KUDELSKI SECURITY

Sigfox

Best Current Practices

SECURITY EVALUATION SCHEME ON SIGFOX DEVICES

KUDELSKI SECURITY

10.04.2017

DOCUMENT PROPERTIES

Version:	1.0
File name:	Sigfox - Best Current Practices v1.0.docx
Publication date:	10.04.2017
Confidentiality Status:	Strictly Confidential
Document Owner:	Dominique Le Floch
Document Recipient:	Melia Telemaco
Document Status:	Draft
Client Company Name:	Sigfox

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document is intended for architects and developers of Sigfox devices (device manufacturers, firmware and application providers).

The purpose of this document is to provide to these partners some security guidelines to help the design of new Sigfox devices.

Device Manufacturers should check that a candidate device follows the best current security practices issued by Sigfox before submitting to a security assessment.

INTRODUCTION

Device manufacturers, application providers should implement some of these recommendations so as to reach a minimum security level and avoid the exposure of Sigfox secrets keys.

The robustness of Sigfox devices could be assessed using the Kudelski Security scheme applied to Sigfox devices (Basic or Medium level).

MAIN THREATS ON SIGFOX DEVICES

Sigfox devices could be physically accessible to attackers. Therefore it should be impossible to retrieve Sigfox secrets by simple or in the best case even advanced manipulations on the device.

- Sigfox secret keys
    - It should not be possible to read the Sigfox secret keys using physical or logical memory interfaces.
- Key management
    - It should not be possible to retrieve Sigfox secret keys during key derivation functions or key manipulations.
- Application and firmware integrity
    - It should not be possible to modify the Sigfox library, the device firmware nor the embedded application in order to get the Sigfox secret keys.

- Interfaces with Sigfox library
    - It should not be possible to use the Sigfox library, the device firmware nor the embedded application to authenticate or encrypt/decrypt invalid messages.

1. SECURITY RECOMMENDATIONS

This section describes the mandatory recommendations for any Sigfox devices and additional recommendations for advanced security levels.

The security recommendations should be applied on any chipset (MCU, SOC, Module) that integrate the Sigfox library or store / manipulate the Sigfox Secret keys.

1.1. MANDATORY RECOMMENDATIONS FOR ALL SIGFOX DEVICES

1.1.1. Physical and logical interfaces

- All unused logical or physical interfaces should be locked.
- Physical and logical interfaces should not provide direct or indirect access to the Sigfox secret keys.

1.1.2. Debug interfaces

- Software debug interfaces should not be available on components that have access to Sigfox secrets.
- Hardware debug interface (JTAG) should not be available on devices (should be locked by fuses or access should be protected).

1.1.3. Usage of internal memories (RAM, NVM)

If the Sigfox secret keys are stored in one way or another inside volatile or non-volatile memory, they are particularly exposed in these places.

- NVM internal memory should be used for Sigfox secret keys storage.
- If possible RAM internal memory should be used when manipulating Sigfox secret keys.

1.1.4. Key storage

- Sigfox secret keys should be stored in an area of the NVM memory not accessible by the application.
- Ideally Sigfox secret keys should be stored in encrypted form (logical or physical) in a memory.

1.1.5. NVM interfaces

- Any logical interfaces or APIs providing read/write access to the NVM containing Sigfox secret keys should implement a strict control of address range.

1.1.6. Key management

- Sigfox secret keys should be erased from RAM memory after usage.
- Sigfox secret keys should be manipulated in a RAM area not accessible by the application.
- Sigfox secret keys could be kept in encrypted form in RAM and only decrypted just before usage.

1.1.7. Firewalling between application / firmware

- A logical separation between the application, Sigfox library and crypto-services should be implemented.

1.1.8. Firewalling between data and secret keys

- A logical separation between the data application and secret keys storage should be implemented.

1.1.9. Download mechanism

- Some applications could have a download mechanism using dedicated interfaces, some integrity and authentication mechanisms should be implemented and verified by the device (installation phase and start-up).

1.1.10. Usage of Sigfox library

1.1.10.1 API status code

- The application should check all status codes returned by the Sigfox library APIs.

1.1.10.2 Error management

- When an error or an invalid status is reported by the Sigfox library during the authentication / encryption / decryption of messages, no corrupted data should be returned by the chip.

1.1.10.3 Number of messages processed by the device

- The usage of Sigfox library could be limited at application level (ex: few messages per day), it will reduce exposal and manipulation of the Sigfox Secret keys.

1.1.11. Sigfox crypto-services

1.1.11.1 Access to Sigfox crypto-algorithms and secret keys

- Only trusted applications or the Sigfox library should be able to use the crypto-algorithms for ciphering or deciphering messages and manipulating Sigfox secret keys.

1.1.11.2 Error management

- No corrupted data should be returned by the APIs in case of error detection during the execution of crypto-services with Sigfox secret keys.

1.1.11.3 Wiping intermediate values

- Crypto-services manipulating Sigfox secret keys should erase intermediate values.

1.1.11.4 AES implementation

- The AES drivers should indicate the choice of implementation (hardware or software) and the counter-measures implemented.

1.1.11.5 Advanced counter-measures

- AES drivers could implement counter-measures against Differential Fault Attacks (DFA) to increase the device robustness.

- AES drivers could implement counter-measures against side channel attacks to increase the device robustness.

1.1.11.6 Anti-tearing mechanisms on NVM drivers

- Firmware could implement anti-tearing or transaction mechanisms to keep NVM consistency (ex: sequence counters).

1.1.12. Device life-cycle

1.1.12.1 Personalization

- Once the personalization is done, it should no longer be possible to have a direct access to Sigfox secret keys using the personalization interfaces.

1.1.12.2 Exploitation

- The interfaces and test procedures, used during the exploitation phase should not provide direct access to Sigfox secret keys or an authentication mechanism should be implemented.

1.1.12.3 Maintenance

- The interfaces and procedures, used during a maintenance phase should not provide direct access to Sigfox secret keys or an authentication mechanism should be implemented.

1.1.12.4 De-provisioning

- The interfaces and procedures, used during a de-provisioning process should not provide direct access to Sigfox secret keys or an authentication mechanism should be implemented.

1.2. RECOMMENDATIONS FOR ADVANCED SECURITY LEVELS

1.2.1. Secure element / Microcontrollers

- Including a secure element for the storage and manipulation of the Sigfox secret keys might increase robustness of the device.

1.2.2. Hardware security guidelines

- Device manufacturers should implement hardware security recommendations provided by chip manufacturers to protect the Sigfox secret keys.

1.2.3. Integrity of chip configuration

- It should not be possible to modify the configuration of the microcontrollers, chipsets, SoC or modules that contain the Sigfox secrets (ex: lock on Flash internal memory).

1.2.4. Chip internal bootloader

- In case of software manipulating the keys, microcontrollers, chipsets, SoC containing the Sigfox secret keys should start with an internal bootloader.

1.2.5. Clock management

- In case of software manipulating the keys, microcontrollers, chipsets, SoC should run on internal clock oscillators.

1.2.6. Physical / Electrical environment

- Operational environment should be limited with thresholds (power, frequency, temperature…).

1.2.7. Hardware firewall

- If the chipset manipulating the Sigfox secrets in software offers a MMU or MPU mechanism, these mechanisms should be used to partition the memory into secure and non-secure parts.

1.2.8. Running applications in low privilege mode

- Some SoCs or modules could support different privileged execution modes, thus applications should be executed in a low privilege mode.

1.2.9. Firmware integrity

- An integrity mechanism could be implemented on the firmware containing the Sigfox library and checked during the boot / at runtime.

1.2.10. Firmware authenticity

- An authenticity mechanism could be implemented and checked during the boot / at runtime so as to validate the authenticity of the firmware containing the Sigfox library.

1.2.11. Packaging

- Some packages could limit physical access to the devices, preventing easy access to the interfaces between internal components.

1.2.12. Routing of signals between components

- The routing of signals / bus between components could increase difficulty for attackers to get access to the inter-die connections.

1.2.13. Software implementation

1.2.13.1 Coding guidelines

- Usage of coding conventions and coding guidelines will increase the code's quality, thus reducing software vulnerabilities.

1.2.13.2 Software security guidelines

- Device manufacturers should implement software security recommendations provided by chip manufacturers to protect Sigfox secret keys.

1.2.13.3 Conformity to specifications

- The software implementation should conform to specifications and should not add undocumented use cases.

1.2.13.4 Compilation options

- The firmware, application, and crypto-services should be compiled with a maximum warning level and the results analyzed to reduce the risk of software vulnerabilities.

1.2.13.5 Code quality

- Tools for static or dynamic code analysis could be used to increase software implementation quality and reduce risk of software bugs.

1.2.13.6 Peer code review

- Peer code reviews between software developers could be done to mitigate the risk of software bugs.

- Additional secure code review could be performed to increase the confidence in the software implementation

1.2.13.7 Debug information

- Debug information / traces should be removed on production devices.

2. SECURITY ASSESSMENT ON SIGFOX DEVICES

Kudelski Security labs have defined a security evaluation scheme to assess robustness of Sigfox devices with different levels and efforts.

The “Framework and Lab deliveries” document (Ref 1) describes the process and the deliveries for the security assessment of devices by Kudelski Security labs.

2.1. Basic security level

- Kudelski Security labs should not be able to retrieve Sigfox secret key within 5 men days using simple physical manipulations, using physical and logical interfaces.

2.2. Medium security level

- Kudelski Security labs should not be able to retrieve Sigfox secret key within 15 men days effort and simple physical manipulations, manipulation of physical and logical interfaces, simple logical or physical attacks (software exploit, timing attacks, simple power analysis and electrical glitch).

2.3. Advanced – Custom security evaluations

- Aimed at critical applications with a secure element. Advanced techniques could be used like Fault (Laser/EM) or Side channel attacks. Attack scenarios will be defined on a case by case basis with the Device Manufacturers.

- Software security code audit on critical applications could be performed by the security lab.

2.4. CSPN – Security certification delivered by French authority (ANSSI)

- KS security labs could execute the security tests so as to get this security certification. A pre-assessment phase could be proposed to the Device Manufacturer so as to estimate the security level before starting this certification.

- KS lab could provide support to the Application providers or Device Manufacturers so as to write the Security Target document needed for the CSPN certification.

2.5. Device identification

2.5.1. Hardware identification

- The hardware silicon revisions for the different components should be documented.

2.5.2. Hardware configuration

- A traceability of the different hardware configurations should be available.

2.5.3. Software identification

- The versioning of the different software components should be available (Sigfox library, firmware, application…).

2.5.4. Software configuration

- A traceability of the different software configurations should be available (compilation options, compilation tool chain…).

2.6. Documentations

2.6.1. Hardware architecture

- An overall description of the hardware architecture should be available including information on the hardware security mechanisms activated to protect Sigfox secret keys.

2.6.2. Software architecture

- An overall description of the software architecture should be available including information on the software security mechanisms activated to protect Sigfox secret keys.

2.6.3. Physical and logical interfaces

- To avoid access to secrets through undocumented (forgotten) interfaces, all physical and logical interfaces on Sigfox devices should be identified:
    - GPIO, serial link, JTAG, interfaces with sensors...
    - AT command interfaces, proprietary control layers…

DOCUMENT HISTORY

Version	Status	Date	Authors	Comments
0.1	Draft	05.03.2017	Dominique Le Floch	Draft version
0.3	Draft	06.03.2017	Dominique Le Floch	Update after KS internal review
1.0	Proposal	07.04.2017	Dominique Le Floch	Update with Sigfox remarks

Reviewer	Position	Date	Document Version
Gerrit Holtrup	Senior Security Engineer	06.03.2017	V0.1
Jullian Stephane	Hardware Security Expert	06.03.2017	V0.1

DOCUMENT RECIPIENTS

Name	Position	Contact Information
David Fernandez	Project Manager	David.Fernandez@Sigfox.com

KUDELSKI SECURITY CONTACTS

Name	Position	Contact Information
MELIA Telemaco	Business Development Manager	Telemaco.melia@kudelskisecurity.com

REFERENCES

[1] Sigfox - Security Assessment on Devices - Framework and Lab deliveries v1.0

ACRONYMS

Acronym	Stands for	Definition
DFA	Differential Fault Attacks	
IoT	Internet of Things	
JTAG		
NVM	Non Volatile Memory	
RAM	Random Access Memory	