13 steps to improve security and privacy when developing a smart lock

Riscure | www.riscure.com

Table of Contents

Introduction

Riscure recently analyzed the security of three popular smart locks: those that typically provide extra functionality like remote unlock, smartphone integration, advanced authentication methods, etc. On all devices, we analyzed the internals, reverse engineered the firmware and corresponding smartphone applications and tried several attacks to see if the locks could be compromised. In summary:

- The first lock could be opened using a physical attack on the external unit. We estimate that, with some practice, this lock can be compromised within a few minutes.
- The second lock could be opened using a cryptographic attack on the wireless communication protocol, allowing the door to be remotely opened within one hour and without physical access to the lock/door.
- The third lock was sufficiently protected to resist our attacks. However, we did find that this lock is vulnerable to a Denial-of-service attack, preventing legitimate users to unlatch the lock. Also, we found a privacy issue described later in this document.

These findings show that smart locks can introduce new risks and that there should be more emphasis on security in the development process. Based on our analysis, combined with our extensive experience in evaluating secure embedded devices, we provide 13 recommendations for smart lock developers in this whitepaper.

1 Proper use of cryptography

Many smart locks rely on cryptographic algorithms to authenticate the user and to keep authentication tokens confidential. Although in general, standard cryptographic algorithms are considered very secure, the way they are implemented and combined typically determines the actual security level.

- **Avoid using proprietary cryptographic protocols**. Standard cryptographic algorithms and protocols are extensively reviewed and researched by cryptography experts. Some developers construct new cryptographic algorithms or combine existing ones in a way that they were not designed to be used. Although this can be done correctly, in many cases this makes the implementation vulnerable to cryptographic attacks.
- **Use a proper random number generator.** For secure communication, authentication and on-device key generation, the use of high-entropy random numbers is crucial. Make sure that that random number used for cryptographic purposes are obtained from a high-quality random number generator. Not doing this properly, for example by involving the current time or other predictable values in a pseudo random number generator, might result in vulnerable implementations.
- **Do not store the same secret on all devices.** If lock cryptography involves pre-shared secret keys also found in other devices, the main risk is that it could be easily compromised without requiring the attacker to break a device in the field.
- **Do not use weak key generation/derivation.** If a key is generated based on predictable data or data known to the attacker, it is straightforward to determine the key for an attacker.
- **Store and transport keys securely.** Many developers claim strong cryptography, such as AES-128. In this case, it is important to make sure that such a key cannot be easily obtained by reading firmware, eavesdropping wireless communication channels, or compromised by exploiting vulnerabilities in a backend system in the cloud.

2 Close test and debug functionality

While developing embedded devices, developers often use debugging interfaces, test functionality or “backdoors” to support their development process. When building a production release, it is important to close these mechanisms as much as possible, as they are also extremely useful for someone attacking the device. As an example, we’ve used these interfaces on smart locks to obtain firmware and runtime access to internal memories. This significantly reduces the time needed to reverse engineer and understand the firmware.

In some cases, developers choose to make test functionality logically inaccessible. However, we recommend to entirely remove the implementation of this functionality from the code base, as some logical attacks could re-enable the test functionality. Also, remaining test functionality typically causes debug strings to be still included and referenced in the firmware. This supports the process of reverse engineering the firmware.

3 Secure update mechanism

Like any other software product, firmware running on smart locks can contain bugs and vulnerabilities. Often embedded devices support firmware upgrades, typically using images downloaded from the Internet. It is important that this update mechanism is implemented securely. We recommend:

- Properly authenticating the update image. For example, by verifying a cryptographic signature before programming the new firmware.
- Preventing firmware downgrades, as this opens up the risk of reintroducing vulnerabilities.
- Making sure that the software update mechanism cannot be invoked externally without the owner's consent.

Besides authenticity, also the confidentiality of the firmware should be considered. Easy access to the firmware, for example, when it can be downloaded from the Internet, simplifies the reverse engineering process.

4 Do not trust the outside world

Do not trust commands from devices or interfaces operating in a hostile environment. This includes commands sent through wireless communication channels but also commands sent by external devices such as keypads mounted on the outside of a door. Since the attacker has (or can get, with some effort) control over these external devices, commands can be easily monitored, replayed or modified.

In case of an external keypad that authenticates users, this external keypad should either only send the authentication data to the inside unit, or it should implement a secure channel to the inside unit and implement tamper resistance. In any case, make sure that it is not easy for someone to add eavesdropping equipment to the external devices.

5 Implement mitigation against relay attacks

Suppose a smart lock is automatically opened when its key (e.g., a mobile phone) is in close proximity. In that case, an attacker could bridge the distance between the smart lock and the owner's smartphone using another communication channel and open the lock. This relay attack can be seen as a man-in-the-middle attack in which the communication is not modified.

We recommend implementing mitigations against such attacks; for example by requiring an intentional action from the owner (e.g., pressing a button to open the door) or by enforcing strict timing requirements for the communication.

6 Use secure coding practices

Many software vulnerabilities are caused by insecure coding practices, such as memory copies of which the length of the data is coming from an untrusted source. This could result in buffer overflows, allowing successful exploitation of the smart lock.

We recommend implementing secure coding practices, such as proper input validation, safe default return values and defense in depth (i.e., implement multiple layers of security mechanisms).

7 Protect against unusual activity

On the analyzed smart locks, we have performed several attacks, both successful and unsuccessful, which could have been noticed by the smart lock software if it would monitor unusual behavior. Typically, brute-force attacks (e.g., exhaustively trying many PINs) or cryptographic attacks require a significant amount of requests to the smart lock to achieve their goal.

We recommend monitoring the communication for unusual activity and taking proper actions. For example, limit the number of requests to the smart lock and inform the owner. Please note that the implementation of these countermeasures must not lead to the possibility of Denial-of-Service (DOS) attacks. For example, keeping the door closed after ten failed attempts would also prevent the legitimate user of the smart lock from entering the door.

8 Implement proper revocation support

In some cases, the key to access a smart lock could be compromised or lost, for example when a smartphone with the smart lock key has been stolen. Another example is an attacker trying to sell a used lock, while still holding a valid unlock token. To prevent unauthenticated access, the developer of a smart lock should provide the functionality of revocation of the potentially compromised keys.

We recommend making sure that the revocation mechanism does not only rely on the key device to delete a token but rather actively involves the lock to make sure that previously valid keys are either deleted or blacklisted.

9 Protect against physical attacks

When a smart lock also consists of an external unit operating in a hostile environment, semi-invasive and invasive attacks on this external unit must be considered. Especially if the external unit sends a “secret” message to an inside unit, it could be possible to physically change the external unit to continuously send the “secret” message.

In the absence of other vulnerabilities, attackers might attempt to perform more advanced attacks such as fault injection or side channel analysis.

A fault injection attack aims to disturb the normal operation of a target by introducing environmental anomalies, such as voltage spikes or electromagnetic pulses. The disturbance may cause the target to skip executing some code, or executing completely different code – with the ultimate goal of bypassing the authentication. These attacks are particularly suited for locks with an external unit, where an attacker can usually control the power supply.

Side channel attacks use leakage from the electronic circuitry to recover secret information. For instance, an attacker might be able to measure the power consumption or EM emanations of a chip to recover a key used in a crypto algorithm, exploiting the implementation rather than a weakness in the algorithm itself.

Even internal units might be attacked using side channel analysis or fault injection attacks. There is a recent example that shows that long-distance side-channel attacks on radio signals emitted by mixed-signal chips are possible [1]. Chips like these are typically found in smart locks.

We, therefore, recommend implementing countermeasures against fault injection and side channel analysis for components operating in hostile environments. For more information, see [2].

10 Consider denial of service attacks

Similar to servers on the Internet, smart locks might be attacked using Denial of Service (DoS) attacks, where an attacker tries to make the lock deny legitimate requests to open the door. The goal of an attacker could be to damage the reputation of a certain brand. The attack could be caused by:

- Invalid commands that lead to some kind of memory corruption.
- Deliberately keeping connections open (e.g., Bluetooth or Wi-Fi).
- Aborted (malicious) firmware upgrade attempts.

We recommend to consider these type of attacks and to make sure that any external request is properly validated and handled.

11 Consider attacks on the “key"

For smart locks, the traditional key is replaced by a digital version. This could be a keypad, smart card, RFID tag or smartphone. Especially when the key is implemented on a smart device, such as a smartphone, attacks on this device must be considered. Third-party applications may try to get access to authentication tokens or keys to open the door. Furthermore, an attacker might create a fake lock that requests an authentication token. A smartphone might assume that the lock is a legitimate one, and disclose secret information.

We recommend implementing mutual authentication between the lock and the key. Furthermore, all the recommendations contained in this whitepaper should be applied to both the lock and the key.

12 Secure the backend system

Many smart locks use a backend system running in the cloud to keep clients updated and manage keys and/or tokens. Therefore, the security of backend systems is as important, if not more important, as the lock security itself. Attacks on the backend system can even scale better and allow an attacker to gain control over a large number of locks.

A detailed analysis of different ways of securing a backend system goes beyond the purpose of this document. We recommend considering appropriate security mechanisms. What this means, depends very much on the architecture of the system.

13 It's not just about opening the door

A smart lock should not only protect the door. It is also important to consider the user's privacy. As an example, burglars use mail piling up as an indicator that the residents are on vacation. In one lock we have identified the possibility of performing the digital equivalent of this, where it is possible to see when a lock was used last. This check can be performed quickly, and a burglar could drive around and scan smart locks for a few days, and build a map of all the houses where the lock hasn't been used in the meantime. Since this information is not visible for the owner of the lock, they are unaware and will not take any countermeasures.

What can Riscure do for you?

Riscure helps its customers throughout the complete training, security development, resilient testing, internal red team and lab development and market validation of their customer's Smart Lock Products. If you are looking for a security partner you can trust or wish to promote the security of your solution, please contact us at inforequest@riscure.com.

References

[1] G. Camurati, S. Poeplau, M. Muench, T. Hayes and A. Francillon, "Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers".

[2] M. Witteman, "Secure Application Programming in the presence of Side Channel Attacks".

riscure

Riscure BV
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands

Phone: + 31 (0) 15 251 4090
inforequest@riscure.com

Riscure North America
550 Kearny St. Suite 330
San Francisco, CA 94108
USA

Phone: +1 (650) 646 9979
inforequest@na.riscure.com

Riscure China
Room 2030-31, No. 989, Changle Road
Shanghai 200031
China

Phone: +86 21 5117 5435
inforcn@riscure.com