Securing the IoT: Introducing an Evaluation Platform for Secure Elements

Lea Zimmerli, Tobias Schläpfer, Andreas Rüst

Zurich University of Applied Science (ZHAW)
Institute of Embedded Systems (InES)

September 28, 2020

Abstract

Security for resource constrained IoT devices is an important subject. Rising awareness and up-coming regulations will require manufacturers to increase the level of security on their IoT devices. Semiconductor vendors are addressing this demand with dedicated chips, so-called secure elements. Secure elements provide hardware accelerated support for cryptographic operations and tamper proof memory for the secure storage of cryptographically sensitive material. Specifically, they physically isolate sensitive cryptographic material from the application. However, experience from various projects shows, that the selection and the integration of a secure element into a specific application represents a challenge. Accordingly, the paper discusses the development of a multi-vendor evaluation platform. Particularly, the platform, adopting the widespread Arduino shield form factor, features secure elements from five different vendors. Together with the provided integration into Zephyr OS, the board can be easily fitted to various microcontroller development boards. The presented work intends to support developers in the selection process for secure elements and therefore to contribute to their adoption in IoT devices.

I. Introduction

With the ever faster growing number of IoT devices that are put into operation, many IoT protocols have emerged as well. One of them is the Thread[1] protocol. Among other things, one of the main characteristics of the Thread protocol is the native IPv6 connectivity through to the end device, without the need for a gateway. Although this brings many new features, such as service discovery, end-to-end security and transparent routing from end device to end device, it also exposes the device to attacks from the outside. The end device must therefore be protected sufficiently to prevent attacks from succeeding. One way to do this, is to encrypt all the messages, which are sent from the end device to the server and vice versa. To provide proper encryption, powerful encryption algorithms are needed and keys and certificates need to be handled. But cryptographic operations need a lot of computing power and have relatively long execution times, thus consuming a lot of energy. The typical end device in an IoT network has scarce resources regarding energy supply and computing power. To solve these issues, secure elements have been introduced to the market, offloading the cryptographic operations to dedicated components. Besides, they also offer a tamper proof secure storage for the management of keys and certificates as well as protection against side channel attacks.

When it comes to integrating such a component, debugging has proven to be rather tedious. The secure elements provide an attacker/developer with the least amount of information possible on a failed request. In terms of security, this is a good characteristic yet somewhat ineffective for development. This, in turn, makes the evaluation of several secure elements a very time-consuming process. However, the evaluation should not be neglected. When developing a new device, the evaluation of components is an important part of the development process. Choosing the right components can prevent the need of corrective measures later in the project and save a lot of time and money. But a good evaluation takes time and can therefore be costly as well. For economic reasons, investment into the evaluation process may come off badly, which leads developers to neglect it. Ideally the evaluation process is fast and straightforward but still thorough, hence the need for an evaluation platform. In the following, an evaluation platform and its featured

secure elements are presented, as well as the application framework. Moreover, the advantages when adopting a secure element in terms of energy consumption and execution time are discussed.
This paper is structured accordingly. Section two specifies the characteristics of secure elements and their role in the IoT environment. Section three presents the developed evaluation platform. In doing so, both the hardware with its featured secure elements and the test application are described. With it, the advantages of adopting the presented evaluation platform are stated. Section four then gives an example of how such an evaluation can be performed. This consists of the description of the test setup and the tested operations. Eventually, the results of power measurements are presented given by a generic example.

II. Secure Elements [2]

This section describes the characteristics of secure elements and introduces two distinct categories. Furthermore, it outlines the usage of secure elements and the process of finding an appropriate secure element for a given application.

i. What is a secure element?

Secure elements emerged from smart/banking card applications. Traditionally, semiconductor vendors designed a pure hardware piece of silicon without any firmware. These security chips were then loaded with an application specific firmware, written by external firmware specialists. Since IoT security has become more important, semiconductor vendors now strive to offer a more general solution with their own firmware. Which should be applicable for many different IoT applications.

So, what is a secure element? A secure element is a small integrated circuit (IC) dedicated to execute cryptographic operations mostly in hardware. Such operations include advanced encryption standard (AES), elliptic curve digital signature algorithm (ECDSA) and allow a fast and energythrifty execution of intensive cryptographic operations. Furthermore, the secure elements provide tamper proof memory for secure storage as well as different physical protection measures, such as actively shielded circuits, voltage and temperature tampering detection. Additionally, secure elements provide protection measures against side channel attacks e.g. screaming side channels [3], power or clock observation-based attacks. One very important feature of all secure elements is, that sensitive data never leaves the tamper proof memory. This is especially important considering that all secure elements, supporting and standalone, use an I2C interface to communicate with the MCU, which is not encrypted by default. The I2C interface poses therefore an additional security risk. The interface has either to be protected by encrypting the communication or by actively shielding the I2C lines. If adopting the first method, one has to consider, that the key for encryption has to be stored in the readable memory of the MCU, which is an additional risk. The second method can be realized by putting the I2C lines between two reference planes (VCC, GND) of the printed circuit board. This makes probing more difficult.

ii. Categories of secure elements

Yet, not all secure elements have the same range of features nor do they target the same application use cases. Essentially, they can be divided in two categories. First there is the category of the supporting secure elements. These secure elements are intended to support the execution of cryptographic software running on a host MCU. Supporting secure elements provide secure storage for sensitive material and support the host MCU with hardware acceleration for a variety of cryptographic operations. This may significantly reduce the energy consumption. If a cryptographic software is needed on the MCU, e.g. because special cryptographic algorithms like the JPAKE cipher suite are used, supporting secure elements are an appropriate choice. The second category are the standalone secure elements. These secure elements are capable to autonomously handle a security layer. To achieve this, they provide a complete set of cryptographic operations with additional control features to establish and maintain a secure and authentic connection without software support on the MCU. In this case, the chosen standalone secure elements are able to handle messages for a (D)TLS [4] session, but of course, other options are possible. Yet, standalone secure elements are also able to support an MCU which uses a cryptographic software, like the supporting secure elements. However, if all the cryptographic operations can be executed on the standalone secure element itself, the overall security of an IoT device can be enhanced, because all the sensitive material for the cryptographic operations always remains within the tamper proof memory of the secure element.

iii. Usage of Secure elements

A secure element can be used in many different ways. Applications include authentication, secure storage of sensitive material, cryptographic operation support, protection of intellectual property and many more. A secure element increases the overall security of an IoT device by providing tamper proof memory and hardware support for cryptographic operations involving protected private and immutable public keys. All secure elements provide mechanisms to generate key pairs, with the private key never leaving the secure element or the possibility to insert private keys at manufactory site in a secure environment. In general, it is not feasible to extract any private key from a secure element, neither by software nor physically. Furthermore, secure elements provide authentication features to prevent usage or altering of data stored within a secure element by unauthorized parties. This also allows for authentic storage of public-key certificates, which may be used to authenticate the device towards services or to authenticate services towards the end device using a stored root of trust certificate. Figure 1 shows a simple application example how a secure element may be used.

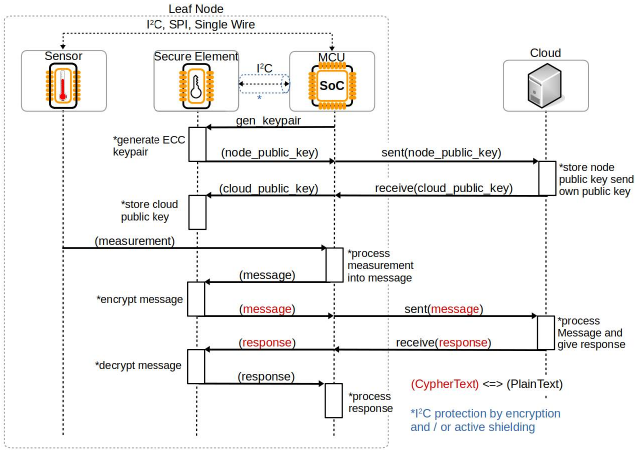


Figure 1: *Simple application example for a secure element*

In the example given, the end device with the secure element on it sends an encrypted sensor value to the cloud and receives the corresponding response. In order to do so, the secure element generates a key pair and sends the public key to the MCU, which forwards it to the cloud. The cloud stores the public key of the secure element and sends its own public key to the MCU, which again forwards it to the secure element. Now the MCU can send the measured value, which it received from the sensor, to the secure element to be encrypted. The encrypted message is sent to the cloud. The cloud processes the message and sends a response to the secure element. The secure element decrypts the message and sends it to the MCU to be processed.

iv. Selection of a secure element

To find an appropriate secure element for an application, a set of key questions has to be answered. These include:

- Which cryptographic operations are needed for the application?
- What's the sensitive material in the application? Where does the sensitive material come from? Generated or preinstalled in a secure environment?
- If ECC is used, which curves need to be supported?
- In case a (D)TLS session is required, which cipher suite is used? How many sessions have to be established and maintained in parallel?
- How often are the cryptographic operations executed, rarely or often?
- Available amount of energy?

Depending on the application, the list of questions continues. By answering these questions, a list of requirements for the secure element is assembled. Furthermore, the answers indicate if a cryptographic software on the MCU is needed or not. In the final step, a fitting secure element has to be selected.

III. Evaluation Platform

This paragraph describes the developed hardware with its featured secure elements and describes the test application design.

i. Secure element shield

To ensure the compatibility and a fast adoption of the secure element shield, the form factor of the widespread Arduino shield has been employed, see Figure 2. Like that, it fits most microcontroller development boards. It features six secure elements from five different vendors; Infineon Technologies, Maxim Integrated, Microchip Technology, NXP Semiconductors and Trusted Objects. Additionally, an interface for the Shield2Go Security OPTIGA^{TM}Trust E board is incorporated. The selection of the featured secure elements has been elaborated in an earlier project. Main

Manufacturer	Secure Element	Standalone	Supporting
Infineon Technologies	OPTIGA™Trust X [7]	x	
Maxim Integrated	MAXQ1061 [5]		x
Microchip Technology	ATECC608A [6]		x
NXP Semiconductors	A71CH [8]		x
	SE050 [9]		x
Trusted Objects	TO136 [10]	x	

Table 1: *List of featured secure elements*

criteria were the range of supported cryptographic features, power consumption and execution times. Table 1 shows a list of the featured secure elements.

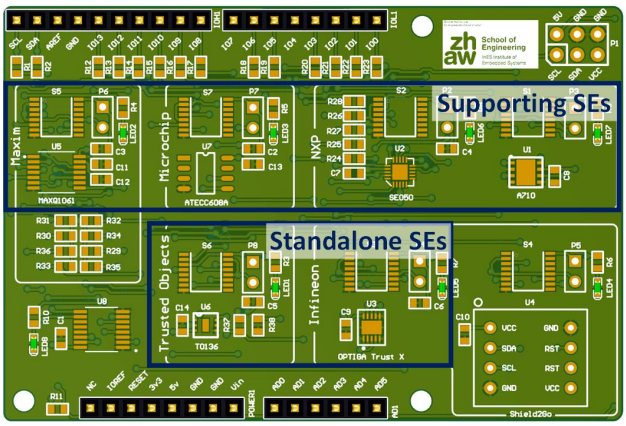


Figure 2: *Secure element shield*

Several features are integrated on the shield that allow to isolate each secure element for individual evaluation. Every secure element can thus be evaluated with the least amount of bias from other components. This makes the shield an ideal platform for evaluation. First of all, each secure element can be completely disconnected from the power supply via a jumper connector if not needed. Therefore, no current will be drawn from disconnected secure elements and hence power measurements will not be distorted. Second, each secure element is connected to the I2C bus via a bus switch. This ensures that the I2C lines are not pulled down due to a secure element, which is not powered but still on the I2C bus. In addition, pin arrays are added where the shield connects to the development board, this will make debugging the I2C line much more convenient. To indicate which secure elements are powered, an LED is assigned to each secure element. The LED may be disconnected for power measurements.

ii. A portable software structure based on Zephyr RTOS

Similarly to the hardware, compatibility between different test environments and a fast adoption to individual setups were the main criteria when developing the test application. As a consequence, all the components of the application are open source, except for some of the secure element libraries. Figure 3 shows the structure of the application. This paragraph gives an overview over the structure and describes the most important components, which consist of the Zephyr RTOS [12], the mbedTLS module and the secure element libraries.

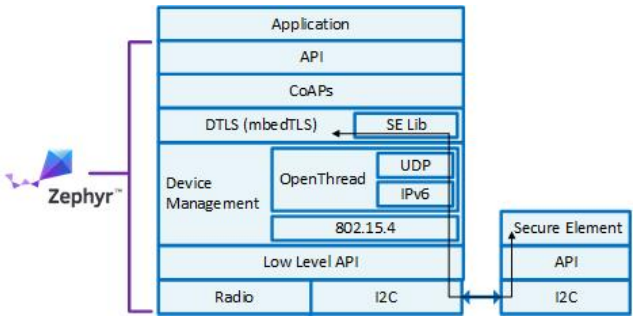


Figure 3: *Application software structure*

The application is based on Zephyr. Zephyr is a small real-time operating system for connected, resource-constrained and embedded devices supporting multiple architectures. Zephyr offers a wide range of protocol stacks, drivers, supported hardware architectures etc. Since Zephyr adopts a modular approach, one can pick and choose only the necessary components. By doing so, the size of the application can be reduced to a minimum and thereby save a lot of memory on the device. One of the core modules used for this application is the mbedTLS library[11], which implements the TLS protocol and its basic cryptographic operations.

mbedTLS provides the basic transport layer security for this application. It has been designed to easily integrate with existing embedded applications. It's not only responsible for secure communication, but it also handles cryptographic operations, key and certificate management. mbedTLS itself can be configured in a way, that only the necessary parts are integrated. Again, minimizing the required memory. But, the most important aspect for this application, is the interface mbedTLS provides for an alternate cryptographic engine, such as a secure element.

The secure elements, much alike the mbedTLS library, provide a range of cryptographic operations, secure storage and some can even handle a security layer on their own, namely the standalone secure elements. All of the featured secure elements are integrated as a supporting secure element. Additionally, the standalone secure elements handle a security layer on their own. Except for the OPTIGA^{TM}Trust X and the ATECC608A, the libraries providing the APIs are not open source.

IV. Evaluation setup

The following outlines the evaluation process. The objective of the evaluation is to measure the average current, execution time and the thereby resulting overall power consumption of each cryptographic engine. Two evaluation setups were done. The first setup focuses on the secure element as hardware acceleration for cryptographic operations (supporting role). The second evaluation setup showcases how a secure element can be used for establishing a secure channel between an IoT device and an application server (supporting and standalone roles). In both cases a reference measurement with mbedTLS as a cryptographic engine is shown. Reason for mbedTLS serving as a reference are NDAs with vendors that prevent the publication of detailed results.

i. Secure element as hardware acceleration for cryptographic operations

Figure 4 shows the first evaluation setup. It consists of the MCU (nRF52840 [13]) and a secure element. The nRF52840 development board is connected to the power supply. On top of the development board is the secure element shield, to which the power analyzer [14] is connected through the jumper connection. In Figure 4, the power analyzer is indicated at the jumper connection of the NXP, A71CH.

For the power measurement, basic cryptographic operations were tested, which are available on all the featured secure elements. Those are:

- Generate random number
- Generate elliptic key pair
- Calculate SHA256 hash
- Calculate ECDSA signature
- Verify ECDSA signature

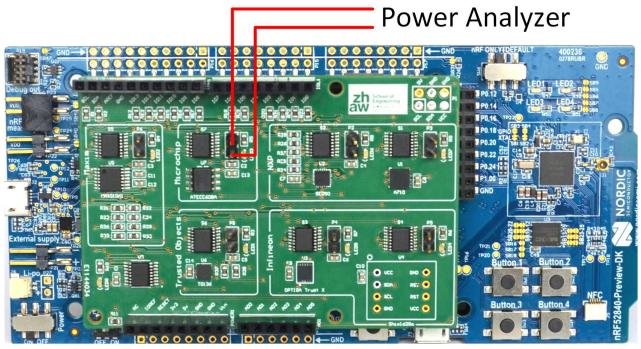
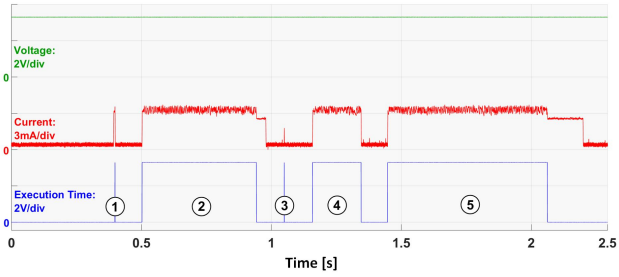


Figure 4: *Power measurement with the nRF52840 development board and the SE shield*

Figure 5 shows the measurement results of the first evaluation setup executing these five cryptographic operations with mbedTLS. On one hand, Figure 5 shows the graph of the measurement and on the other, it lists the individual operations with their execution time, average current and resulting energy consumption of the nRF52840. The operations are marked with the corresponding number in the graph. The application executes the five operations consecutively. Between each operation, the MCU is put into sleep mode for 100 milliseconds. To measure the execution time, a GPIO pin is set to high before the function call and to low directly after the function exits.



Nr.	Operation	Exec. Time [ms]	Avg. Current [mA]	Power [nWh]
1	Random Number	0.3	3.61	0.9
2	ECC Key Pair	439.7	3.27	1318.0
3	SHA256	0.4	3.46	1.1
4	ECDSA Sign	187.9	3.25	559.8
5	ECDSA Verify	607.2	3.27	1820.1

Figure 5: *Cryptographic operations with mbedTLS without support of a secure element*

For the listed operations, we have measurement results available for all secure elements. However, they are not presented here due to existing NDAs. The results show that in some cases, the compute intense operations 2, 4 and 5 are significantly faster when executed on a secure element compared to an execution in software on the MCU. In contrast, the short operations 1 and 3 are faster when executed in software on the MCU. However, as can be seen in Figure 5, operations 1 and 3 have rather short execution times compared to the remaining operations. This is also true for all secure elements. Therefore the impact of those two operations on the overall energy consumption is relatively small. In addition, one has to consider, that the random number generated by a secure element is a true random number whereas the MCU only provides a deterministic random number (pseudo random number). Deterministic random numbers are generated by algorithms and can be reproduced if the starting sequence is known. As a consequence, the random number generated by the secure element provides higher security.

ii. Secure elements authenticating a device in an IoT network

The goal of the second evaluation setup is to secure the communication between the end device and the application server by means of a DTLS session. The test environment is intentionally kept simple, but the same functionality can of course be applied to more complex environments. Figure 6 shows the environment of the second test setup. It includes only a single end device[15], a border router[16], an IPv6 router[17] and a Californium CoAPs server[18].

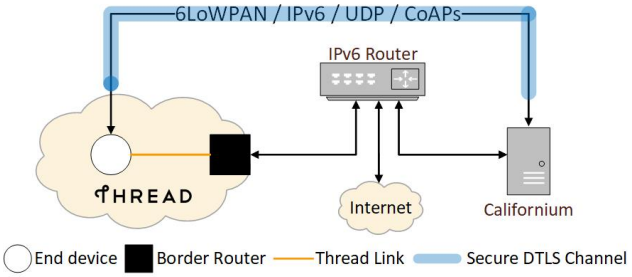
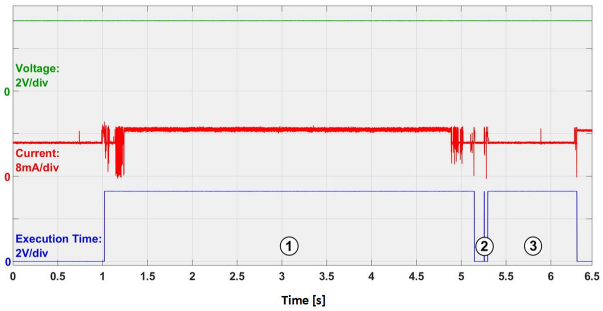


Figure 6: *Network test setup*

For the second evaluation setup the mbedTLS handshake was measured. After the handshake, a CoAP GET message is sent on the encrypted channel from the end device to the application server. The application server then responds with "hello security". The message is then decrypted on the MCU. Among other things, the handshake performs operations such as generating and verifying signatures and encrypting messages. Therefore, it represents a process that is demanding for the MCU.

Figure 7 shows the measured handshake as well as sending and receiving encrypted data. The measurements show the execution time, average current and power consumption of the nRF52840. The handshake has a relatively long execution time and therefore also a high energy consumption. The measurements have shown, that secure elements can cut the execution time of the handshake in half reduce the required energy by a factor five.



Nr.	Operation	Exec. Time [ms]	Avg. Current [mA]	Power [nWh]
1	Handshake	4140.0	8.52	32257.2
2	Send	1.6	8.92	13.1
3	Receive & Decrypt	980.1	6.32	5677.4

Figure 7: *DTLS handshake and communication in software using mbedTLS*

V. Summary and Outlook

Secure elments are an important option to bring security to resource constrained devices. Our measurements with devices from several manufacturers confirm fast execution times and low power consumption. However, so far, the integration of such secure elements needs a lot of effort. Targeting different applications, several manufacturers are offering their secure elements on the market, each of them with its individual firmware, API and certificate management. This makes the seletion of a fitting device challenging and costly.

Our evaluation results for various secure elements show, that there are indeed significant differences in terms of execution time and power consumption. But, since every application has distinct requirements, it is best to test each component separately in a demonstration setup. As a consequence, we have created an evaluation platform for secure elements. The provided hardware and software integration allows the developer of an IoT device to easily add a secure element to an existing setup. As a result, secure elements from different vendors can be directly evaluated and compared for various applications. Clearly, this facilitates the selection of an appropriate secure element.

References

[1] Thread Group, September 28, 2020. URL: https://www.threadgroup.org/what-Is-thread

[2] T. Schläpfer, A. Rüst. Embedded Security for IoT: Opportunities and Challenges of Secure Elements

[3] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon. Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers, EURECOM, 2018. URL: http://s3.eurecom.fr/docs/ccs18n camuratin preprint.pdf

[4] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2, RFC 6347, Jan. 2012. URL: https://rfc-editor.org/rfc/rfc6347.txt

[5] Maxim Integrated, MAXQ1061, DeepCover Cryptographic Controller for Embedded Devices URL: https://www.maximintegrated.com/en/products/microcontrollers/MAXQ1061.html

[6] Microchip Technology Inc. ATECC608A, Secure element to secure authentication URL: https://www.microchip.com/wwwproducts/en/ATECC608A

[7] Infineon Technologies AG OPTIGA^{TM}TRUST X SLS 32AIA URL: https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-trust/optiga-trust-x-sls-32aia/

[8] NXP Semiconductors A71CH, Plug & Trust Secure Element URL: https://www.nxp.com/docs/en/data-sheet/A71CH-SDS.pdf

[9] NXP Semiconductors SE050, Plug & Trust Secure Element URL: https://www.nxp.com/docs/en/data-sheet/SE050-DATASHEET.pdf

[10] TRUSTED OBJECTS TO136 Secure Element URL: https://www.trusted-objects.com/webtest/index.php?page=en-TO136-secure-element

[11] Arm Limited Mbed TLS, version 2.16.0 URL: https://www.mbed.com/en/technologies/security/mbed-tls/

[12] Zephyr Project, September 28, 2020. URL: https://www.zephyrproject.org/what-is-zephyr/

[13] Nordic Semiconductors nRF52840 DK URL: https://www.nordicsemi.com/Software-and-Tools/Development-Kits/nRF52840-DK

[14] Keysight Technologies N6705B DC Power Analyzer, Modular, 600 W, 4 Slots URL: https://www.keysight.com/en/pd-1842303-pn-N6705B/dc-power-analyzer-modular-600-w-4-slots?cc=CH&lc=ger

[15] nRF52840 DK with the secure element shield and test application

[16] Silicon Labs, Border Router, firmware version 2.3.0 GA

[17] Ubiquiti Inc. Ubiquiti VPN-Router EdgeRouter X SFP ER-X-SFP URL: https://www.ui.com/edgemax/edgerouter-x-sfp/

[18] Eclipse Eclipse Californium (Cf) CoAP Framework URL: https://projects.eclipse.org/projects/iot.californium

[19] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.