Using Secure Microcontrollers in IoT Applications:
Insights from a Hands-on Evaluation

Tobias Schläpfer, Andreas Rüst
Zurich University of Applied Science (ZHAW)
Institute of Embedded Systems (InES)
Winterthur, Switzerland
tobias.schlaepfer@zhaw.ch
andreas.ruest@zhaw.ch

***Abstract*—Security in IoT devices is a major topic that IoT is facing. Rising awareness from the customer side and up-coming regulations will force manufacturers to increase the level of security on their IoT devices. Particularly, it is a challenge to leverage the elaborate, well-known computer security algorithms to resource-constrained IoT devices. For the Cortex-A processors Arm® has already introduced their security extension TrustZone® for quite a while. With the new generation of secure microcontrollers, Arm® TrustZone® is now available for battery-powered IoT devices. Furthermore, these secure microcontrollers provide additional security features, such as hardware accelerators for cryptographic operations, secure key storage, and sophisticated random number generators, therefore, increasing security on resource-constrained IoT devices. The paper introduces the concept of these new secure microcontrollers and provides an overview of their features, by showing an application example that covers the topics of secure boot and the usage of TrustZone®. Furthermore, the paper presents energy measurements of the implemented example comparing them to the execution on conventional microcontrollers without TrustZone®. Finally, the paper summarizes advantages and weaknesses of secure microcontrollers compared to dedicated off-chip solutions like secure elements.**

Keywords—IoT security, secure microcontrollers, TrustZone®, ARMv8-M, trusted execution environment, secure firmware, hardware cryptography, resource-constrained devices

I. Introduction

As embedded devices provide increased connectivity and are deployed in the field, they provided multiple attack vectors for attackers. Even worse, most embedded devices lack security measures to prevent attacks. Recently, various incidents such as the IoT botnet [1] or the Las Vegas fish tank hack [2] have shown that embedded IoT devices are a valuable target for attackers. Although the security issue of IoT devices has been recognized by the industry, still only a small percentage of the devices provide adequate security measures. The market seems to have little interest in raising the security level, which has also been noticed by regulators. Hence, there will be new regulations that require embedded IoT devices to provide a higher level of security.

To provide security on their Cortex-A processors, Arm® has designed the so-called TrustZone®. A single hardware processor on which two virtual processors are running, commonly known as the secure world and non-secure world. These two worlds are connected through a security monitor, which protects the stored data in the secure world from leaking into the non-secure world and controls access to the secure world. With the new ARMv8-M architecture released in 2016, TrustZone® has become available for small, energy-constrained devices. Silicon vendors have now introduced new microcontroller units (MCU), which are using the Arm® TrustZone® along with other security features to provide a new generation of secure MCUs (SMCUs).

This paper introduces the concept of the Arm® TrustZone® on the new Cortex-M23 and Cortex-M33 processors, providing hands-on experience from an implemented application example. The example makes use of an open-source bootloader for secure boot. Furthermore, it shows how to partition the memory and peripherals into the secure and non-secure world. Additionally, the example shows how to build a secure application, using the secure world to execute cryptographic operations, store sensitive data and establish a secure communication channel. The application example is implemented using an open-source real-time operating system (RTOS) called Zephyr [3]. Zephyr is specifically designed for embedded devices, aiming to provide a small memory footprint, effortless peripheral configuration, and simple hardware portability. In terms of security, Zephyr provides support for TrustZone® applications and integrates cryptographic software like mbedTLS [4] and MCUBoot [5]. The paper presents energy measurement results of the implemented application executed on an SMCU with and without TrustZone® support. Furthermore, the application has been implemented and measured on conventional MCUs. The results of these measurements are also compared to the results of the SMCU. Finally, the paper compares SMCUs to dedicated off-chip solutions called secure elements.

This paper is structured accordingly. Section 2 discusses SMCUs and the ARMv8-M architecture in general. Furthermore, providing a table of common SMCU features from currently available SMCUs. Section 3 describes the Arm® TrustZone® in detail, specifically showing how the protection measures work. The following section describes the implemented application example, which is using a secure bootloader and TrustZone® to build a secure embedded device. Section 5 discusses the results of the performed energy measurements. Finally, the paper discusses the differences between SMCUs and secure elements, followed by a summary of the advantages and weaknesses of SMCUs. The paper closes with appropriate conclusions.

II. Secure microcontrollers

So what are these new secure microcontrollers? They are equipped with the new generation of Cortex-M processors, namely the M23 and M33. Therefore, these processors provide support for the newly designed TrustZone® for Cortex-M processors. TrustZone® enables developers to implement firmware in a so-called trusted execution environment (TEE). Increasing security on embedded devices by executing security-related tasks in a trusted (secure) environment, to which the untrusted (non-secure) application has limited access. Fig. 1 shows the firmware evolution from conventional to secure firmware on SMCUs.

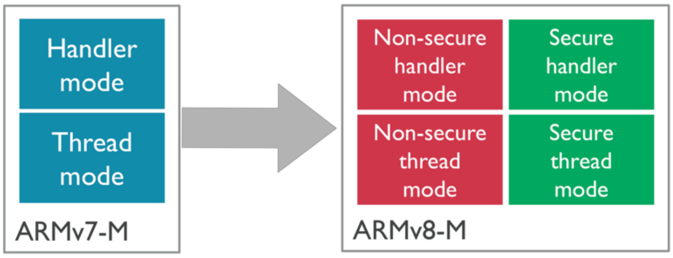


Fig. 1 Change from conventional, to secure firmware [6]

In addition to TrustZone®, SMCUs also provide hardware security features, such as hardware accelerators for fast and energy-efficient execution of cryptographic operations. Furthermore, SMCUs are equipped with a sophisticated random number generator (RNG). These so-called true random number generators (TRNG) are certified by approved institutions such as the German federal office for information security (AIS-31 [7]) or the American National Institute of Standards and Technology (NIST, 800-90 [8]).

A. *ARMv8-M architecture*

The Cortex-M23 and M33 are the latest generation of Arm® processors on the market. Both processors are based on the ARMv8-M architecture. The M23 is comparable to the Cortex-M0+ and the M33 to a Cortex-M4, as is displayed in Fig. 2.

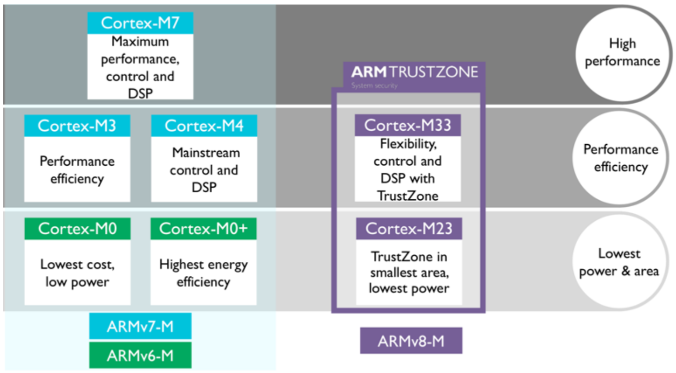


Fig. 2 Overview of Arm® Cortex-M processors [9]

These processors provide standard MCU features such as a memory protection unit (MPU), a nested vectored interrupt controller (NVIC) and an advanced high-performance bus (AHB). The difference with the new ARMv8-M architecture is the support for TrustZone® in the central processing unit (CPU). Fig. 3 shows the block diagrams of the M23 and M33.

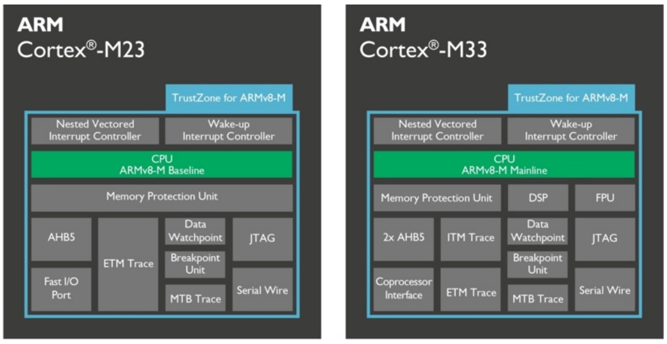


Fig. 3 Block diagrams of Cortex-M23 and Cortex-M33 [10]

The ARMv8-M architecture is divided into two versions, the Baseline version for the smaller, low power M23 and the Mainline version for the M33, providing digital signal processing (DSP) and floating point (FPU) units. While both versions provide stack limit registers for the secure world, only the Mainline version provides stack limit registers for the non-secure world. Furthermore, the Mainline provides a specific SecurityFault exception in case secure data is accessed without permission. On the Baseline, this exception gets handled by a general HardFault exception in the secure world.

To enable TrustZone® support, yet preserving low interrupt latency of previous Arm® architectures, the ARMv8-M architecture has several additions, examples include:

- Four stack pointers: MSP_S (Secure Main Stack Pointer) and PSP_S (Secure Process Stack Pointer) and MSP_NS and PSP_NS
- Two Sys Tick timers, i.e. one for each world
- Two separate sets of configuration registers to configure the memory protection unit (MPU)
- Configurable MPU regions with size granularities of 32 bytes

- The Secure Attribution Unit (SAU) and Implementation Defined Attribution Unit (IDAU), to define the security state of memory regions and peripherals
- The Test Target instruction to allow software to determine access permissions and security attributes of objects in memory

These additions enable the processor to have a Handler and Thread mode in both worlds, whereas the Thread mode may additionally be either privileged or unprivileged, as displayed in Fig. 4. On a conventional ARMv7-M processor there are only the three modes in the red square.

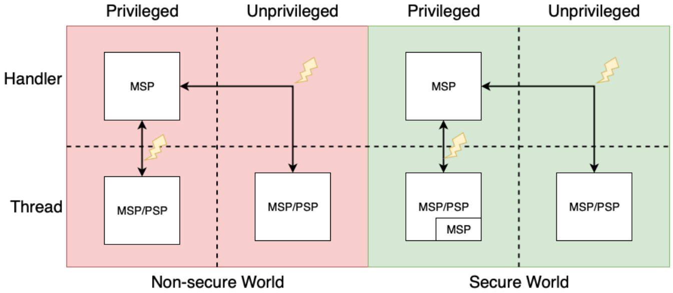


Fig. 4 Processor modes on an ARMv8-M processor

However, Cortex-M processors also need to have low power consumption. Providing two full sets of processor and other hardware registers would increase the size of the die and power consumption. Therefore, several registers on the ARMv8-M architecture are banked for shared use from both worlds, examples include:

- The processor registers *R0* to *R15*
- The Vector Table Offset Register (VTOR)

B. *SMCU features*

SMCUs are available from major MCU vendors such as Microchip [11], Nordic Semiconductor [12], NXP Semiconductors [13] and STMicroelectronics [14]. To provide an overview of currently available SMCUs and their security features, one SMCU from each of the four major vendors has been selected for evaluation. The selected SMCUs are:

- Microchip SAM L11 [15]
- Nordic nRF9160 [16]
- NXP LPC55S69 [17]
- STMicroelectronics STM32L562 [18]

There are currently only three development kits (DK) of the selected SMCU available. Fig. 5 shows the available DKs from Microchip, Nordic and NXP. According to ST, the DK for the STM32L562 SMCU will be available by October 2019.

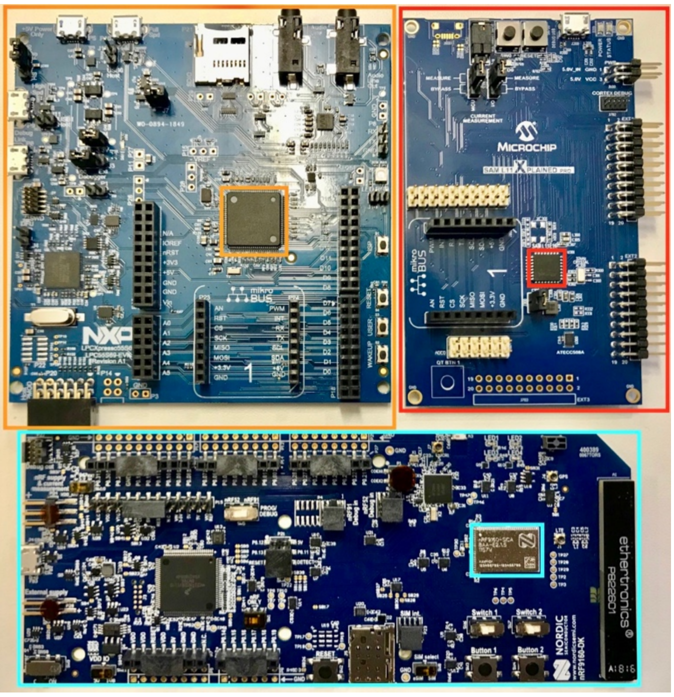
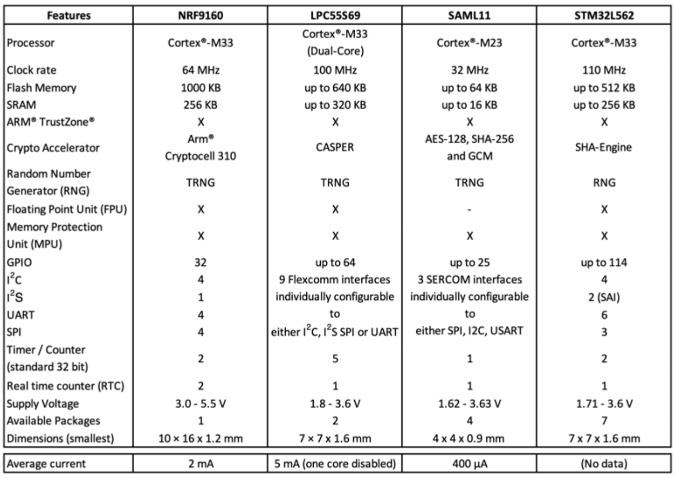


Fig. 5 Available DKs from Microchip, Nordic and NXP

Fig. 6 provides an overview of the features of the evaluated SMCUs. The information has been gathered from the associated datasheets. The listed average currents are measured on the available DKs, during the execution of a small application example, calculating two values within the secure world and returning them to the non-secure world.



Features	NRF9160	LPC55S69	SAML11	STM32L562
Processor	Cortex®-M33	Cortex®-M33 (Dual-Core)	Cortex®-M23	Cortex®-M33
Clock rate	64 MHz	100 MHz	32 MHz	110 MHz
Flash Memory	1000 KB	up to 640 KB	up to 64 KB	up to 512 KB
SRAM	256 KB	up to 320 KB	up to 16 KB	up to 256 KB
ARM® TrustZone®	X	X	X	X
Crypto Accelerator	Arm® Cryptocell 310	CASPER	AES-128, SHA-256 and GCM	SHA-Engine
Random Number Generator (RNG)	TRNG	TRNG	TRNG	RNG
Floating Point Unit (FPU)	X	X	-	X
Memory Protection Unit (MPU)	X	X	X	X
GPIO	32	up to 64	up to 25	up to 114
I^2C	4	9 Flexcomm interfaces individually configurable to either I^2C, I^2S SPI or UART	3 SERCOM interfaces individually configurable to either SPI, I2C, USART	4
I^2S	1			2 (SAI)
UART	4			6
SPI	4			3
Timer / Counter (standard 32 bit)	2	5	1	2
Real time counter (RTC)	2	1	1	1
Supply Voltage	3.0 - 5.5 V	1.8 - 3.6 V	1.62 - 3.63 V	1.71 - 3.6 V
Available Packages	1	2	4	7
Dimensions (smallest)	10 × 16 x 1.2 mm	7 × 7 x 1.6 mm	4 x 4 x 0.9 mm	7 x 7 x 1.6 mm
Average current	2 mA	5 mA (one core disabled)	400 µA	(No data)

Fig. 6 Features of the four evaluated SMCUs

The nRF9160 provides a lot of Flash and SRAM memory which is important for applications using TrustZone®, due to the memory partitioning. The CC310 Arm® Cryptocell supports the execution of cryptographic operations, by making them faster and more energy efficient. Unfortunately, the nRF9160 needs a minimum operating voltage of at least 3.0 V. Also, the nRF9160 is only available in a single package version.

The LPC55S69 is available with one or two CPUs. Having two cores may allow building a secure application where one processor is entirely dedicated to security-related tasks and the other executes the application. A mailbox between the two cores may handle the data exchange. The CASPER crypto co-processor is comparable to the Arm® Cryptocell.

The SAM L11 is the only SMCU of the four evaluated, which has a Cortex-M23 instead of an M33, which is reflected in the slower clock rate and lower memory capacity. Especially the limited memory capacity could be an issue when developing a secure application. For example, the application implemented for this paper, could not be implemented on the SAM L11, as mbedTLS on its own already requires 16 KB of SRAM. However, smaller applications may benefit from the SAM L11, due to its hardware accelerator, the TRNG, and low energy consumption.

The STM32L562 provides a Cortex-M33 with the highest clock rate of all evaluated SMCUs. Furthermore, the STM32L562 provides sufficient Flash and SRAM memory for an application using TrustZone®. However, compared to the other SMCUs, the STM32L562 provides a less sophisticated hardware accelerator, since it only supports SHA operations. The datasheet claims to provide a TRNG but there is no information regarding its certification, therefore the table in Fig. 6 lists it as an RNG only.

III. Arm® TrustZone®

This section focuses on the differences between TrustZone® on a Cortex-A and Cortex-M processor. Furthermore, this section provides a detailed description of how to switch between the non-secure and secure world.

The principle behind TrustZone® is the separation of a single processor into a so-called non-secure (untrusted) and secure (trusted) world, see Fig. 7. These two worlds are separated by a security barrier, which controls interactions and data flow between the two worlds. This separation allows for the development of firmware in a TEE where each world has specific responsibilities and privileges. Whereas the secure world has access to secure and non-secure memory, the non-secure world can only access non-secure memory. The idea is to run the main application in the non-secure world, while the secure world handles the applications security tasks and stores sensitive data.

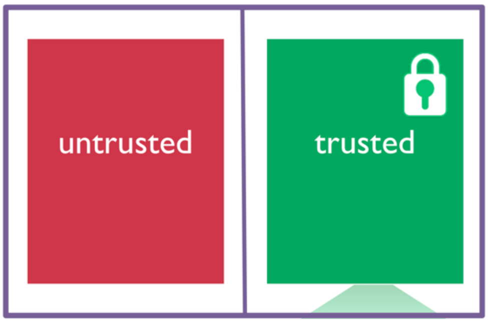


Fig. 7 Firmware on a processor with TrustZone® [9]

A. *TrustZone® differences between Cortex-A and Cortex-M*

For an application running in the non-secure world to use a security function located in the secure world, there have to be precisely defined methods for interaction between both worlds. To switch between the execution in the non-secure world and the secure world and vice versa, a so-called context switch has to be executed. On a Cortex-A processor, a secure monitor is responsible for this context switch, see Fig. 8. The secure monitor serves as a single point of entry between the two worlds. It protects the secure world from leaking data to the non-secure world and manages access to the secure world.

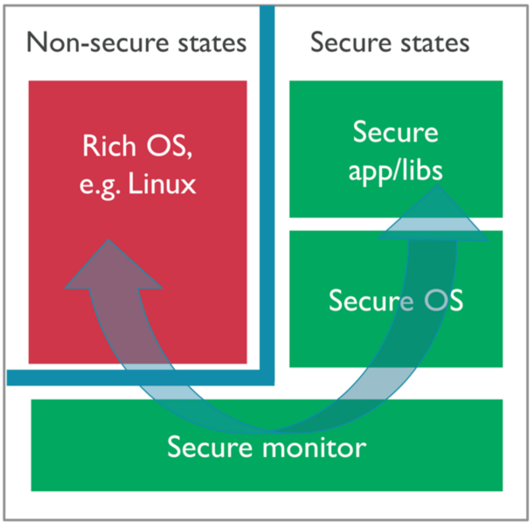


Fig. 8 Secure monitor on a Cortex-A processor [6]

Embedded devices with a Cortex-A processor run on up to 1 GHz and provide a high-performance OS, e.g. a Raspberry Pi or Beaglebone. On the other hand, these embedded devices require mains power and may have long interrupt latencies. In contrast, embedded devices that use a Cortex-M processor run on up to 200 MHz and are mostly battery powered. They feature, low interrupt latency and low power consumption. Due to these requirements, having a single point of entry between the secure and non-secure world as well as having an additional component that consumes power is not reasonable for a Cortex-M processor. Therefore, the main difference between TrustZone® on a Cortex-A and Cortex-M is the lack of a secure monitor on the Cortex-M, see Fig.9. The non-secure world can directly interact with the secure world through newly added specific instructions for fast, energy-efficient yet secure, context switching. As a result, it is possible to serve non-secure interrupts and exceptions, although the processor might be running in the secure world at the time of the triggering event. This allows for preserving the low interrupt latency from previous Cortex-M processors. The presence of shared registers allows a direct exchange of data between the two worlds, which makes the secure monitor dispensable. Therefore, this helps to reduce the overall power consumption. However, the shared registers constitute an inherent security risk that needs to be mitigated.

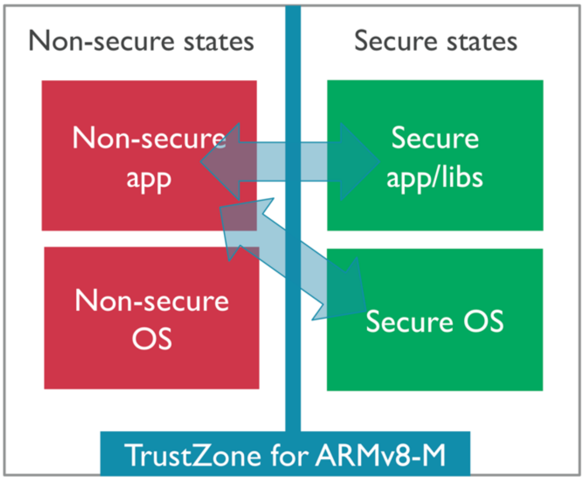


Fig. 9 Direct interaction between the non-secure and secure world on a Cortex-M processor [6]

B. *Context switching on a Cortex-M processor*

Fig. 10 provides an overview of the different interactions between the different processor states. Whenever an arrow crosses the dashed grey line the current processor context has to be switched.

As pointed out, fast context switching is key to embedded applications running on a Cortex-M processor. However, context switching represents an inherent security risk. Due to the presence of shared registers, secure data may be exposed to the non-secure world if context switching is not done properly.

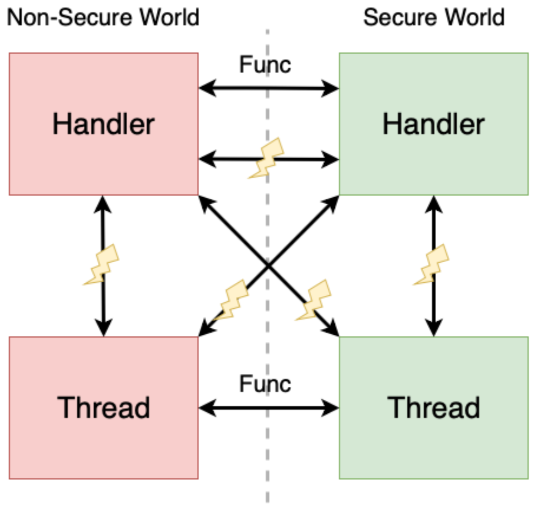


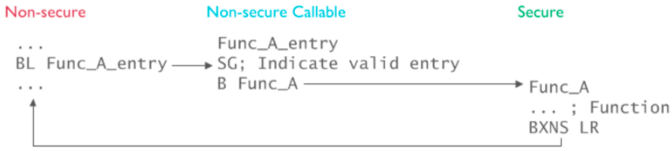
Fig. 10 Interactions between the different processor states

For example, the Arm® Procedure Call Standard for Arm® Architecture [19] specifies that *R0* to *R3* are used by the compiler to pass parameters and return values in case of a function call. Therefore, data stored within these registers could be leaked to the non-secure world when returning from a function located in the secure world. To prevent the exposure of secure data, Arm® has introduced three methods to securely switch context on a Cortex-M processor. The first method is through a so-called Secure Gateway (SG) instruction. This instruction can be used from the non-secure world to switch to the secure world using a direct secure API call. The second method is through a BXNS (branch with exchange to the non-secure world) instruction which is used from the secure world to return to the non-secure world. Lastly, a context switch can be executed through a BLXNS (branch with link and exchange to the non-secure world) instruction, used by the secure world to call functions provided by the non-secure world.

1) *Secure Gateway and BXNS instructions:*

An SG instruction serves as an entry point for the non-secure world to access functions within the secure world. Fig. 11 shows the control flow if the non-secure world calls a function (*Func_A)* located in the secure world. For *Func_A* to be callable from the non-secure world in the first place, the secure world has to define *Func_A* as a so-called non-secure callable (NSC) function. The result of this declaration is *Func_A_entry* which is located in the non-secure callable region. If the non-secure world calls *Func_A_entry*, the SG instruction branches the processor to *Func_A* located in the secure world. Once *Func_A* completes its execution, a BXNS instruction will branch the processor back to the address in the non-secure world. However, before branching to the non-secure world, the processor automatically clears all processor registers that contain data from the secure world.

If the non-secure world attempts to branch, or call an address in the secure world, without using an SG instruction as a valid point of entry, e.g. calling *Func_A* directly, a fault event is generated.

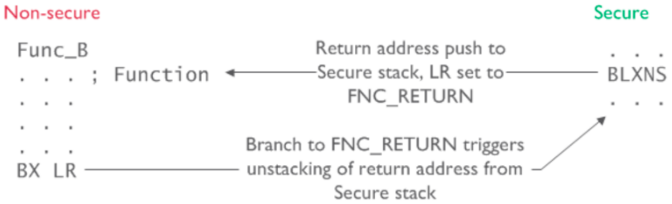


```
...                  Func_A_entry
BL Func_A_entry ---> SG; Indicate valid entry
...                  B Func_A ------------------> Func_A
                                                  ... ; Function
                                                  BXNS LR
```

Fig. 11 Control flow when the non-secure world calls a function located in the secure world [20]

2) *BLXNS instruction:*

Fig. 12 shows a BLXNS instruction used by the secure world to call a function *Func_B* in the non-secure world.


Non-secure
Secure
Return address push to Secure stack, LR set to FNC_RETURN
Branch to FNC_RETURN triggers unstacking of return address from Secure stack

```
Func_B                                . . .
. . . . ; Function                    BLXNS
. . .                                 . . .
. . .
. . .
BX LR
```

Fig. 12 Control flow when the secure world calls a function in the non-secure world [20]

To ensure that no data is leaked through the shared registers, the processor automatically executes the following steps before calling a function in the non-secure world. First, the return address to the secure world along with selected processor state information from the *xPSR* register is pushed on the stack of the secure world. Second, the processor automatically zeros-out all registers from *R0* to *R15* which are not used for parameter passing as well as the processor status register (*PSR*) register. Finally, the processor loads a pseudo return address into the stack pointer (SP) register (*R13*). The value of the return address is called FNC_RETURN and links to a micro-coded operation to retrieve the actual return address stored on the stack of the secure world, once the called non-secure function has been executed. Fig. 13 shows the final register composition before calling a non-secure function.

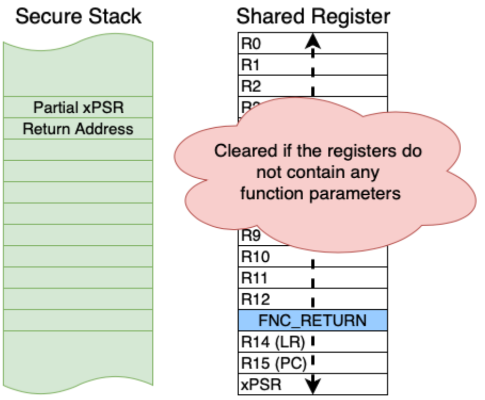


Fig. 13 Secure world stack and shared registers before calling a BLXNS instruction

So far, only the deterministic ways to interact between the worlds have been discussed. However, there is also the possibility of a context switch triggered by an interrupt or exception, e.g. if a non-secure interrupt occurs while the processor is executing code in the secure world. As with function calls, the processor has to ensure that no data is leaked to the non-secure world. The procedure to protect secure data in case of a non-secure interrupt or exception is as follows. First, the processor pushes the content of all shared registers to the secure stack. Additionally to the registers, a signature is added on the secure stack, ensuring the integrity of the register content stored on the secure stack. Afterwards, the registers *R0* to *R12* as well as the *xPSR* register are zeroed out. Fig. 14 shows the register composition before the context switch to the non-secure interrupt or exception is executed.

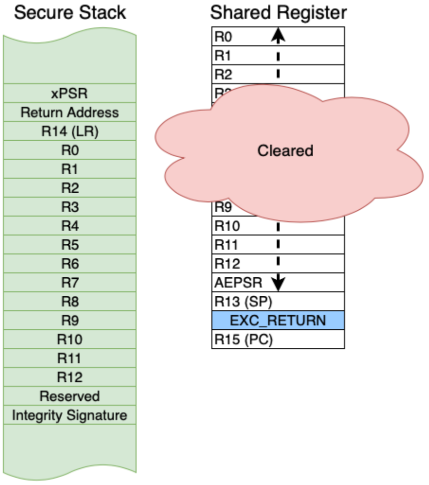
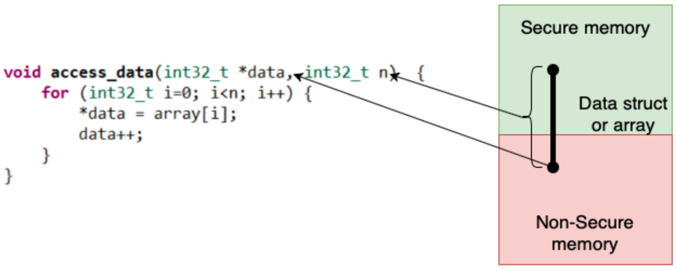


Fig. 14 Secure world stack and shared registers before execution of an ISR

C. Secure data exchange

As context switching, data exchange between the non-secure and secure world may pose a security risk. A secure application should never trust parameters given by the non-secure world. For example, if the non-secure world provides a pointer to an array and a size value, an attacker may try to extend the size value until the array expands into a secure memory region. This may lead to the corruption of secure data. Fig. 15 displays the discussed problem.



```
void access_data(int32_t *data, int32_t n) {
    for (int32_t i=0; i<n; i++) {
        *data = array[i];
        data++;
    }
}
```

Fig. 15 Security risk due to provided parameters from the non-secure world

To mitigate this risk, Arm® has introduced the so-called Test Target (TT) instruction. This instruction allows software to determine the security attribute of a memory location. The security attribute includes access permissions, different security states and privilege levels. Furthermore, if executed in the secure world, the result of the TT instruction also includes the SAU and IDAU configuration of the specified address.

Although the TT instruction cannot be accessed directly by C/C++ code, Arm® provides several intrinsic C functions to make use of the TT instruction. The following two functions are the most important ones:

```
/* Checks address range from p to p + (size – 1) */
void* cmse_check_adress_range(void* p, size_t size,
    int flag)

/* Same as function before, with range from p to
p + (sizeof(p) – 1) */
void* cmse_check_pointed_object(void* p, int flag)
```

These functions assume that memory regions do not overlap each other, which is granted by the ARMv8-M architecture.

IV. Application

To provide hands-on experience from working with SMCUs, a real-world application example has been implemented on the Nordic nRF9160 DK. The application has been implemented with the Zephyr operating system and the nRF Connect SDK [21]. The example covers four topics:

- Secure boot with MCUBoot
- Secure partitioning of memory and peripherals
- Execution of cryptographic operations in the secure world
- Establishment of a (D)TLS session in the secure world

The application example closely represents the control flow of a typical secure firmware project displayed in Fig. 16 with the addition of a secure bootloader.

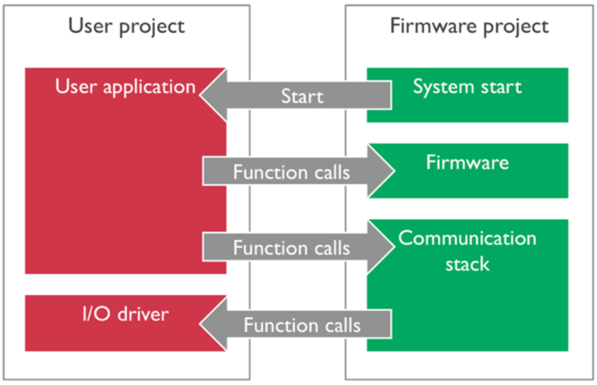


Fig. 16 Control flow on a processor with TrustZone® support [22]

A. *Secure boot with MCUBoot*

Having a secure bootloader is a key feature for a secure embedded application. A secure bootloader prevents embedded devices from running unauthorized or manipulated code. Furthermore, allowing for secure firmware update enables long term support and maintenance of embedded devices. Fig. 17 shows the boot process on an SMCU using MCUBoot. First MCUBoot it-self is executed, verifying the application image stored in Slot_0_S and Slot_0_NS, which is the firmware of the secure and non-secure world respectively. After the initial tasks of the secure world have been executed, the secure world branches to the image stored in Slot_0_NS, starting the main application in the non-secure world. To enable secure firmware update, two additional slots, Slot_1_S, and Slot_1_NS, are required by MCUBoot. With them, both images can be updated and even reverted in case the update does not run properly.

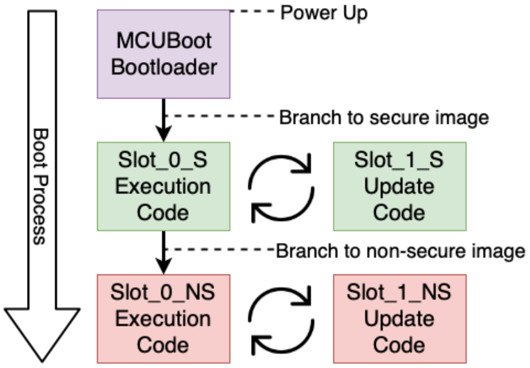


Fig. 17 Boot process with MCUBoot as bootloader

B. *Secure partitioning of memory and peripherals*

Every application using TrustZone® has to do the separation of memory and peripherals into non-secure and secure. This configuration is done in the main function of the secure world which is defined as the starting point of all applications using TrustZone®. A secure partition manager (SPM) application may be used to configure the memory and peripherals. The SPM uses the SAU and IDAU units for the configuration.

C. *Execution of cryptographic operations in the secure world*

To show how the secure world may provide secure firmware for the non-secure world, the application example implements five cryptographic operations as NSC functions. The execution of cryptographic operations in the secure world improves an application on several points. First, only the secure world provides access to the CC310 hardware accelerator, which makes cryptographic operations execute faster and energy efficient. Furthermore, the CC310 provides a sophisticated TRNG, which increases the entropy of generated keys and random numbers. Second, cryptographic sensitive material, e.g. private keys can be made accessible only by the secure world, reducing the risk of an application to get compromised. Lastly, if properly coded, there is no need for mbedTLS within the application in the non-secure world, often allowing to reduce the memory footprint of the application. Fig. 18 illustrates these points based on the generation of a key pair for elliptic curve cryptographic (ECC), where only the public key, in the form of a *uint8_t* array, is returned to the non-secure world.

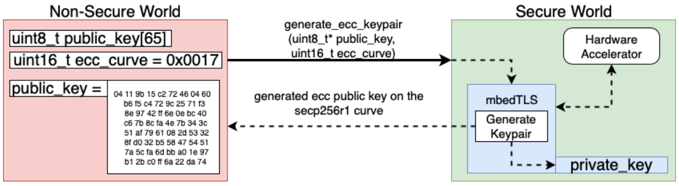


Fig. 18 Generation of an ECC key pair in the secure world

D. *Establishment of a (D)TLS session in the secure world*

The last topic covered by the application example, is the establishment of a secure channel between a client (referred to as node) and a server. For this purpose, a network as displayed in Fig. 19 has been set up. The server is implemented on a Raspberry Pi connected to the local network. To communicate with the server, the node requires connectivity. Since there is no common communication protocol on all the available DKs, a WiFi extension board is used.

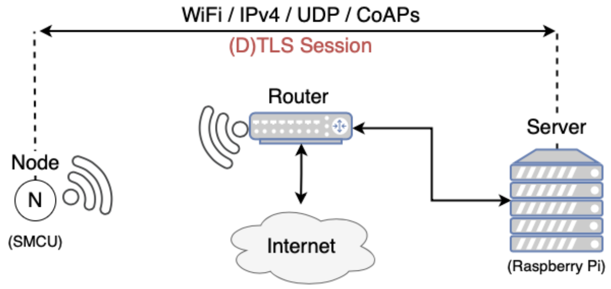


Fig. 19 Network setup

A commonly used protocol to establish a secure channel is the (Datagram) Transport Layer Security (D)TLS [23] protocol. To establish such a secure channel, a so-called handshake has to be executed in which symmetric session keys for authentication and encryption are generated. For this example, (D)TLS has been used, due to the server supporting CoAPs which is based on UDP. As displayed in Fig. 16, the non-secure world is responsible for the I/O driver, i.e. in this case the WiFi module. In contrast, the secure world handles the communication stack, i.e. in this case, the execution of the (D)TLS handshake using mbedTLS. Fig. 20 shows the resulting procedure to execute the (D)TLS handshake.

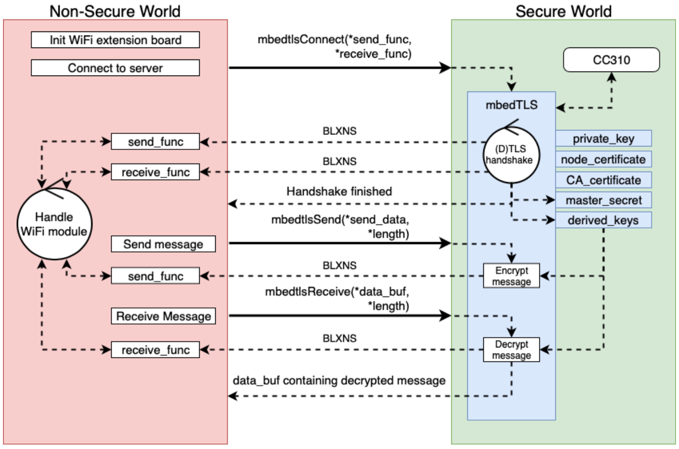


Fig. 20 (D)TLS handshake executed in the secure world

Executing a (D)TLS handshake within the secure world provides the same benefits as with the execution of cryptographic operations. However, their influence on the application might be even bigger, due to two reasons. First, a (D)TLS handshake is energy consuming, due to the execution of elaborate cryptographic operations and the exchange of multiple messages. The result is an intensive period, that may last up to multiple seconds, which may significantly reduce the battery lifetime of a resource-constrained device. Therefore, the support of a crypto accelerator significantly helps to maintain battery lifetime. Second, although the node and certificate authority (CA) certificates may be known to everyone, they have to be protected against manipulation. Storing them in the secure world reduces the risk that certificates may get corrupted and allows to authenticate the node, e.g. preventing intellectual property (IP) theft. Furthermore, having the established symmetric secrets stored in a secure place may also save battery. Due to the secure storage of these secrets, a once established session may be kept open for a long period or be resumed with an abbreviated handshake without reducing the overall security. As a result, the number of performed handshakes is reduced, which also helps to maintain battery lifetime.

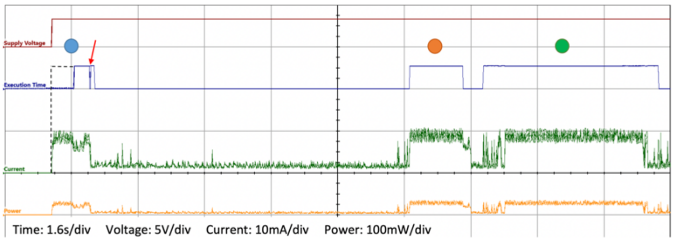
V. Measurements

This section presents measurement results to quantify the effects of TrustZone® on the execution time and the energy consumption of an application. For this purpose, the example application has been measured with and without using TrustZone® on the NRF9160 DK. Furthermore, it is interesting to see the differences between previous MCUs and the new generation of SMCUs. Therefore, the example application has also been implemented and measured on an nRF52840 DK. This measurement serves as a reference for the measurements performed with the nRF9160. To indicate the execution times of the application parts of interest, a GPIO is set high before their execution starts. The supply voltage for both DKs has been set to 3.3V. The measurement setups of the two DKs are displayed in the Appendix.

There are some differences between the measurements with the nRF52840 and the nRF9160. First, due to the absence of TrustZone® support, there is no SPM implementation required on the nRF52840. Second, the nRF52840 DK has native Thread [24] support. Therefore, Thread is used as a communication protocol instead of WiFi. Zephyr initializes Thread as communication protocol during boot and automatically connects to the specified Thread network. As a result, the application with the nRF52840 DK starts only after about 2.5 seconds.

A. Impact of TrustZone® on an application

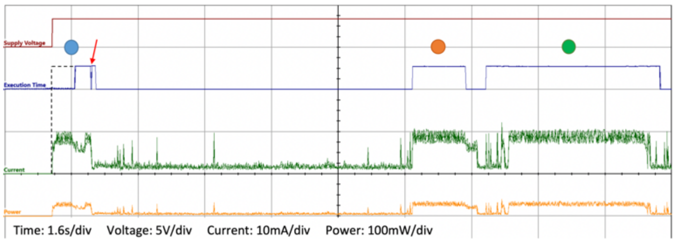
Fig. 21 shows the measurement results of the execution of the cryptographic operations and the (D)TLS handshake in the secure world. The red arrow marks the branch from the secure to the non-secure world. To indicate where the application starts, a GPIO is set high for about 100 ms (amplitude without a color dot). During the gap between the start of the application and the execution of the cryptographic operations, the WiFi module is initialized and connects to the network.



	Operation	Ex. Time [s]	Avg. Current [mA]	Energy [µWh]
●	Secure Boot & SPM	0.95	7.29	6.43
●	Crypto Operations	1.33	8.74	10.70
●	(D)TLS Handshake	4.35	7.47	29.84

Fig. 21 Example application executed with TrustZone® support on the nRF9160 DK

Fig. 22 shows the measurement results, executing the application without TrustZone® support.



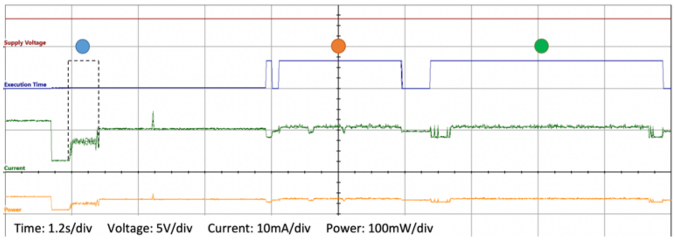
	Operation	Ex. Time [s]	Avg. Current [mA]	Energy [µWh]
●	Secure Boot & SPM	0.96	7.31	6.45
●	Crypto Operations	1.31	8.62	10.45
●	(D)TLS Handshake	4.30	7.50	29.63

Fig. 22 Example application executed with TrustZone® support on the nRF9160 DK

The comparison of the nRF9160 measurement results confirms the claim of Arm® that, the use of TrustZone® has only a minimal impact on execution time and energy consumption. Comparing the execution times show a difference of 20 ms for the execution of the cryptographic operation and 50 ms for the execution of the (D)TLS handshake. The average current remains constant. As a result, the slightly increased energy consumption is due to the increased execution time. Summarizing the results, one can state that the support of TrustZone® only slightly increases the execution time and energy consumption. Therefore, it does not have a substantial impact on an application.

B. Comparison between MCU and SMCU

Fig. 23 shows the application executed on a nRF52840 DK.



	Operation	Ex. Time [s]	Avg. Current [mA]	Energy [µWh]
●	Secure Boot	0.5	7.10	3.50
●	Crypto Operations	2.22	10.74	21.96
●	(D)TLS Handshake	4.19	10.50	40.46

Fig. 23 Example application executed on the nRF52840 DK

The comparison between the nRF52840 and the nRF9160 shows that the boot with MCUBoot takes about the same time on both MCUs, the additional time on the nRF9160 is due to SPM application. However, when comparing the execution times of the cryptographic operations, the nRF9160 is much faster. This is due to the increased processing power of the new M33 processor. Comparing the execution time of the (D)TLS handshake is difficult since the applications use different protocols and hardware to send data, which has a major impact on the execution time and energy consumption. However, 4 seconds is a typical time to execute a (D)TLS handshake. Interestingly, the nRF9160 requires about 2mA less than the nRF52840. As a consequence of the reduced current consumption and execution time, the energy consumption of the nRF9160 is substantially reduced.

VI. Comparison to Secure Elements

There is more than one way to secure embedded devices, secure elements are one of the alternatives to SMCUs. Secure elements are dedicated off-chip solutions, which provide hardware acceleration and tamper-proof memory and at the same time require little energy. Furthermore, secure elements protect against side channel attacks. In a side channel attack, an adversary tries to recover sensitive data from the observation of hardware properties such as energy consumption, execution time or magnetic leakage. Since secure elements are off-chip solutions, a serial interface between the MCU and the secure element is used to communicate, e.g. I^2C. Secure elements provide a sophisticated API written by specialists, which prevents the exposure of sensitive material due to implementation errors. TrustZone® and secure elements, both increase the security level of embedded devices by isolating cryptographic material in memory which is not accessible to the application. In addition, secure elements provide tamper-proof memory, which protects stored data against hardware attacks. This hardware protection increases the security level even further.

Summarizing, secure elements increase the security level due to their hardware protection, which makes them a valid solution if a particularly high level of security is required. However, the downsides of secure elements include the need for extra space on the printed circuit board (PCB), additional cost and the required protection of the serial communication interface between the MCU and the secure elements.

VII. Advantages and Weaknesses of SMCUs

SMCUs increase the security level on embedded devices with the use of TrustZone®, isolating sensitive material and security-related tasks from the application. Furthermore, the presence of a hardware accelerator makes the execution of cryptographic operations faster and energy efficient. Additionally, the ARMv8-M architecture increases processing power at the same time reducing energy consumption. The following list summarizes the advantages of SMCUs:

- TrustZone® support
  - Isolation of cryptographic sensitive material and tasks
  - Secure handling of communication protocols
  - Secure, reliable storage of authentication material
- Cryptographic hardware accelerator support
- Sophisticated random number generator support

As a result, SMCUs are a valuable solution to provide an adequate level of security on embedded devices. However, there are certain security measures SMCUs cannot provide. As embedded devices are deployed in the field, they are exposed to physical attacks, such as hardware and side channel attacks. SMCUs do not provide measures against such physical attacks. Particularly, there is the so-called screaming side channel attack [25] which may effect SMCUs due to having a radio and the processor on the same die. Furthermore, TrustZone® and the other security measures provided by SMCUs are just tools, the developer has to use them in the right way. Therefore, the effort to develop secure firmware is increased and requires specific know-how. The following list summarizes the weaknesses of SMCUs:

- No tamper protection
- Missing measures against physical attacks
- Increased development effort

To also provide tamper protection, Arm® has already released an additional processor, the Cortex-M35P [26] which provides anti-tampering measures. However, there are currently no SMCUs available which are equipped with a Cortex-M35P processor.

VIII. KEY FINDINGS

SMCUs enable the development of secure firmware in the field of embedded IoT, therefore providing great potential for securing millions of IoT devices. Their support for TrustZone® protects the firmware running in the secure world from malicious firmware running in the non-secure world, preventing the exposure of sensitive data. Yet, SMCUs preserve the characteristics of conventional MCUs in terms of interrupt latency and low energy consumption. Although, MCUs, such as the nRF52840, also provide hardware accelerators for cryptographic operations, with SMCUs their usage is restricted to the secure world. As a result, sensitive material used and generated by the hardware accelerator is protected in the secure world, similar to secure elements. However, SMCUs lack measures to prevent hardware and side-channel attacks. If an application has to withstand physical attacks a secure element is an appropriate choice. TrustZone® increases the security level of embedded devices to an acceptable state, which should be the minimal standard for future embedded devices. With additional tools such as a secure bootloader and secure elements, all necessary hardware components are available to build secure embedded IoT devices.

IX. CONCLUSION

This paper provides a feature overview of multiple SMCUs and introduces the concept of TrustZone® on Cortex-M processors. Furthermore, the paper provides experience from working with the nRF9160 SMCU in combination with Zephyr and MCUBoot. The implemented application covers the topic of secure boot, the partition of memory and peripherals into secure and non-secure, the execution of cryptographic operations in the secure world and the establishment of a secure channel executed in the secure world. Furthermore, energy measurements with a conventional MCU and a new SMCU have been performed. Uncovering that the use of TrustZone® only has minimal impact on an application in terms of execution time and energy consumption. Although SMCUs do not provide the same level of security as secure elements, SMCUs increase the security level to an adequate level, having the potential to secure millions of embedded IoT devices.

X. APPENDIX

TABLE I: Hardware components

Identifier	Vendor	Version
nRF9160	Nordic Semiconductor	0.8.2 pca10090
nRF52840	Nordic Semiconductor	1.0.0 pca10056
WiFi ESP Board	mikoBUS	1.0
Raspberry Pi	Raspberry Pi Foundation	3 B+
Power Analyzer N6705B	Keysight	Last calibration 10. Oct 2018

TABLE II: Software components

Identifier	Repository	Hash
Zephyr	Zephyr Project	1378a1c7ac
MCUboot	JuulLabs-OSS	a4db98d
mbedTLS	ARMmbed	53546ea09
nRF Connect SDK	Nordic Semiconductor Playground	b779665
Californium	Eclipse Foundation	9f6f90e5
Border Router	OpenThread	7f9bc33

Fig. 24 shows how the ESP WiFi module is connected with the nRF9160 Dk.

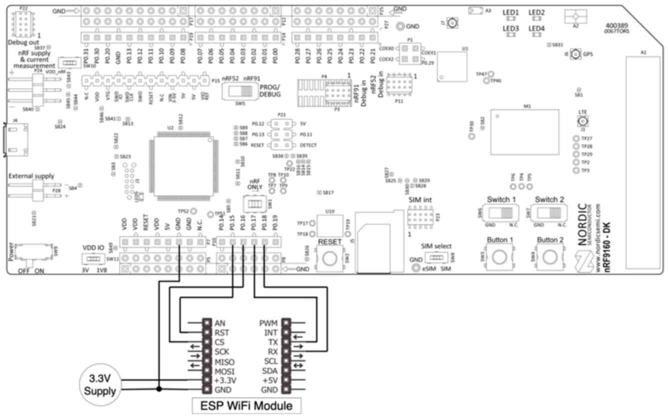


Fig. 24 Connection between nRF9160 and the ESP WiFi module

Fig. 25 shows the measurement setup with the nRF9160 DK.

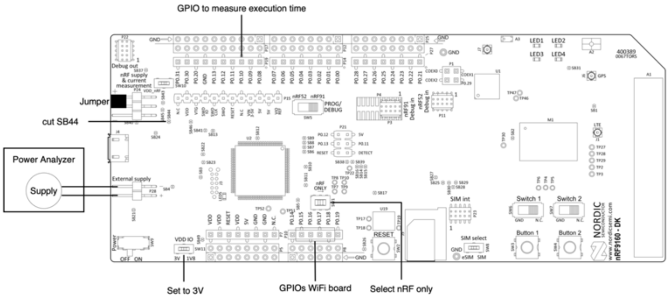


Fig. 25 Measurement setup with the nRF9160

Fig. 26 shows the measurement setup with the nRF52840 DK.

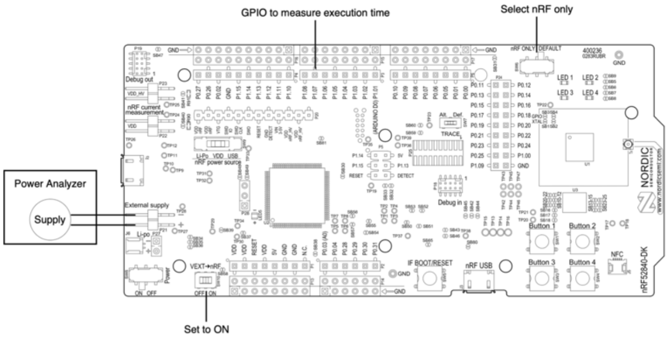


Fig. 26 Measurement setup with the nRF52840

References

[1] D. Smith, Radware Ltd., 10 2018. [Online]. Available: https://blog.radware.com/security/2018/10/iot-botnets-on-the-rise/.

[2] J. Miley, Interesting Engineering, 4 2018. [Online]. Available: https://interestingengineering.com/a-casinos-database-was-hacked-through-a-smart-fish-tank-thermometer.

[3] Linux Foundation, [Online]. Available: https://www.zephyrproject.org.

[4] Arm mbedTLS, 7 2018. [Online]. Available: https://tls.mbed.org.

[5] Juul Labs, [Online]. Available: https://github.com/JuulLabs-OSS/mcuboot.

[6] T. E. Christopher Seidl, 11 2016. [Online]. Available: https://www.arm.com/files/event/2016_ATS_India_C6_Ashok_Bhat.pdf.

[7] German Federal Office for Information Security, 5 2013. [Online]. Available: https://www.bsi.bund.de/DE/Themen/ZertifizierungundAnerkennung/Produktzertifizierung/ZertifizierungnachCC/AnwendungshinweiseundInterpretationen/AIS-Liste.html.

[8] E. Barker, J. Kelsey and J. B. Secretary, "NIST DRAFT Special Publication 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation," 2012. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/sp/800-90c/draft/documents/draft-sp800-90c.pdf.

[9] N. Nayampally, 12 2016. [Online]. Available: https://www.arm.com/files/event/2016_ATS_India_A4_Nandan_Nayampally.pdf.

[10] C. Windeck, 10 2016. [Online]. Available: https://www.heise.de/imgs/18/1/9/1/5/3/0/9/ARM-Cortex-M23-M33-61f194416ab901cc.jpeg.

[11] Microchip, [Online]. Available: https://www.microchip.com/about-us/company-information/about.

[12] Nordic Semiconductor, [Online]. Available: https://www.nordicsemi.com/About-us.

[13] NXP Semiconductors, [Online]. Available: https://www.nxp.com/about/about-nxp/about-nxp:ABOUT-NXP.

[14] STMicroelectronics, [Online]. Available: https://www.st.com/content/st_com/en/about/st_company_information/who-we-are.html.

[15] Microchip, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/SAM-L10-L11-Family-Data-Sheet-DS60001513C.pdf.

[16] N. Semiconductor, "nRF9160, Objective Product Specification," Nordic Semiconductor, [Online]. Available: https://www.nordicsemi.com/-/media/DocLib/Other/Product_Spec/nRF9160OPSv071pdf.pdf.

[17] NXP Semiconductors, [Online]. Available: https://www.nxp.com/docs/en/data-sheet/LPC55S6x.pdf.

[18] STMicroelectronics, [Online]. Available: https://www.st.com/resource/en/data_brief/stm32l562ce.pdf.

[19] Arm Procedure Call Standard, 11 2015. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ihi0042f/IHI0042F_aapcs.pdf.

[20] Arm TrustZone, 2017. [Online]. Available: https://static.docs.arm.com/100690/0200/armv8m_trustzone_technology_100690_0200.pdf.

[21] Nordic Semiconductor, [Online]. Available: http://developer.nordicsemi.com/.NCS_PV/doc/nrf/index.html.

[22] A. Bhat, 12 2016. [Online]. Available: https://www.arm.com/files/event/2016_ATS_India_C6_Ashok_Bhat.pdf.

[23] E. Rescorla and N. Modadugu, *Datagram Transport Layer Security Version 1.2,* RFC Editor, 2012.

[24] T. Group, "What is Thread," [Online]. Available: https://www.threadgroup.org/What-is-Thread.

[25] G. Camurati, S. Poeplau, M. Muench, T. Hayes and A. Francillon, "Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers," 2018. [Online]. Available: http://s3.eurecom.fr/docs/ccs18camuratipreprint.pdf.

[26] Arm, "Cortex-M35P," [Online]. Available: https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m35p.