used to ensure the security of payments via the Internet or even using near field communication.

*Set-top Boxes.* While an entirely different field of application, the payed subscription and video on-demand industry has been one of the first industries to realize the pattern of replaceable cryptography in hardware. The business model relies on smart cards with advanced cryptographic functions to ensure that only paying customers can decrypt the video stream since the broadcast region of a satellite cannot be constrained. Whenever a cryptographic cipher has been broken by a malicious third party, they simply have to send out a new generation of smart cards to fix the problem.

*Two-factor authentication.* In order to secure online interactions many companies offer to log-in via two-factor authentication. In addition to a conventional password, a second proof of identity is required. While many use an application for smartphones or even SMS for this, for high security demands, hardware tokens are also used. These tokens use a challenge-response protocol with the cryptographic functions typically implemented in hardware. Similar to the previous applications, if the cryptography would no longer provide sufficient protection, these physical tokens (often USB drives or smart-cards) can easily be replaced.

## Related Pattern

IoT devices require more than just secure communication. Many of the patterns useful for such devices are described by Reinfurt et al. [**?** ]. Given that merely adding a co-processor for offloading the cryptographic functions does not result in a secure device by default, these functions still need to be used correctly, e.g., to establish a secure channel as described by Sinnhofer et al. [**?** ]. The latter provides a pattern that highlights the need to establish a secured connection, while the hardware IoT security pattern provides a means to achieve this in a upgradeable way on even the most constrained devices. In addition, to ensure that the whole system is secure, the entire application needs to be designed with security in mind. A good overview of many different security patterns for software is given by Bunke et al. [**?** ].

## 3 CONCLUSION

In this work we have shown that, while common knowledge dictates that security cannot be added as an afterthought but should always be "designed in", there are still ways to achieve this. We show that it is possible to bring 10-20 year old devices into the cloud using state of the art cryptographic functions and keep them safe for the foreseeable future.

## ACKNOWLEDGMENTS

We thank our shepherd Tobias Rauter, in particular for stopping this patter from turning into a technical manual. In addition, we would like to thank Andrea Höller for the help with a company’s view on this topic as well as the focus group for the great input provided during the writers workshop and beyond.