

**DETERMINING THE PERFORMANCE COSTS IN ESTABLISHING
CRYPTOGRAPHY SERVICES AS PART OF A SECURE ENDPOINT
DEVICE FOR THE INDUSTRIAL INTERNET OF THINGS**

by

Lehlogonolo P.I. Ledwaba

Submitted in partial fulfilment of the requirements for the degree
Master of Science (Applied Sciences)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

November 2017

SUMMARY

DETERMINING THE PERFORMANCE COSTS IN ESTABLISHING CRYPTOGRAPHY SERVICES AS PART OF A SECURE ENDPOINT DEVICE FOR THE INDUSTRIAL INTERNET OF THINGS

by

Lehlogonolo P.I. Ledwaba

Supervisor: Dr. G.P. Hancke
Department: Electrical, Electronic and Computer Engineering
Co-Supervisor: Prof. H.S. Venter
Department: Computer Science
University: University of Pretoria
Degree: Master of Science (Applied Sciences)
Keywords: Cryptography, Cyber-Physical Systems, Industrial Internet of Things, Security, Endpoint security, Performance evaluation

Endpoint devices are integral in the realisation of any industrial cyber-physical system (ICPS) application. As part of the work of promoting safer and more secure industrial Internet of Things (IIoT) networks and devices, the Industrial Internet Consortium (IIC) and the OpenFog Consortium have developed security framework specifications detailing security techniques and technologies that should be employed during the design of an IIoT network. Previous work in establishing cryptographic services on platforms intended for wireless sensor networks (WSN) and the Internet of Things (IoT) has concluded that security mechanisms cannot be implemented using software libraries owing to the lack of memory and processing resources, the longevity requirements of the processor platforms, and the hard real-time requirements of industrial operations. Over a decade has passed since this body of knowledge was created, however, and IoT processors have seen a vast

improvement in the available operating and memory resources while maintaining minimal power consumption.

This study aims to update the body of knowledge regarding the provision of security services on an IoT platform by conducting a detailed analysis regarding the performance of new generation IoT platforms when running software cryptographic services. The research considers execution time, power consumption and memory occupation and works towards a general, implementable design of a secure, IIoT edge device. This is realised by identifying security features recommended for IIoT endpoint devices; identifying currently available security standards and technologies for the IIoT; and highlighting the trade-offs that the application of security will have on device size, performance, memory requirements and monetary cost.

LIST OF ABBREVIATIONS

3DES	Triple DES
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CPS	Cyber-Physical System
CPU	Central Processing Unit
CSP	Critical Security Parameters
CTR	Counter Mode
DARPA	Device Attestation Resilient to Physical Attacks
DES	Data Encryption Standard
DH	Diffie-Hellman
DoS	Denial of Service
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECQV	Elliptic Curve Qu-Vanstone
FIPS	Federal Information Processing Standard
FW	Firmware
GCM	Galois Counter Mode
GMAC	Galois Message Authentication Code
GPIO	General Purpose Input Output
HMAC	Hash-based Message Authentication Code
HSM	Hardware Security Module
HW-RoT	Hardware Root of Trust
ICPS	Industrial Cyber-Physical System
ICT	Information and Communications Technology

IDE	Integrated Development Environment
IIC	Industrial Internet Consortium
IIoT	Industrial Internet of Things
IoT	Internet of Things
MCU	Microcontroller Unit
MPU	Memory Protection Unit
NDA	Non-Disclosure Agreement
NIST	National Institute of Standards and Technology
OS	Operating System
PC	Personal Computer
PCB	Printed Circuit Board
PLCs	Programmable Logic Controllers
PP	Protection Profiles
PRNG	Pseudo Random Number Generator
PUF	Physical Unclonable Function
RAM	Random Access Memory
RAS	Reliability-Availability-Serviceability
RC5	Rivest Cipher 5 algorithm
RC6	Rivest Cipher 6 algorithm
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest-Shamir-Adleman algorithm
RTOS	Real-Time Operating System
SEDA	Scalable Embedded Device Attestation
SHA	Secure Hashing Algorithm
SHA-2	SHA with digest values of 224, 256, 384 or 512 bits
SMART	Secure and Minimal Architecture for (establishing a dynamic) Root of Trust
SoC	Systems on Chip
STM	STMicroelectronics
SW	Software
TCG	Trusted Computing Group
TEA	Tiny Encryption Algorithm
TEE	Trusted Execution Environment

TOE	Target of Evaluation
TPM	Trusted Platform Module
TRNG	True Random Number Generator
WSN	Wireless Sensor Network

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	PROBLEM STATEMENT	1
1.1.1	Context of the problem	1
1.1.2	Research gap	2
1.2	RESEARCH OBJECTIVE AND QUESTIONS	2
1.3	APPROACH.....	3
1.4	RESEARCH GOALS	4
1.5	RESEARCH CONTRIBUTION	5
1.6	RESEARCH OUTPUTS	5
1.7	OVERVIEW OF STUDY	5
CHAPTER 2	LITERATURE STUDY	7
2.1	CHAPTER OBJECTIVES	7
2.2	SECURITY STANDARDS AND FRAMEWORKS FOR THE INDUSTRIAL INTERNET OF THINGS.....	8
2.2.1	The Industrial Internet Consortium Reference and Security Architecture ..	11
2.2.2	The OpenFog Reference Architecture for Fog Computing	16
2.3	DESIGNING A SECURE MOTE FOR THE IIOT	18
2.3.1	Physical Security.....	18
2.3.2	Secure and Trusted Execution	23
2.3.3	Isolation.....	27
2.3.4	Attestation	28
2.3.5	Cryptography	34
2.4	TRADE-OFFS IN ESTABLISHING SECURITY FOR THE IIOT	45
2.4.1	Physical Security.....	45
2.4.2	Secure and Trusted Execution	46
2.4.3	Isolation.....	46

2.4.4	Attestation	47
2.4.5	Cryptography	47
2.5	CHAPTER SUMMARY	54
CHAPTER 3	METHODS.....	56
3.1	CHAPTER OBJECTIVES	56
3.2	AIM	56
3.3	CRYPTOGRAPHIC ALGORITHMS	57
3.4	EQUIPMENT AND TOOLS	58
3.4.1	The ARM Cortex-M Processor Family.....	58
3.4.2	STM Cube MX and X-Cube Cryptographic library	61
3.4.3	Atollic TrueSudio.....	62
3.5	EXPERIMENT METHODOLOGY	63
3.5.1	Determining cryptography execution time	63
3.5.2	Determining cryptography power consumption	66
3.5.3	Determining memory occupation	68
3.6	CHAPTER SUMMARY	69
CHAPTER 4	RESULTS.....	71
4.1	CHAPTER OBJECTIVES	71
4.2	EXECUTION PERFORMANCE	72
4.2.1	AES128-CTR.....	72
4.2.2	ECDSA	77
4.2.3	SHA256.....	80
4.3	POWER CONSUMPTION	85
4.3.1	AES128-CTR.....	85
4.3.2	ECDSA	88
4.3.3	SHA256.....	92
4.4	MEMORY OCCUPATION	93
4.4.1	RAM	94
4.4.2	Flash.....	96
4.5	CHAPTER SUMMARY	98
CHAPTER 5	DISCUSSION.....	99
5.1	CHAPTER OBJECTIVES	99

5.2 PERFORMANCE OF SOFTWARE-IMPLEMENTED CRYPTOGRAPHY	100
5.3 TOWARDS DESIGNING A SECURE MOTE FOR THE IIOT	111
5.4 CHAPTER SUMMARY	114
CHAPTER 6 CONCLUSION	115
6.1 SUMMARY OF CONCLUSIONS	115
6.2 RESEARCH CONTRIBUTION	117
6.3 FUTURE WORK	119
REFERENCES... ..	120

LIST OF FIGURES

Figure 2.1 Industrial internet reference architecture structure	13
Figure 2.2 Energy consumption of Mica2 node as a function of operating time	42
Figure 2.3 Memory occupation of cryptographic algorithms in a Mica2 node as a function of operating time.....	43
Figure 3.1 STM32 Development boards	60
Figure 3.2 Release timeline and operating frequencies of Cortex-M series processors.....	60
Figure 3.3 STM CubeMX graphic configuration for the STM32F767-Nucleo144	61
Figure 3.4 Execution time experiment setup	64
Figure 3.5 Execution time waveforms for a single run of AES128-CTR experiment on STM32F0Discovery	65
Figure 3.6 Physical setup of the power consumption experiment.....	67
Figure 3.7 Execution time and bounded voltage waveform for SHA256 running on the STM32VDiscovery	68
Figure 3.8 Memory occupation of AES128-CTR on the STM32F4Discovery.....	69
Figure 4.1 Execution time of Cortex-M series processors running AES128-CTR	73
Figure 4.2 Execution time of Cortex-M series processors running AES128-CTR (encryption only)	73
Figure 4.3 Execution time of Cortex-M series processors running AES128-CTR (decryption only)	74
Figure 4.4 Execution time of Cortex-M series processors running ECDSA (Sign-Verify)	77
Figure 4.5 Execution time of Cortex-M series processors running ECDSA (Key Gen-Sign-Verify)	78
Figure 4.6 Execution time of Cortex-M series processors running SHA256.....	81
Figure 4.7 Execution time of Cortex-M series processors running SHA256 (hash generation only).....	81

Figure 4.8 Execution time of Cortex-M series processors running SHA256 (message digest checking only)	82
Figure 4.9 Power consumption of Cortex-M series processors running AES128-CTR.....	86
Figure 4.10 Power consumption of Cortex-M series processors running ECDSA (Sign-Verify)	88
Figure 4.11 Power consumption of Cortex-M series processors running ECDSA (Key Gen-Sign-Verify).....	89
Figure 4.12 Power consumption of Cortex-M series processors running SHA256	92
Figure 4.13 Comparison of the percentage RAM occupation of cryptographic algorithms loaded onto Cortex-M processors.....	95
Figure 4.14 Comparison of the percentage Flash occupation of cryptographic algorithms loaded onto Cortex-M processors.....	97
Figure 5.1 Average execution times of cryptographic algorithms on Cortex-M processors	101
Figure 5.2 Average power consumption of cryptographic algorithms on Cortex-M Processors	103
Figure 5.3 Average power consumption per MHz of cryptographic algorithms on Cortex-M processors	104
Figure 5.4 Overall performance of cryptographic algorithms on Cortex-M processors ...	106
Figure 5.5 Energy consumption and execution time comparison for Atmega128L and Cortex-M processors	108
Figure 5.6 Energy consumption per MHz comparison for Atmega128L and Cortex-M processors	109
Figure 5.7 Memory occupation comparison for Atmega128L and Cortex-M processors.	109
Figure 5.8 Energy consumption and execution time comparison for Cortex M0 and M4 processors	112

LIST OF TABLES

Table 2.1 FIPS 140-2 security level definitions	10
Table 2.2 Industrial Internet Consortium security objective breakdown for IIoT endpoint protection	15
Table 2.3 OpenFog Consortium security mechanism recommendations for node security	17
Table 2.4 FIPS 140-2 physical security requirements	21
Table 2.5 Tamper resistance levels for physical security solutions	22
Table 2.6 Specifications for the most common trusted platform modules for IoT applications	24
Table 2.7 Sensor network device configurations	34
Table 2.8 Energy consumption of cryptographic algorithms implemented on Mica nodes	37
Table 2.9 Operating time requirements for cryptographic algorithms on a Mica2 node	39
Table 2.10 Memory occupation for cryptographic algorithms on Mica2 node	40
Table 2.11 Summary of security solutions and trade-offs	48
Table 2.12 Current secure MCUs for IIoT applications	51
Table 3.1 GPIO pins used for cryptographic algorithm timing	64
Table 3.2 Summary of jumper pin numbers for the current consumption module	66
Table 4.1 Execution Time Mean, Standard Deviation and Standard Error for MCUs running AES128-CTR	75
Table 4.2 Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (encryption only)	75
Table 4.3 Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (decryption only)	76
Table 4.4 Execution time mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify)	79
Table 4.5 Execution time mean, standard deviation and standard error for MCUs running ECDSA (Key Gen- Sign-Verify)	79
Table 4.6 Execution time mean, standard deviation and standard error for MCUs running SHA256	83

Table 4.7 Execution time mean, standard deviation and standard error for MCUs running SHA256 (hash generation only)	83
Table 4.8 Execution time mean, standard deviation and standard error for MCUs running SHA256 (message digest checking only)	84
Table 4.9 Power consumption mean, standard deviation and standard error for MCUs running AES128-CTR	87
Table 4.10 Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify)	90
Table 4.11 Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Key Gen- Sign-Verify)	90
Table 4.12 Power consumption mean, standard deviation and standard error for MCUs running SHA256	93
Table 4.13 RAM Occupation of cryptographic algorithms loaded onto Cortex-M series processors	94
Table 4.14 Flash occupation of cryptographic algorithms loaded onto Cortex-M series processors	96

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

The concept of the industrial internet represents the incorporation of the IoT, machinery control and operational techniques, Information and Communications Technology (ICT) as well as people within a larger ICPS to realise the use of advanced data analytics as an effort to improve business outcomes [1]. Industrial systems connected using the industrial internet typically operate in mission-critical environments and have high standards of safety, security and resilience – beyond those demanded of the consumer and commercial sectors – for all components within the network architecture [1]. While new generation devices have improved over the years, securing these endpoint devices is still seen as difficult, owing to a variety of constraints including: limited device energy, memory and processing resources, communication latencies, constrained maintenance windows, real-time or near real-time operation and size restrictions [2]. An endpoint device is defined as a device that can be found at the edge of the IIoT network, such as gateways and sensor nodes forming a WSN [2]. Covering the spectrum of devices at the IIoT network edge, endpoints include sensors, programmable logic controllers (PLCs), and large, high computing cloud servers. Consequently, endpoints may be used as parts of control networks, between multi-communication streams, or as traffic routers inside a cloud infrastructure [2]. Given the broad definition of what constitutes an IIoT endpoint, for the purposes of this research, the definition of an endpoint device is restricted to an embedded IIoT mote, to which sensing capability could be added.

With the development of security frameworks for and the improvement to IIoT endpoint devices, the problem occurs in determining the extent to which security can be implemented onto the endpoint devices without reducing their performance to the point where they are no longer capable of operating mission-critical applications. Focusing further, the question under consideration of this research is whether cryptographic algorithms, providing encryption and decryption services, can be implemented on an IIoT endpoint device without resulting in significant losses in performance and longevity.

1.1.2 Research gap

This study aims to determine the performance trade-offs associated with applying cryptographic services on new generation IIoT devices and highlight a security scheme that is appropriate to use towards securing the low-power, resource-constrained endpoints found at the edge of an industrial cyber-physical network. Currently, a prevailing belief exists, formed approximately a decade ago, that endpoints are limited so severely in available resources that the application of cryptography services on the device is incapacitating. This research wishes to revisit that viewpoint with consideration given to the advances in embedded hardware intended for deployment in the IIoT.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

Given the large scope of the work required to design a secure endpoint efficiently for the IIoT edge, as a starting point, one needs to establish the current capabilities available on platforms intended for the IoT and IIoT. To that end, the following questions will form the focus of this research:

- Can encryption/decryption services be implemented successfully on new generation, low power, IIoT edge devices without significant impact to their operation?
 - What are the time costs associated with applying encryption/decryption services on low power devices?

- What are the associated power consumption and memory utilisation costs in applying encryption/decryption services on IIoT endpoint devices?

In line with the aforementioned research focus established, over the course of the research, this work aims to meet the following objectives:

- Identify the security requirements of IIoT endpoint nodes.
- Identify general purpose and security enabled new-generation IIoT platforms
- Identify open source, standard cryptographic algorithms best suited for application onto an IIoT endpoint node.
- Determine possible performance trade-offs – e.g. power, memory, throughput, or cost – in applying cryptographic techniques, such as encryption and decryption, on an IIoT node.
- Determine the best-suited cryptography scheme for securing a low power, IIoT node.
- Identify the best method to integrate cryptography services as part of the construction of a secure IIoT mote.

1.3 APPROACH

In order to address the research question and meet the research objectives adequately, a systematic approach, which utilises the material covered in the previous steps for progression, was required. The approach taken towards the completion of this research is given below:

- Conduct a state of the art survey and literature review for a prospective conference or journal paper:
 - Determine essential security features for IIoT endpoint
 - Establish the current state of trusted execution and side channel resistance.
 - Determine the best hardware MCU for the construction of 32-bit test mote.
- Identify standard cryptographic algorithms to test on mote and implement.

- Determine device cryptographic communication delays and execution times for encryption/decryption activities using a selected group of representative endpoints:
 - Hardware-implemented cryptography.
 - Software-implemented cryptography.
- Identify the least invasive cryptography implementation for IIoT endpoints based on observed trade-offs.
- Implement cryptography as part of a security framework design for a secure IIoT endpoint device.
- Summarise findings and prepare paper submission to an identified journal.

1.4 RESEARCH GOALS

The vulnerability of ICPSs to security attacks is a continuous concern as more deployments are being established globally. Network endpoint devices are areas of high security concern and vulnerability as the typical scale of industrial deployments means that the devices are left unattended for extended periods of time, making them vulnerable to physical tampering and malicious attacks as mechanisms to access and compromise the wider ICPS.

It is the broader goal of this research to provide a security implementation design for IIoT endpoint devices that would form a base from which the wider industrial sector can adjust and improve, depending on the specific application requirements. This would result in the formation of secure IIoT network deployments from the onset of the adoption of the Industrie 4.0 paradigm, instead of the deployment of unsecured devices for which security needs to be established retrospectively. The targeted goal of the research conducted within this study is to establish the operating capabilities of currently available IIoT platforms in supporting software cryptographic services; as a starting point towards the hardware selection processes required for the initial design of a secure endpoint device.

1.5 RESEARCH CONTRIBUTION

Current work in determining the performance costs of applying cryptographic algorithms on motes used in the IIoT has primarily focused on older generation, 8 or 16-bit platforms. With technology for the IoT having improved over the years, this study aims to determine the performance costs associated with applying cryptographic services, such as encryption and decryption, on new generation, 32-bit platforms and to highlight an appropriate security scheme that could be used to secure the low power, resource-constrained devices found at the edge of an industrial cyber-physical network without a significant loss of their operational capability.

1.6 RESEARCH OUTPUTS

The following publications have been derived from the research conducted within this study:

L.P.I. Ledwaba, G.P. Hancke, H.S. Venter, S.J. Isaac. “Trade-Offs in Securing an Industrial Internet of Things Endpoint Device,” to be published.

L.P.I. Ledwaba, G.P. Hancke, H.S. Venter, S.J. Isaac. “Performance Costs towards Securing New-Generation Industrial Internet-of-Energy Endpoint Devices,” to be published.

L.P.I. Ledwaba, G.P. Hancke, H.S. Venter, S.J. Isaac. “Designing Secure Endpoint Devices for the Smart Mine Fog,” to be published.

1.7 OVERVIEW OF STUDY

A literature study on the current state of the art was performed and is reflected in Chapter 2. Open security architectures for the industrial internet were identified and detailed research was conducted into the security recommendations made by the architectures.

Technologies and platforms currently available for the IIoT towards implementing the recommendations were identified alongside the associated trade-offs seen with the establishment of security in the IIoT.

In Chapter 3, the equipment and tools used in conducting this research are listed, and the main research methodology is introduced and discussed.

In Chapter 4, the results obtained for the execution times, power consumption and memory occupation for the software-secured Cortex-M series processors, are provided.

In Chapter 5, the results presented in Chapter 4 are analysed in further detail and the main observations and recommendations for the use of software cryptographic services with the Cortex-M series are given.

In Chapter 6, concluding remarks regarding the ability of new generation processors to provide security services are made, and areas of future work are identified.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OBJECTIVES

The addition of computing capability to industrial processes brings a variety of challenges. Standardisation for the production of IoT devices, their communication protocols, and the degree of security that the devices are capable of providing is essential for deployment into industrial processes with strict safety guidelines.

In this chapter, an overview of the current literature for security architectures, requirements, mechanisms and trade-offs for the IIoT is provided.

In Section 2.2 an introduction to the open security architectures developed by the IIC and the OpenFog Consortium is given. The recommendations made by both organisations regarding the design of a secure IIoT endpoint device are highlighted.

In Section 2.3 a detailed look is taken into the recommendations made by the architectures; highlighting the technologies currently available for use in the IIoT sector and identifying areas in which further development is required.

In Section 2.4 the chapter is concluded and serves to identify the trade-offs associated with the application of security on an IIoT endpoint node.

Section 2.5 provides a brief summary of the points covered in the literary review and serves to conclude the chapter.

2.2 SECURITY STANDARDS AND FRAMEWORKS FOR THE INDUSTRIAL INTERNET OF THINGS

Building the IIoT is based on the use of cyber-physical systems (CPSs) [3]. A CPS represents a joining of physical processes, such as actuation, control and sensing, with cyber processing through the use of technologies equipped with ICT capability [4], [5]. The idea behind the CPS paradigm is one of interconnecting processes and equipment that have previously operated in isolation to form a single self-aware, self-actuating and self-healing network capable of determining the optimal conditions for operation in real time [5]. In the context of smart factories, the production lines will be aware of their own health, processing specification, and identity within the larger process chain [3]. This serves to increase the efficiency of production processes and the reduction of excessive waste; improving the carbon footprint of production processes. The scale at which IIoT deployments are usually conducted requires that future solutions developed for the IIoT should be highly scalable [3]. The availability and integrity of the IIoT network should always be preserved to be able to meet strict, real-time deadlines and to prevent cascading failures, which could result in physical harm to humans or to the operating environment [3]. The constraint of resources such as available power, processing and memory, as well as a required operational period of months or years, means that developed IIoT solutions should be able to support low power operations, and to occupy and use a small portion of the memory and processor resources [3].

The challenges seen with IIoT devices also extend into the domain of security. IIoT devices are vulnerable to physical attacks, such as tampering and theft as large-scale deployments are often unmonitored [3]. The devices are also subject to eavesdropping, man-in-the-middle, denial-of-service, and masquerade attacks as the peer-to-peer nature of an IIoT network means that wireless communications take the form of open, broadcast networks without few mechanisms to verify the authenticity of the data received [3], [6]. Implementing traditional IT security techniques fails to secure these devices as the delays introduced often compromise the availability of the system for security, which is

unacceptable for industrial contexts. Specially designed security solutions for the IIoT context capable of securing networks while minimising trade-offs in power consumption, processing capacity and memory footprint are high priorities in order to be able to meet the strict safety standards required by the industrial sector.

Security standards, frameworks and architectures can be used to aid in meeting the safety requirements of the industrial sector by defining what security is expected for an IIoT endpoint and the depth to which the security should be provided. Standards can also be used to validate the security mechanisms and solutions designed to secure IIoT endpoints and further allow for vendor culpability. One such example is the Federal Information Processing Standard (FIPS) 140-2 Security Requirements for Cryptographic Modules standard [7]. Table 2.1 gives a brief definition of the security levels identified by the FIPS standard and their minimum requirements for a cryptographic module. It defines four (4) levels for which security can be established across the various entities of a module implementing cryptographic processes, such as the physical security, operating environment, and user and device authentication. For each increasing level of security, the requirements of the previous level should be met, unless they are superseded by the requirements of the current level. Certain areas of FIPS 140-2 can also be combined with the Common Criteria (CC) Protection Profiles (PP) for further certification.

Table 2.1 FIPS 140-2 security level definitions

	Requirements
Security Level 1	Lowest level of security
	No specific physical security mechanisms required beyond production-grade components
	Software (SW) and Firmware (FW) can be executed on unevaluated operating system (OS)
Security Level 2	Tamper-evidence mechanisms for module
	Role-based authentication mechanisms
	SW and FW only executable on OS that meets functional requirements from CC PP list
	OS evaluated at CC evaluation assurance level (EAL) EAL2 or higher
Security Level 3	Tamper detection and zeroization response when removable cover/doors are opened
	Identity-based authentication
	Entry and output of plaintext CSPs should be performed by physically separated ports or using a logically separated, trusted path
	SW and FW only executable on OS meeting functional requirements in the PP list with additional Trusted Path (FTP_TRP.1) requirement
	OS is evaluated at CC evaluation assurance level EAL3 or higher with additional assurance requirement of Informal Target of Evaluation (TOE) Security Policy Model (ADV_SPM.1)
	Equivalently evaluated trusted OS may be used

	Requirements
Security Level 4	Highest level of security
	Physical security provides a complete envelope of protection with detection of and immediate zeroization during tamper events
	Protect against security compromise owing to environmental conditions or fluctuations outside normal operation ranges
	Undergo rigorous environment failure testing
	SW and FW only executable on OS that meets functional requirements specified in Level 3
	OS is evaluated at CC evaluation assurance level EAL4 or higher
	Equivalently evaluated trusted OS may be used
	Useful for operation in physically unprotected environments

Source: Security Requirements for Cryptographic Modules. FIPS 140-2. 2001 [7].

In addition to the standard for cryptographic modules, the Industrial Internet Consortium (IIC) and the OpenFog Consortium have developed frameworks detailing the security requirements of an IIoT and fog computing network. The framework requirements complement each other and serve to give a detailed guideline as to which security features and capabilities are needed at the edge of the IIoT/Fog space. The following sections serve to introduce the security frameworks and take a detailed look into the requirements and recommendations made towards the design of a secure IIoT endpoint device.

2.2.1 The Industrial Internet Consortium Reference and Security Architecture

The IIC is a collaboration between various businesses and academic institutions working towards improving and accelerating the move towards the IIoT. The consortium conducts work in five of the main IIoT industries, namely healthcare, manufacturing, smart cities, transportation and energy. In an effort towards standardising the manner in which IIoT networks are developed and deployed, a general and security architecture was published after heavy input from the many member organisations.

The reference architecture proposed by the IIC aims to make the industrial internet easily understandable and supported by “widely applicable, standard-based, open architecture frameworks and reference architectures” [1] regardless of vendor in order to ensure that future technologies can be easily integrated and are interoperable, allowing for the faster expansion of industrial internet networks into key and stressed application areas.

The architecture defines four main viewpoints- business, usage, functional and implementation – which can be further decomposed to address various domains in the construction of an IIoT network [1].

The **business** viewpoint represents the beginning of the network design process and serves to identify the relevant stakeholders within the system and their business vision, values and objectives in the establishment of the industrial internet system, in both business and regulatory contexts [1].

The **usage** viewpoint serves to address the concerns in the use of the IIoT system and typically consists of activity sequences involving human or logical users [1].

The **implementation** viewpoint focuses on the technologies that would be needed to implement the functional components of the IIoT system, their required communication schemes and their lifecycle procedures [1].

The **functional** viewpoint gives focus to the functional components within an industrial internet system: how they interrelate with other components, how they are to be structured, what interfaces are needed; which interactions are required between them; and what the relations and interactions of the system are with external elements in the environment [1].

This viewpoint is the most important regarding the development of a successful security solution for IIoT systems and will form part of the main focus of this research. The full architecture details may be found at [1], [2].

The functional viewpoint can be decomposed further, giving the control, operations, information, application and business domains. Each functional domain operates under a different degree of granularity and runs on different temporal cycles. As one moves up the domains, the coarseness seen within domain interactions increases; the temporal cycles become longer and the scope of impact for the network becomes larger. As information moves up the functional domains, the scope of information becomes richer and broader, as new information and intelligence emerge from the larger contexts [1]. The control domain presents the representation of the functions that are performed by the industrial control systems, ranging from fine-grained, closed loops, sensing, which is the data retrieval from sensor nodes, application of control rules and logic as well as the excursion of control over physical systems through actuators, also known as actuation [1]. It is within this domain that the security of IIoT endpoints is to be applied. To visualise the structure of the main reference architecture more clearly, Figure 2.1 provides a graphical breakdown of the highlighted portions of the architecture with a focus on the areas applicable to the security of IIoT endpoint devices. Double arrows are used to illustrate the duplex nature of information flow between the various viewpoints and domains.

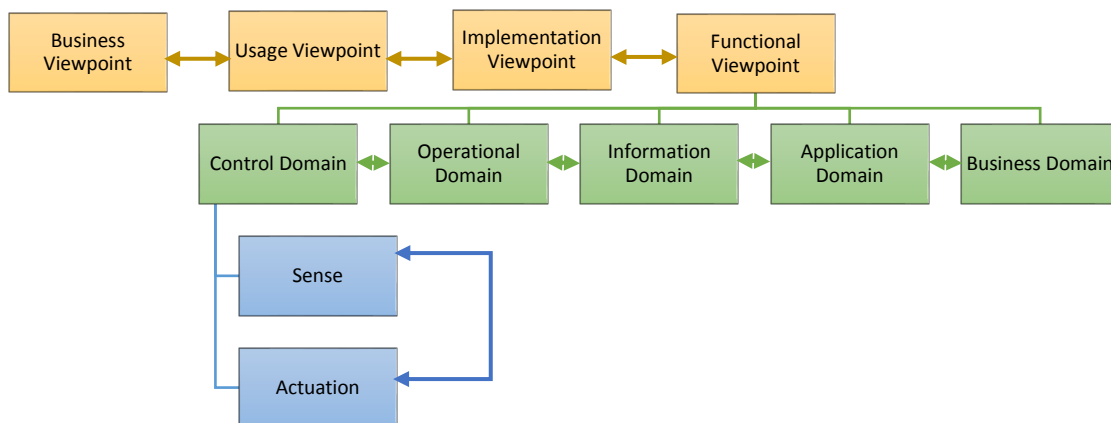


Figure 2.1 Industrial internet reference architecture structure

The security requirements for an industrial internet network development are given in a complementary security framework, which is to be used in conjunction with the reference framework. Six inter-operational building blocks, organised within three layers, form the functional viewpoint of the security framework. The top layer comprises four (4) foundations, namely “endpoint protection, communication and connectivity protection, security monitoring and analysis and security configuration management” [2]. The foundations are then “supported by a data protection layer and a system-wide security model and policy layer” [2].

Endpoint protection exists to implement defensive capabilities on devices typically found at the edge of an industrial network and in the cloud. The primary security concerns for this portion of the network include “physical security functions, cyber security techniques, and authoritative identity” [2]. This protection alone is insufficient for protecting the whole industrial network as the endpoints must have the ability to communicate with each other and with other network devices, and these communications may become a source of vulnerability; however, it provides a solid base upon which further security can be built [2]. Without this base, the deployment would still be highly vulnerable to security attacks despite how much security is applied to other parts of the network.

Table 2.2 provides a summary of the functions and techniques recommended for endpoint security in the IIoT security framework. Also highlighted are the security concerns that the security functions aim to address, given in terms of the Availability-Integrity-Confidentiality triad of security.

Table 2.2 Industrial Internet Consortium security objective breakdown for IIoT endpoint protection

Functions and Techniques	Security Objectives		
	Availability	Integrity	Confidentiality
Physical Security	X	X	X
Trust		X	
Identity	X	X	X
Access Control	X	X	X
Integrity Protection		X	
Data Protection	X	X	X
Monitoring and analysis	X	X	X
Configuration and Management	X	X	X
Cryptographic techniques		X	X
Isolation techniques	X	X	X

Source: Industrial Internet Consortium. (2017, January) Industrial Internet Security Framework. [Online]. [2]

The main problem with the establishment of any security mechanism is the resulting trade-offs that occur as a result of allocating additional resources towards protecting the device from malicious activities. These trade-offs are of great concern in the context of IIoT devices as these devices are often more highly resource-constrained as compared to traditional ICT devices, and are required to operate at low power for months or years after their initial deployment. Adding security capability has the potential to deplete the endpoint resources or introduce delays such that the device becomes unsuitable for the real-time, mission-critical contexts in which it is required to operate. A more detailed look into the trade-offs associated with the identified security functions is given in Section 2.3 of this work.

2.2.2 The OpenFog Reference Architecture for Fog Computing

The OpenFog Reference Architecture for Fog Computing was developed by the OpenFog Consortium in respect of the need for an open, fog computing architecture capable of ensuring interoperable and secure systems, and one that is independent of but fully supported by the wider vendor space [8]. As defined by the OpenFog Consortium, fog computing architectures are used to “selectively move comput[ing], storage, communication, control and decision making closer to the network edge where data is being generated in order to solve the limitations in current infrastructure to enable mission-critical, data-dense use cases” [8], essentially allowing for the IIoT edge to interface with wider cloud services as fog computing maintains the benefits of a cloud computing scheme [8].

The reference architecture defines eight main pillars– “security, scalability, openness, autonomy, RAS (reliability-availability and serviceability), agility, hierarchy and programmability” [8] – as well as the relevant stakeholders and their roles in the wider fog value chain. These include silicon manufacturers, application developers, operating systems, etc. [8]. As with the IIC reference architecture, this study will focus on the security pillar defined by the OpenFog architecture, looking more specifically into the requirements of node security. The full architecture details may be found in [8].

The security pillar describes the functions and mechanisms that could be applied to secure a fog node, from the silicon utilised in the node design to the software applications used on and with the node. Privacy, anonymity, integrity, trust, attestation, verification and measurement are identified by the architecture as key security attributes which should be guaranteed on a node to the best of one’s ability [8]. As a basis for a secure design, a secure node must provide an immutable root of trust, preferably hardware-based. The root of trust should then be attestable by the software agents running within and throughout the fog infrastructure. Edge nodes should provide the first point of access control and encryption within the wider network in addition to providing contextual integrity, isolation and control aggregation of privacy-sensitive data prior to their departure from the network

edge. Should there be any network components that cannot be attestable, they should be prevented from participating within and with the fog nodes and should be deemed to provide data that is not fully trustworthy [8].

A detailed breakdown of the security mechanism recommendations made within the architecture is given in Table 2.3.

Table 2.3 OpenFog Consortium security mechanism recommendations for node security

Functions and Techniques	Security Mechanisms
Physical Security and Anti-Tamper mechanisms	Tamper resistance, evidence, detection and response
Trusted Computing Base	Hardware root of trust (HW-RoT), secure or verified boot, trusted or measured boot, secure boot processes
Identification	Immutable identifier with attestation
Attestation	Remote attestation across multiple interfaces
Cryptographic Functions	AES with at least 128-bit keys, 3DES, DH, RSA, DSA, ECDH, ECDSA, ECQV, SHA-2 to SHA-5, True RNG, CCM, GCM, GMAC, CMAC, HMAC
Integrity	Run-time integrity checking and introspection

Source: OpenFog Consortium. (2017, February). OpenFog Reference Architecture for Fog Computing. USA. [Online]. [8]

Comparing the architectures developed by the IIC and the OpenFog Consortium, one can see that they are complementary in the recommendations made for node security by both organisations. Where the IIC provides a more general guideline as to what functions should be included and the objectives that they should meet, the OpenFog Consortium provides a recommendation as to the mechanisms that could be used in order to provide those functions. Combining the two architectures gives a solid, standard base design upon which a secure endpoint can be derived, as the uncertainty associated with the required functions

for node security and the tools that are to be used in order to meet the objectives set for the functions have been removed with the publication of the two open architectures. A more detailed exploration of the capabilities of the security mechanisms recommended by the OpenFog Consortium is provided in the following section.

2.3 DESIGNING A SECURE MOTE FOR THE IIOT

A variety of security mechanisms and techniques for IIoT endpoint devices can address the security concerns and recommendations highlighted by the two IIoT frameworks. The following sections take a detailed look into the existing security technologies currently available in an effort to identify the trade-offs that occur once a security structure is implemented on an endpoint node.

2.3.1 Physical Security

One of the main challenges with the deployment of large-scale IIoT networks is securing the endpoints physically against attack. Endpoints can easily be removed, damaged, or tampered with as vast deployments result in no possibility for continuous monitoring. Endpoints can also be exposed to the outside elements and damage can occur to the interior electronics should there be any water leakage, excessive dust, or connection interruptions by small animals.

To safeguard the endpoints physically, the use of tamper-resistant hardware components and secure devices should be employed [2]. Mechanisms to detect tampering may be built into the device casing and changes to the device hardware should be reported, with the compromised endpoint being isolated from interaction with the remainder of the network [2]. Essential components should ideally be tagged to enable tracing and deactivation of the components when used outside of their configured contexts, providing a deterrent from component theft [2].

IIoT endpoint devices are vulnerable to four (4) main types of attack which arise as a result of compromised physical security: invasive, non-invasive, fault injection and software attacks [9]. Invasive attacks require the physical capture of the endpoint and often involve physical intrusion at device level; where physical intrusion occurs to the product enclosure, or at chip level; where intrusion occurs to the chip packaging [9], [10]. Non-invasive attacks do not include physical intrusion or damage to the endpoint device but are the result of observing the behaviour of the endpoint as security operations are carried out [9]. Side-channel attacks, such as timing analysis attacks, electromagnetic analysis and power analysis attacks, are examples of common endpoint non-invasive attacks [9]. Fault injection attacks occur when the attacker alters the environment or operating conditions of the IIoT endpoint in order to initiate a malfunction that compromises device security [9]. Over-or under-voltage attacks, over-or under-temperature attacks and timing attacks are common examples of fault injection attacks [9]. Software attacks are typically launched through the communication interfaces of the device, such as debug interfaces, programming interfaces and communication interfaces [9].

The vastness of IIoT endpoint deployments means that it is highly infeasible to completely prevent node capture [10]. Tamper protection mechanisms, therefore, need to be employed to improve the physical security of isolated network devices as the first building block towards securing the entire IIoT network. Complete physical security solutions require the inclusion of tamper detection, tamper response, tamper resistance, if possible, and tamper evidence logging [9].

Starting at the outside, device enclosures should be tamper protected. This can be achieved through the use of enclosure monitoring sensors such as light, temperature, pressure or vibration sensors, which can be connected to tamper input pins on the MCU [9]. On-Off switches embedded within the product enclosure and connected to a tamper detection pin on the MCU can provide a low-power monitoring solution [9]. Printed circuit board (PCB) tamper mesh can be employed for active tamper monitoring of the MCU packaging but is typically only employed on MCUs that are marketed as secure or security hardened and cannot be incorporated on an existing MCU. On- or off-chip voltage and temperature

sensors can be employed for monitoring of voltage- or temperature-based fault injection attacks, generating a tamper event for the device should the sensed device conditions fall into predetermined tamper event thresholds [9]. On-chip sensors are typically very low power and can remain in an enabled state for continuous monitoring [9]. Counteracting power analysis attacks may also occur in a variety of ways, either through the use of hardware or software solutions. Electromagnetic leakage shields could be incorporated within the device or chip enclosure. The addition of amplitude or temporal noise to side channels could be done in order decrease signal to noise ratios, or physical unclonable functions (PUFs) may be employed using the characteristics and fluctuations of the physical device; making the functions unique to the chip and more difficult for an attacker to duplicate [11]. The technology of PUFs was employed by Microsemi in addition to other tamper protection mechanisms, such as anti-tamper mesh, for securing their low-power, field programmable devices such as the IGLOO2 [11]. Having secured the endpoints against tampering, it is important to be able to log and respond to tamper events as they occur. This includes having the device enter a safe, non-operational mode to prevent it being used to infiltrate the wider network, sending a highly visible notice of tampering to the main system with event timestamps and source logs and immediately erasing sensitive security data, such as a master key, from the device [9].

Various efforts are being made to standardise what physical security measures are needed for embedded solutions. Standardisation, licensing or certification specifications are mechanisms which can be used as guidelines in the design of a security solution and to test for compliance. Table 2.4 gives the requirements for physical security, as defined in the FIPS 140-2 standard, for the four (4) security levels given in Table 2.1.

Table 2.4 FIPS 140-2 physical security requirements

	General Requirements	Multiple-Chip Embedded Cryptographic Modules
Security Level 1	Production-grade components with standard passivation	Production-grade enclosure or removable cover
	Plaintext keys and unprotected CPS should be zeroized during physical maintenance	
Security Level 2	Evidence of tampering	Opaque tamper evident encapsulating material or enclosure with tamper evident seal or pick-resistant locks for doors or removable covers
Security Level 3	Automatic zeroization when accessing the maintenance interface	Hard opaque potting material encapsulation of multiple chip circuitry embodiment or applicable multiple chip standalone security level requirements
	Tamper response and zeroization circuitry	
	Protected vents preventing undetected physical probing	
Security Level 4	EFP or EFT for temperature and voltage	Tamper detection envelope with tamper response and zeroization circuitry

Source: Security Requirements for Cryptographic Modules. FIPS Standard 140-2. 2001 [7].

Table 2.5 gives the six levels of physical security protection as defined by IBM, which also have the possibility of intermediary levels [12]. The levels were developed in an effort to have a standardised method by which to quantify the degree to which a device is tamper-resistant [12].

Table 2.5 Tamper resistance levels for physical security solutions

Tamper Resistance Level	Definition
ZERO	No security features have been used on the system. Components are easily accessible and open to inspection. No cost associated with tampering.
LOW	Some security features have been used, but they are compromised with minimal tools, time and at low cost less than \$1 000 (~ R14 000) in total.
MODL	Security features are capable of protecting against most low-cost attacks. Specialist knowledge and more costly tools, up to approximately \$10 000 (approx. R140 000) are required.
MOD	Specialised tools, up to approximately \$100 000 (~ R1 400 000), and knowledge are required in order to defeat the security features implemented.
MODH	Customised security protection means that highly specialised equipment, up to approximately \$1 000 000 (~ R14 000 000), skills and knowledge is required for equipment use and attack. Group of skilled, specialised attackers may be required to perform attack sequences.
HIGH	All known attacks are defeated by the security features, and research is required to find new attacks capable of defeating the security features. Very highly specialised equipment in excess of \$1 000 000 (in excess of ~ R14 000 000) is required, which may need to be built; however, success of the attack is not guaranteed. Attacks most likely to be carried out by government-funded organisations.

Source: Reproduced from [12] with permission of Springer in the format Thesis/Dissertation via Copyright Clearance Center.

Most recently, Rambus Incorporated has begun licensing cryptographic modules compliant with their defined standard for protection against side channel attacks. The testing process, which is conducted as part of their validation program at accredited, independent testing labs, looks for the implementation of security countermeasures against side-channel attacks, such as leakage reduction, noise introduction, obfuscation, protocol level countermeasures and the incorporation of randomness [13], [14]. Certified solutions are then identified by the DPA Security lock logo with current licensees including Atmel, Infineon, Microsemi, NXP Semiconductors and ST Microelectronics, amongst others [14].

2.3.2 Secure and Trusted Execution

For an industrial internet application, it is essential to define metrics of trust for network components, communications, and maintenance installations. Trust, for computing purposes, can be identified as either static or dynamic. Static trust is based on “evaluations against a specific set of security requirements” such as the multiple international standards for security [15]. Dynamic trust is highly dependent on the continued running state of the system under consideration, with trust being measured throughout the lifecycle of the system. Here, fundamentally, trust is determined through the existence of a secure and reliable means within the system capable of providing evidence that the trust state is unchanged; with the system remaining in an expected, secure state [15]. With consideration of the definition of trust in computing, the IIC framework recommends establishing a root of trust from which mechanisms for identification and integrity checking can be derived. The root of trust exists to establish initial confidence within the system operations, which then further support the establishment of confidence in knowing that entities requesting network access are both authorised to access network resources and that they cannot access resources for which they do not have access permission [2]. The root of trust also aids with establishing network integrity by providing a baseline for identifying and preventing unauthorised access attempts [2].

To create a secure network root of trust, the security framework recommends the use of a HRoT mechanism such as a hardware security module (HSM) [2]. HSMs are Systems on Chip (SoC) solutions that can be used to provide minimum cryptographic functions such as encryption, decryption, key generation, digital signing, and hashing [16]. The chips also offer a measure of physical tamper resistance to prevent key capture [16]. Employing the use of an HSM provides a number of advantages and disadvantages. A validated HSM is ensured as trusted as it has passed a baseline security standard as a result of various security tests performed at accredited testing facilities. The devices utilise widely accepted and open, secure cryptographic algorithms; they provide strong random number generation that is critical for many cryptographic functions and, in the event of detected physical tampering, the device will erase all sensitive data [16]. One of the main drawbacks in the

use of an HSM, in general, is the difficulty in employing future upgrades. As security is provided by a physical device in the IoT endpoint circuitry, in the event where a weakness is exposed in a cryptographic algorithm, a new upgrade would not be possible unless the chip is changed for a newer, backwards-compatible chip which addresses the exposed weakness [16].

One example of an HSM is the Trusted Platform Module (TPM) designed according to the TPM standard created by the Trusted Computing Group. A TPM is capable of storing security artefacts used to authenticate a platform as well as to store platform measurements securely that help ensure its trustworthiness [17]. Version 1.2 of the TPM standard provides widely known cryptographic algorithms such as RSA, SHA-1 and HMAC as these algorithms have been thoroughly tested and gradually improved over the period during which they have been released into the public domain [17]. TPMs would serve to isolate security and cryptographic functions physically away from the normal operations of an IoT endpoint as these functions occur on a separate chip from the endpoint CPU. Brief specifications for the three most popular TPMs, with their vendors, for IoT applications, are highlighted in Table 2.6.

Table 2.6 Specifications for the most common trusted platform modules for IoT applications

Chip	Supplier	TCG Std	Security	Key Size	Speed	Cost (\$)
AT97SC3205 [18]	Atmel	1.2	RSA AES SHA-1 SHA-2	2048	2048 RSA: 200ms 64-byte SHA-1: 20 μ s	4.00
SLB9665XQ2.0 [19]	Infineon	1.2 2.0	RSA 2048 ECC256 SHA-256	2048	Not Given	2.00
ST33TPM12LPC [20]	STM	1.2	RSA 512- 2048 AES128 SHA-1 SHA-256	2048	64- byte SHA-1: 155 μ s 2048 Sign: 150ms 1024 Sign: 30ms	Not Given

Looking at the specifications, several metrics immediately stand out. Price-wise, the cost of an individual TPM chip is competitive when compared to the prices of other integrated chips with the individual prices per chip dropping as the number of chips ordered increases.

Of the three TPMs, only the Infineon module gives an estimate of expected current consumption with 2.5 mA used for the chip's Active mode, 0.9 mA used for the chip's Sleep mode, and 150 μ A used for the chip's Sleep Mode with a stopped clock [19]. The Atmel and STM datasheets only indicate that both TPMs support low power modes of operation [18], [20]. The speed at which cryptographic operations are performed could have a detrimental effect on the usefulness of the IoT endpoint. While SHA-1 calculations can be done within microseconds, RSA calculations average out at hundreds of milliseconds. This added delay to the transmission of information from the endpoint has the potential to be unacceptably long in mission-critical applications where the transmission of and actuation on endpoint readings is required in real or near real time. Additional care would need to be taken into researching the maximum accepted transmission delay tolerance for the applications in which the TPM-secured endpoint is to be used to ensure that the cryptographic function selected does not infringe upon the maximum threshold time.

After having established trust in the network operation, establishing trust in network users is the next challenge to be handled. The use of credentials verifies the identity of the various endpoints communicating within the network and can be used to establish varying levels of trust and, consequently, varying levels of access privilege [2]. Choosing an appropriate credential scheme to be applied to endpoints is highly dependent on the uniqueness and strength of the credentials and the context in which the endpoint will be operating [2]. Care needs to be taken to ensure that the credentials offer sufficient uniqueness and strength so as to prevent the falsification of an endpoint's identity [2]. ISO/IEC 24760-1 [21] provides detailed guidelines in determining the three levels of trust – identity, unique identity and secure identity – for endpoint identities, and the Industrie

4.0 documentation [22], provides additional information on the requirements of a secure identity technology that is to be used in industrial contexts.

Sometimes, MCUs implementing a trusted environment do so through the use of a trusted execution environment (TEE). A TEE is “a tamper resistant processing environment that runs on a separation kernel. It guarantees the authenticity of the executed code, the integrity of the runtime states and the confidentiality of its code, data and runtime states stored on a persistent memory” [15]. A TEE implements a trusted environment in a similar manner to that of a TPM but is usually achieved within the software of an enabled device. The goals of a TEE are to achieve “isolated execution, secure storage, remote attestation, secure provisioning and a trusted path” [23]. In alignment with the definition of trust given in Section IV, trust for a TEE can be categorised as a hybrid of static and dynamic trust; as certification of the TEE is required prior to its deployment, but trust is maintained by the separation kernel once the TEE is in operation [15]. At its core, a TEE provides secure booting, secure scheduling, inter-environment communication, secure storage and a trusted I/O path [15]. The most prevalent example for IIoT applications is ARM’s TrustZone, from which multiple TrustZone-based TEE derivatives are forming; many of which are being defined for smartphone use, such as Nokia’s ObC, NVida’s TLK, Trustonic’s <t-base and Microsoft’s TLR [15].

ARM’s TrustZone is a virtualization-based technology, which provides support for memory, Input/Output operations (I/O), and interrupt-virtualisation within the hardware of the TrustZone capable device [15]. Hardware and software resources are split between a ‘normal’ world and a ‘secure’ world with a 33rd processor bit used to indicate the current execution world for the processor [24]. The TrustZone API suite specifies how interactions between the non-secure and secure world are to occur [24]. MCUs utilising ARMv6 or later processors come already equipped with TrustZone; however, in its native form, TrustZone operation has been found lacking in its adherence to the real-time processing requirements seen within an IIoT network [24]. The authors in [24] have attempted to improve upon the technology by implementing the basis of the TEE architecture as a low-priority thread in a real-time operating system (RTOS). Called, IIoTEED, the TrustZone-

based solution was developed for implementation on IIoT edge devices, with the current incarnation targeting IIoT gateway devices [24]. The author found that, when compared to an unaltered RTOS, the modified operating system produced negligible performance overhead and determinism variance [24]. When evaluating the interrupt latency, the authors found that a worst-case scenario occurred when a fast interrupt request arrived, as a switch from the secure to non-secure world began executing. The handling of the interrupt had to wait for two world switches to occur, resulting in an interrupt latency of 8.11ms [24]. Security-wise, the authors found that IIoTEED preserved only partial system confidentiality through “TrustZone’s strong spatial isolation mechanisms” [24]. A lack of resource access authentication meant that the solution was subject to “man-in-the-middle attacks and the interception and manipulation of messages transferred through the communication channel” [24]. Integrity was found to be preserved only at boot time and not for data over time while a high level of availability was seen, owing to the temporal isolation guaranteed by an asymmetric scheduling policy and the use of privileged and unprivileged interrupt sources [24].

In general, despite TEEs being purported to be the ‘silver bullet’ solution for embedded security, there remain a number of flaws compromising the security provided by a TEE. One such flaw is the non-disclosure agreements which prevent security experts from conducting thorough testing of TEE solutions, thereby preventing checks for compliance with security standards. Others are the software exceptions, hardware exceptions, shared memory interfaces, peripherals, and TEE-specific calls which present vulnerabilities that may provide an opening to an attacker capable of code execution using kernel privileges [15].

2.3.3 Isolation

Isolation techniques can be used to shelter parts of an endpoint device to prevent the cascade of undesirable effects caused by the failure of other parts of the device [2]. In this manner, minimum operation of the device can be guaranteed even during the event of a malicious attack. Physical isolation techniques may also be used to provide security

separately from the endpoint device by employing the use of a separate device. One such example is the use of a dedicated security gateway for endpoint security. This technique is often employed for older, legacy systems where device firmware cannot be upgraded to provide security according to updated security policies owing to insufficient resources or a lack of legacy support in the new security firmware [2]. Isolation can be achieved through the use of the operating system to isolate business and operational processes from security processes (process isolation); or the use of boundaries as determined by hardware, software or a hybrid implementation (container isolation); or through a hypervisor configured to isolate each running instance on an endpoint device (virtual isolation) [2].

Isolation practices can be seen in some of the solutions already highlighted. HSMs, such as a TPM, provide physical isolation of security processes by implementing security functions on a separate, physical device. Security modes, such as those implemented by a TEE, provide a form of virtual isolation through the separation of security processes and security resources by making them unavailable to normal operations functioning outside of the secure world.

Currently, hypervisor and container-based technologies remain heavily focused on securing traditional ICT technologies and operating systems. Solutions for the IIoT are slowly emerging with implementations focusing on the development of container technologies for IoT cloud services or Linux-based embedded operating systems designed to support gateway functions.

2.3.4 Attestation

Assuring the integrity of the endpoint data is often achieved by using a digital signature. The signature key is to be protected in storage using an HRoT and the signing operation would be conducted in a trusted execution environment such as within a trusted platform module [2]. In using a digital signature, an endpoint device would be able to validate the integrity of firmware updates prior to installation and configuration, and log files could be signed to ensure their integrity for future use [2].

Attestation is another technique that is utilised towards the assurance of device integrity. The basis of attestation is that “the entity that is to be tested, called the *prover*, sends a status report of its current configuration to another party, called the *verifier*, to demonstrate that it is in a known and thus trustworthy state” [25], [26]. To provide attestation, a trusted third party needs to be provided, along with a mechanism to provide provable information fields that can be bound together with a digital signature, called an attest [27]. A variety of attestation methods have been previously used to provide trust and integrity within IIoT networks, with varying degrees of success and shortcomings.

Remote attestation schemes assume that the prover is provided with a trusted mechanism such as a TPM, with integrity measurements being taken and securely stored during the secure boot process [26]. When conducting the attestation, the verifier sends a request for the device configuration measurements. Then the prover retrieves and signs the measurements, through the use of a digital signature algorithm or a digital certificate from a trusted third party, before sending them to the verifier [26]. The verifier then verifies the signature and compares the measurements against expected measurements for that device configuration [26]. Various shortcomings have been seen with the remote attestation scheme when applied to an IoT/CPS configuration. Firstly, as it is best suited for single-provers settings, it is infeasible for the verifier to know every possible device configuration in the network, especially given large-scale IIoT deployments [26], [28]. Secondly, with IIoT endpoints that are being left largely unattended and in remote deployments, the assumption about no physical attacks occurring on the devices can no longer be considered valid [27].

Software-based attestation was typically targeted for the resource-constrained devices at the edge of a WSN. Differing from the HRoT-based remote attestation, software attestation uses challenge-response techniques which allow for the verifier to check the integrity of the prover’s memory contents against modification, relying on checking the computation time of the prover in responding to the attestation challenge as an indicator of whether the device has been compromised [25]. Traditionally, the technique is heavily reliant on the assumption that an attacker is not actively attacking the network during the attestation

period [25]. Again, previous implementations of software-based attestation focused on single-prover scenarios, making existing commercial attestation solutions unsuitable for use in WSN/IoT applications.

Seshadri *et al.* [29] developed a software attestation solution for embedded devices using the challenge-response protocol. The solution was designed such that a correct response to the attestation challenge sent could only be produced if the memory contents of the embedded device corresponded to the verifier's locally computed answer [29]. The verification procedure used to answer the challenge was to be "pre-programmed into the device memory or downloaded from the verifier prior to verification" [29]. The main problem identified by the authors with previous attestation protocols using MAC addresses was that an attacker could move the memory contents— from which the verification procedures are based – into an empty memory space on the device. The attacker could then compute the MAC attestation functions from these contents during the device verification, effectively subverting the attestation process [29]. The use of checksums on the memory contents as part of the SWATT process attempted to guard against the reply-type attack as the checksum process failed with a high probability when the memory contents of the device differed from the expected checksum [29].

When designing the SWATT attestation procedure, the authors assume that the verification procedure is either pre-loaded or can be downloaded onto the embedded device. They also assume that the verifier is aware of the specific architecture for the hardware device and of the memory contents of the contesting device [29]. To verify the device, the SWATT protocol requires "pseudo-random memory traversal, resistance to pre-computation and replay attacks, high probability of single-byte changes, small code sizes, efficient implementation and non-parallelization" [29]. The attacker is assumed to have full control of the compromised device without having modified its hardware. The protocol is designed so that, should the attacker insert an 'if' statement to circumvent the verification procedure, there is a detectable increase in the procedure run-time.

Having implemented a genuine and attacker version of SWATT on an 8-bit Harvard architecture with 16kB of program memory and 1kB of data memory, the authors show

that the time differences between the attacker and legitimate verification code increases as the number of memory locations included in the checksum procedures increased, making the detection of a compromised device easier [29]. The authors note, however, that the main vulnerability seen with the SWATT procedure is that the time at which memory verification is conducted may not be the time at which the device is used on the network and during the intermediate time interval, it is possible that an attacker could change the memory contents of the device [29].

The attestation scheme developed by Asokan *et al.* [25] for large-scale device swarms also rules out the possibility of physical attacks on the devices in the design of the attestation protocol; however, the authors propose various mitigation techniques, acknowledging that physical attack is not completely out of the realm of possibility. The Scalable Embedded Device Attestation (SEDA) protocol is required to be able to “remotely verify integrity as a whole” [25]; be more efficient than individual device attestation; not require the verifier to know the detailed configuration of the prover; support parallel or overlapping attestation instances; and be independent of the integrity measurement used by the network devices [25]. The protocol is implemented in two phases; one where devices are introduced into the device swarm, known as the offline phase; and another where attestation is performed, known as the online phase. The offline phase consists of device initialisation and registration and is only executed once [25]. The online phase is executed multiple times, servicing every attestation request made by the verifier. Each device in the swarm is attested as an individual, with an accumulation of the attestations being reported to the verifier [25]. Global session indicators are used for each attestation instance, which is then used to construct a spanning tree for the swarm. Each swarm device attests all its children in the spanning tree, accumulating the results reported by the children into a report, which is then sent along with the attestation report for itself to its parent device [25]. Attestation for the swarm is achieved when the verifier receives an attestation report from the root device in the spanning tree, generating an output bit 1 if attestation was successful and a 0 otherwise [25].

To test the efficiency of the attestation protocol, two implementations based on the 8 MHz Secure and Minimal Architecture for (establishing a dynamic) Root of Trust (SMART) [30] and 24 MHz TrustLite [31] architectures were considered. Communication overhead for the root device was found to be $48g + 176$ bytes per send operation and $20 + 56g$ bytes per receive operation where g represents the number of neighbours for the root device [25]. The memory costs for the network nodes were found to be $20g + 168$ bytes with g representing the number of neighbours for the node under consideration [25]. The runtime for the SMART architecture was found to be $56,900 + 256g$ ms for the initiating device and $96 + 256(g-1)$ ms for the other network devices [25]. On the TrustLite architecture, the runtime was found to be $347.2 + 4.4g$ ms for the initiating device and $0.6 + 4.4(g-1)$ ms for the other network devices [25]. The result, however, is not surprising as TrustLite, with more computing power available, was seen by the authors to run SEDA significantly faster than the SMART architecture [25]. The authors also found that energy consumption for SEDA grew linearly in relation to the number of device neighbours, allowing an even energy consumption across the swarm should each device have the same number of neighbours [25]. Energy consumption for the initiating device was, however, to be higher than the other swarm devices owing to the computational intensity of computing the attestation sign procedure [25]. SEDA's main shortcomings are in that it is incapable of attesting the swarm topologies. It merely reports the number of devices that have passed attestation and only considers "remote software-based attacks" [28]; however, the authors note that extensions of the protocol could be made to allow for the identification of compromised devices, the assignment of different attestation priorities, support for dynamic swarms with changing topologies and mitigation of Denial of Service type attacks [25].

Ibrahim, Sadeghi and Tsudik improved upon previously seen attestation protocols in [28] by proposing the Device Attestation Resilient to Physical Attacks (DARPA) scheme. The main assumption behind DARPA is that the network under consideration has been left mostly unattended since deployment and, as a result, is highly susceptible to physical attack. The protocol requires devices to broadcast a "secure heartbeat" [28] to its neighbour to prove its continued presence in the network. The heartbeat is then forwarded by the

neighbouring devices to their own neighbours along with their own heartbeat signals [28]. Each device collects, verifies and logs the heartbeats it receives, which are then collated by a verifier during the next attestation cycle, performing a collective attestation scheme similar to that implemented by SEDA [28]. DARPA considers three types of adversary scenarios for its adversary model: remote software compromise, physical capture and a hybrid device compromise [28]. The scope of attack is limited to the assumption that there is no omnipotent adversary, meaning that all but one of the network devices can be compromised, no non-invasive physical attacks are employed, and no DoS attacks are employed [28].

To protect against physical attacks, DARPA assumes that an adversary cannot compromise any device that they (the adversary) have not captured. As an uncaptured device's heartbeat signal cannot be forged and is tied to a particular time instance of the heartbeat protocol, and every device supposedly emits its own time-based heartbeat signal at set intervals while collecting the heartbeat signals of its neighbouring devices, within the next attestation time period, the signal log of at least one uncaptured device will be missing at least one other device's heartbeat signal for at least one heartbeat protocol instance [28]. This is owing to the fact that an adversary will not be able to extract secrets from a captured device in order to forge a missing heartbeat within the heartbeat attestation time period [28]. To protect against software attacks, the attestation protocol needs to be able to verify the software integrity of the entire network. To do this, it is assumed that each network device has, at a minimum, read-only memory (ROM) for the storage of attestation code and cryptographic keys and a memory protection unit (MPU) [28]. The MPU is then responsible for ensuring that the integrity measurements in attestation code are immune to software attacks by limiting access to the cryptographic key to attestation code; ensuring that executing attestation code is not interrupted; and that registers and other temporary storage used by attestation code are flushed at the end of execution [28].

It can be seen that, in spite of the benefits achieved by the scheme, DARPA is still subject to various limitations, impractical assumptions and costly operations. The authors found that the use of a reliable clock was insufficient to protect against a hybrid adversary, thus

requiring the use of a reliable read-only clock or secure writable memory, accessible only by the ROM, on each device [28]. Other areas of improvement include the generation of false positives in the event of device failures and temporary network unreachability, a remaining lack of identification for compromised devices, and the relatively high overheads in heartbeat protocol as a result of heavy digital signing usage [28].

2.3.5 Cryptography

Cryptographic techniques are to be used on endpoint devices in order to establish and maintain integrity and confidentiality in the ICPS. Under the guidelines given in [2], IIoT endpoints should use standard cryptographic algorithms with regularly maintained and updated libraries [2]. The framework recommends the use of hardware RNG to ensure the randomness and uniqueness of cryptographic keys and a key revocation scheme should the invalidation of a key be required prior to its expiration [2].

In the past, performing cryptographic operations on IoT endpoint devices has been a continuous challenge owing to their resource-constrained nature. The sensor node configurations that are given in Table 2.7 highlight the resource limits that endpoint devices have had available for sensing, transmission, and processing purposes in previous years. More recently, endpoint devices are being fitted with 32-bit MCU processors, but the Random Access Memory (RAM) and Read-Only Memory (ROM) available on these devices are still much less than what can be found on a traditional personal computer (PC).

Table 2.7 Sensor network device configurations

Device Name	CPU	RAM	ROM/ Flash
TelosB [32]	16-bit MSP430	10kB	48kB
Mica2 [33]	8-bit Atmega128L	4kB	128kB
MicaZ [34]	8-bit Atmega128L	4kB	128kB
Ember EM2420 [35]	8-bit Atmega128L	4kB	128kB

Previous work ([36], [37], [38], [39]) has been done in attempting to use algorithms such as RC5 [40], RC6 [41], AES [42], DES [43], SkipJack [44], SHA-1 [45] and TEA [46] as mechanisms for securing sensor nodes. Energy consumption, memory utilisation and execution time were determined in the four studies using the Mica2 mote, which has the ATmega128L microprocessor, as the testing platform, with the exception of [36] which utilised the similar MicaZ mote.

Antonopoulos *et al.* [36] utilised AES, RC5 and SkipJack as their cryptographic algorithms of choice in order to determine the effect of security processes on the ATmega128L processor. Using Omnet++ 4.2 and the MiXiM framework for WSNs, network simulation was conducted in order to determine the execution time and energy consumption of the processor at the end of the key setup, packet encryption, and packet decryption phases [36]. Summation of these measured values then provided the total execution time and energy consumption for the cryptographic algorithms. In the study, the authors acknowledge that while the key setup phases of the algorithms could be optimised to give a better result for the algorithm performance, such techniques were found to impose increased memory requirements on the sensor node and potentially compromised the security level provided by the algorithms [36].

The study conducted by Chang, Meftic and Nagel [37] determined the energy consumption of RC5, DES with cipher block chaining (CBC), AES and SHA1 and operation times for only DES-CBC on the Mica2 and Ember motes. Owing to the size of the available memory resources, the size of the algorithm code and the algorithm's use of system resources, the authors were unable to load the cryptographic algorithms directly into the microprocessor. Instead, a divide and conquer technique was utilised in order to re-use portions of the processor memory during the execution of the cryptographic algorithm [37]. Even with the use of the divide and conquer technique, AES was not capable of running on the EM2420 node as the compiler utilised too much of the processor ROM [37]. The energy consumption of the running algorithm was determined using the voltages measured over a shunt resistor circuit. The measured voltages were used to calculate the current from

Ohm's law, and the current results with the supply voltage of the node batteries were used to calculate the power consumed during the operation of the cryptographic algorithm [37].

Guimarães *et al.* [38] tested the energy utilisation and execution time of SkipJack, RC5, RC6, REA and DES. Measurement of the processor in active mode without the inclusion of security processes gave the authors a CPU current of 8mA and an operational voltage of 3V; which was used in combination with the energy equation $E = V \times I \times \Delta T$ in order to determine a control energy consumption of 0.4104 mJ for the processor [38]. After having determined the base energy consumption measurements for the processor prior to the inclusion of the cryptographic algorithms, the authors noted that the increase in energy consumption would be determined by the added execution and transmission time as a result of the cryptographic processes [38]. An oscilloscope was used to obtain the execution time interval of the algorithms by monitoring the logical change in a general purpose input/output pin (GPIO) connected to the ATmega128 processor [38].

Trad, Bahattab and Othman [39] analysed AES, RC5 and RC6 in terms of the energy consumption, operational time and memory occupation on the ATmega processor. As in [37] and [38], to calculate the energy consumption of the cryptographic algorithms, the authors measured the voltage drop across two resistors using an oscilloscope [39]. The execution times of the key setup, encryption and decryption processes were measured with repeated execution of the cryptographic processes used to generate an average, estimate operational time value for the total algorithm. Both measurements were then used, in conjunction with the PowerTOSSIM tool, to calculate the energy consumption of the processor [39].

Table 2.8 Energy consumption of cryptographic algorithms implemented on Mica nodes

AES						
<i>Source</i>	[36]		[37]		[39]	
Block Size	16B		16B		16B	
Energy (μ J)	191.3		339		191.3	
RC5						
<i>Source</i>	[36]	[37]	[37]	[39]	[37]	[38]
Block Size	8B	8B	16B	16B	32B	29B
Energy (μ J)	783.05	111	124	139.44	150	36
RC6						
<i>Source</i>	[38]		[39]		[39]	
Block Size	29B		16B		32B	
Energy (μ J)	258.72		189.4		203.96	
SkipJack						
<i>Source</i>	[36]		[36]		[38]	
Block Size	8B		32B Est.		29B	
Energy (μ J)	11.04		44.16		51.84	
DES						
<i>Source</i>		[37]		[38]		
Block Size		32B		29B		
Energy (μ J)		126		14,592		

Table 2.8 summarises the results of the obtained energy consumptions observed by the authors in the four (4) studies, Table 2.9 summarises the operating times observed in studies [36] and [39], for the implementations of the RC5, AES, RC6 and SkipJack algorithms, and [37], for the implementation of DES. The observed operation times included key setup, encryption, and decryption of the data payload on a Mica2 node. Table

2.10 summarises the observed memory occupations for RC5, RC6, SkipJack, DES and AES as recorded in studies [38] and [39]. In the instances that the relevant cryptographic algorithm was not tested by a study, the column has been greyed out.

From the cryptographic algorithms studied, SkipJack gives the lowest, average energy consumption, with $35.68\mu\text{J}$ observed over two studies, when implemented on the Mica2, followed by DES and RC5, with $126\mu\text{J}$ and $139,36\mu\text{J}$ observed over one and three studies respectively. It can also be observed that the energy consumptions observed over the four (4) studies are similar. Differences in the implementations of the algorithms, the testing techniques and environments, and the block size used for the algorithms could be the cause of some of the fluctuations seen in the observed energy consumptions, including those cases which provided outlier consumption results when compared to similar studies; however, without the detailed implementation and testing details for the experiments conducted, a comparison of techniques cannot be done. These outlier values, however, are still useful as boundary values for any future testing that may be conducted on this topic.

Using the results of these four (4) studies it is shown that under typical operating conditions, the Mica2 CPU could be expected to consume, as an average, over four different message lengths, $3.75\mu\text{J}$ of energy [37].

Table 2.9 Operating time requirements for cryptographic algorithms on a Mica2 node

RC5			
<i>Source</i>	[36]	[39]	[37]
Block size	8B	16B	
Operating Time (ms)	30.64	6.81	
AES			
Block size	16B		
Operating Time (ms)	7.49	7.54	
RC6			
Block size		16B	
Operating Time (ms)		14.78	
SkipJack			
Block size	8B		
Operating Time (ms)	0.44		
DES			
Block size			32B
Operating Time (ms)			6.30

SkipJack once again prevailed with the fastest operating time of 0.44ms. This is due to the algorithm not having to perform key setup operations. DES was seen to have the second best operating time at 6.30ms, the fastest of the algorithms, which includes a key setup phase, followed by AES at an average of 7.52ms observed over two studies. RC5 and RC6 were observed as the slowest performing algorithms with averages of 18.73ms and 14.78ms respectively.

Table 2.10 Memory occupation for cryptographic algorithms on Mica2 node

RC5		
<i>Source</i>	[38]	[39]
Block Size	29B	32B
ROM (kB)	18.5	3
RAM (kB)	0.86	0.15
% Occupied (ROM)	14.45	2.34
% Occupied (RAM)	21.5	3.75
RC6		
Block Size	29B	32B
ROM (kB)	18.5	4.45
RAM (kB)	0.88	0.05
% Occupied (ROM)	14.45	2.82
% Occupied (RAM)	22	1.25
SkipJack		
Block Size	29B	
ROM (kB)	19.5	
RAM (kB)	0.69	
% Occupied (ROM)	15.23	
% Occupied (RAM)	17.25	

DES		
Block Size	29B	
ROM (kB)	30.5	
RAM (kB)	0.79	
% Occupied (ROM)	23.83	
% Occupied (RAM)	19.75	
AES		
Block Size		32B
ROM (kB)		9.00
RAM (kB)		2.14
% Occupied (ROM)		7.03
% Occupied (RAM)		53.5

The results from the studies observing the memory occupation of the cryptographic algorithms were interesting. On average, the percentages of ROM occupied by the cryptographic algorithms were very similar; with the percentage average for the algorithms using a 29-byte payload at 16.99% memory occupation and for the algorithms using a 32-byte payload at 4.063% memory occupation. RAM occupation was also relatively small for both payload sizes with the exception of AES, which occupied 53.5% of the RAM, more than half of the memory available to the device. This would lead to the slower processing of other device processes, compromising the performance of the sensor device.

In each of the four (4) studies, the authors noted that the application of cryptography did serve to compromise the performance of the sensor nodes. They further noted that the compromise in performance could worsen as the number of nodes observed during testing was to be increased to the scale of a typical sensor network deployment. Using the

averages calculated from the results given in the four studies, an estimation was made regarding the performance of the Mica2 when encrypting a packet generated using the IEEE 802.15.4 protocol.

Figure 2.2 shows the estimated energy consumption of the Mica2 node as a function of the time taken for a cryptographic algorithm to run to completion. The results for the plotted points are derived using the mean energy consumptions measured in [36] — [39] in order to estimate the operating times and energy consumptions that would be required to encrypt and decrypt packet size of 127 bytes. Figure 2.3 shows the estimated memory occupation, that would be required to encrypt and decrypt a packet size of 127 bytes.

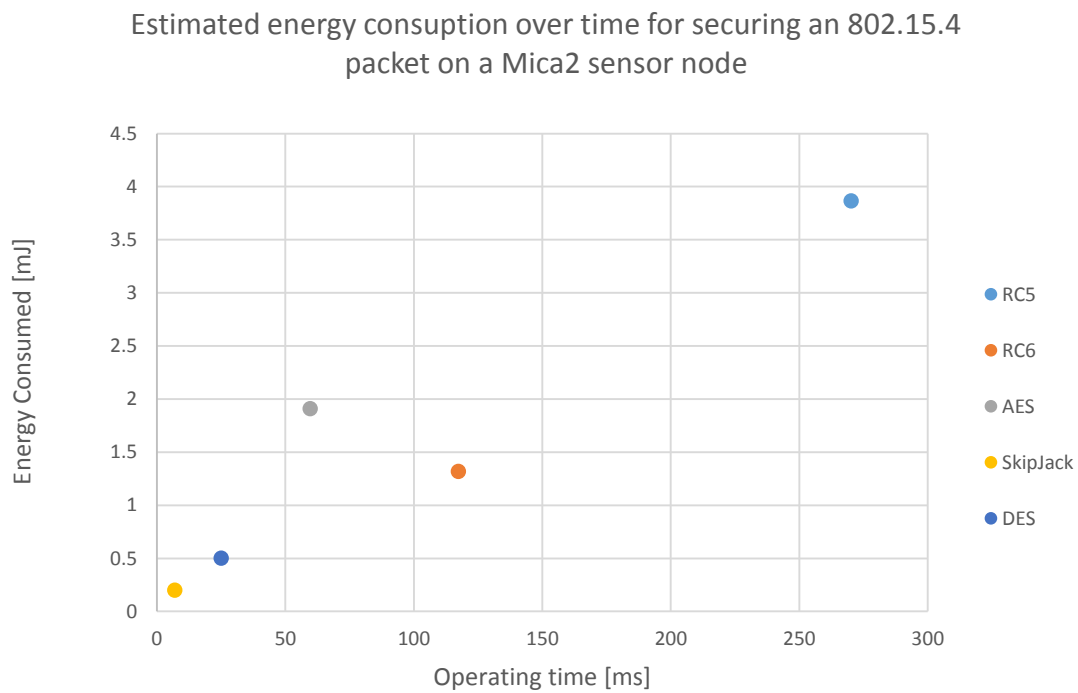


Figure 2.2 Energy consumption of Mica2 node as a function of operating time

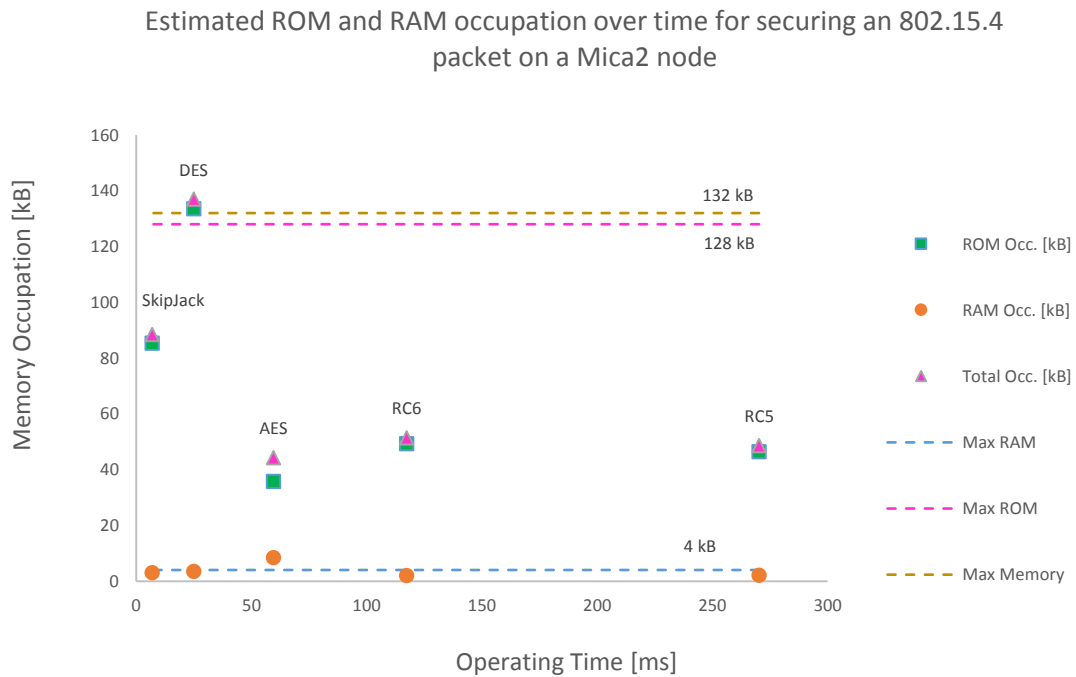


Figure 2.3 Memory occupation of cryptographic algorithms in a Mica2 node as a function of operating time

Looking at Figure 2.2, it could be seen that the longer a cryptographic algorithm ran on the Mica2, the more energy was consumed. For a 127-byte packet, SkipJack gave the shortest operating time along with the least energy consumption; followed by DES. AES and RC6 performed similarly, with AES operating in less time but consuming more energy and RC6 giving a longer operating time but consuming less energy. A large spike was then seen in the operating time of RC5, which gave the longest operating time, and consumed the most energy. This would make the algorithm a less suitable candidate for long-term, time sensitive network deployments.

Figure 2.3 shows the memory occupation of the cryptographic algorithms over the occupation time. Boundary lines are given in order to illustrate the maximum available memory resources on the Mica2's processor. Looking at the RAM scatter, one could see that the memory occupation estimated for AES exceeded the amount of RAM available on the Mica2 while the occupation estimated for DES exceeded the amount of ROM available. This means that the node would not be able to successfully encrypt and decrypt

the 127-byte packet owing to the depletion of resources. Considering the scatter again, the remaining algorithms occupied more than half of the available RAM. DES, as the second largest RAM consumer, occupied nearly all of the available RAM on the processor. Looking at the ROM scatter, one could see that, apart from DES, which depleted the available ROM resources, SkipJack was the largest ROM consumer, with AES consuming the least ROM.

As some of the algorithm occupation heavily favour either RAM or ROM memory, the total memory occupied was plotted in order to provide a better comparison point across the algorithms. As available ROM largely exceeds available RAM, the scatter for the total memory largely follows the ROM scatter pattern. From that, one could see that DES consumed the largest amount of the available memory resources, although it gave one of the shortest operating time of the four algorithms. RC5 and RC6 utilised similar amounts of the available memory resources however, RC6 gave a significantly faster performance than RC5. AES consumed the least amount of memory resources and had a relatively short operating time at near 60ms. This was still seen to be a relatively long delay that could cascade in larger network operations however, as its memory requirements exceed the available RAM in the processor, AES cannot be used to secure a 127-byte packet without additional RAM.

The results from the studies provide a good indication of the capability of older generation sensor nodes to handle cryptographic algorithms; however, a number of things can be noted which compromise the application of the conclusions drawn to the current IIoT sector. Of the six algorithms tested, four (4) may not be the most appropriate to use towards safeguarding an IIoT endpoint. SkipJack is an old cryptographic algorithm and was withdrawn from use by the NIST in 2016 [47]. DES is both an old and weak algorithm that was withdrawn in 2005 when it was superseded by AES [43]. RC6 is a non-standard algorithm, which was developed as a candidate for AES [48], while RC5, as a proprietary algorithm of RSA Data Security Laboratories, may not be as freely available as an open-source standard for public modification and use [40]. As a result, it may not be available

for public use or be an acceptable candidate for use in the industrial sector where the use of standardised, safety-tested equipment and software is required.

2.4 TRADE-OFFS IN ESTABLISHING SECURITY FOR THE IIOT

Security for an IIoT endpoint unfortunately comes at the cost of a variety of trade-offs in the overall endpoint design. These trade-offs may be in the form of the device performance, cost or in the device size. To be able to design a secure IIoT endpoint, it is important to identify where compromise will occur and to choose security solutions with a trade-off that does not impact the device's usefulness to the application for which it is intended. By considering the trade-offs given in line with the industrial standards of safety and security, a secured endpoint device can be designed in compliance with the different industry regulations. Designers are also able to choose solutions that are future-upgradable, preventing the need for physical redesigning as security solutions improve.

To that end, the trade-offs for the security solutions discussed in previous sections of this work are identified and discussed with a brief summary of the main points being highlighted in Table 2.11.

2.4.1 Physical Security

While it is vital that physical security measures be designed and included from the design stage of a secure mote, they come at a variety of costs, which also need to be factored into the design of the mote. External tamper sensing protections such as enclosure monitoring sensors and electromagnetic leakage shields will need to be provided with sufficient space and ventilation; leading to possible increases in enclosure sizes. Should the size increase not be constrained, situations will arise in which the enclosure size becomes a limitation in the application areas where the mote is used. Other considerations for external tamper sensing protection include the consideration of power sources for the sensing circuitry; the impact the additional drain may have on the lifetime of the mote's power source; and the design of maintenance accesses and strategies can that allow for upgrade work while not

introducing exploitable weak points that can be used by a malicious attacker. Physical security measures for the mote processor such as anti-tamper mesh and physical unclonable functions require careful design in order to disguise properly the signal and wiring patterns that are of interest to malicious attackers, so not to impact the performance of the processor. These measures also need to be implemented during the design phase of the MCU, making their inclusion on legacy devices expensive or very difficult to achieve.

2.4.2 Secure and Trusted Execution

The use of HSM to implement a root of trust brings with it a variety of trade-offs in terms of the power consumption and upgradability of the secure mote. The use of hardware security chips as a security device shortens the security lifetime of the secure mote. As the Trusted Computing Group (TCG) is continually working to upgrade the standard for TPM, one may find that the standard on the HSM employed to provide a RoT may be superseded by the newer standard sooner than expected, decreasing the level of trust that the secure mote provides. As the chips are hard-soldered into the mote design, they would be difficult to replace, and with large network deployments, such an operation would be highly expensive and infeasible. The use of a separate hardware module could also lead to an increase in the power consumption for the secure mote, both while active and while asleep. Testing would need to be conducted in order to determine the factor by which the added power drain reduces the effective lifetime of the mote's power source. Addition of a separate chip also serves to increase the possible PCB size for the mote in order to accommodate the chip and could introduce delay in the MCU start-up and processing times, as communication would now need to be routed through to the security module and back. Again, tests would need to be conducted to determine the added delay time and adjust the network operations to accommodate it within the application area requirements.

2.4.3 Isolation

The main problem facing the use of isolation techniques with the IIoT is the lack of appropriate solutions. Hypervisor use is still primarily within tradition ICT systems.

Although isolation is provided by within the ARM TrustZone TEE, the use of TrustZone is currently limited to ARM MCU solutions and even then, to ARM MCUs with an architecture that is TrustZone capable. Another trade-off with the use of TrustZone is that the solution cannot be independently tested by developers for security compliance as a result of a non-disclosure agreement. One is then limited to trusting the manufacturer's claims of compliance with security standards. The inability for independent compliance verification may also negate the use of the TrustZone solution in safety-critical applications where additional standards of safety may be required.

2.4.4 Attestation

As with isolation, the use of attestation in the IIoT lacks appropriate solutions that can be implemented in a secure mote design. Commercially available solutions for attestation remain primarily focused on single-prover methods, which are inappropriate for the peer-to-peer nature of an IIoT network deployment. Academic solutions for attestation attempt at designing multi-prover methods; however, these are still subject to a wide variety of shortcomings that are to be handled as future work and lack overall consensus on methodology, making the choice of a standard methodology with which to provide attestation most difficult. In addition, academic solutions would need to be taken regarding a lengthy, commercial development cycle in which verification and testing against industry standards would still be required.

2.4.5 Cryptography

With any cryptographic solution, a number of trade-offs will occur, starting with the need to ensure continuously that standard cryptography algorithms are in use and to check that deprecations have not occurred for implemented algorithms. As with the HSM, a hardware crypto accelerator would be difficult to upgrade in the event of the provided algorithm's deprecation. Additional care would also need to be taken to protect the communication paths between the MCU and the crypto accelerator to ensure that no security information is leaked. With the use of software cryptographic algorithm implementations, large increases

in memory occupation, large computation delays and increased power consumptions were observed when implemented on older generation devices. Although these observed performances may improve with the use of new generation IoT processors, software-implementations of cryptography are unsuitable for use on legacy devices. This would then either require a redeployment of the endpoint devices with newer, more future-proof solutions or result in a network with a mixture of secure and insecure devices, which fails to address adequately the security requirements of the network. In such instances, the use of a security gateway may be able to provide a cryptographic ability for communications originating from legacy, but these result in an increase in the overall network size. Also, care would need to be taken to adjust the network with appropriate routing protocols in order to prevent communication delays as a result of message queuing or instances of message dropping should multiple devices try to communicate with the gateway at once.

Table 2.11 Summary of security solutions and trade-offs

Existing Solution	Trade-Off
<i>Physical Security</i>	
Enclosure Monitoring Sensors	Increase in enclosure size to accommodate tamper sensors
Electromagnetic leakage shields	Increase in IIoT mote size to accommodate shields
Physical Unclonable Functions	Increased delay, decrease in available ROM and RAM
Anti-Tamper Mesh	Inclusion needed at design phase
	Careful pattern design needed
	Expensive/Difficult to include on legacy devices
<i>Secure and Trusted Execution</i>	
Hardware Security Modules	Increased device power requirements
	Not upgradable in future
	Increased PCB size to accommodate new integrated circuit
	Added delay to transmit encrypted data

Existing Solution	Trade-Off
<i>Isolation</i>	
TEEs and ARM TrustZone	Requires use of ARM MCU
	Untested for security compliance because of non-disclosure agreement
<i>Attestation</i>	
Commercial Solutions	Remain focussed on single-prover attestation
Academic Solutions	Still subject to a wide variety of shortcomings and lack of consensus on methodology
	Would still need to be verified and tested against industry standards
<i>Cryptography</i>	
Software Implementations	Large increase in memory occupation owing to large code sizes
	Long computation delays introduced into network
	Increased power consumption by endpoints
	Need to use standard cryptography algorithms and constantly check for algorithm deprecations
Hardware Crypto Accelerators	Difficult to upgrade if algorithm is deprecated
	Need to ensure protection of communication between accelerator and MCU

From the trade-offs given, a general design for a secure mote can be made. By considering the trade-offs given in line with the industrial standards of safety and security, a secured endpoint device can be designed in compliance with the different industry regulations. Designers are also able to choose solutions that are future-upgradable, preventing the need for physical redesigning as security solutions improve.

Although a good number of solutions have been identified to create an implementation of the IIC security framework, there are still gaps that have yet to be filled. Data loss

prevention techniques, hypervisor and container isolation techniques are still largely focused on implementations for the traditional ICT devices such as PCs, and existing solutions are highly unsuitable for use on an IIoT device. Device monitoring solutions such as intrusion detection are still largely academic with no commercial, standard solutions available for implementation on IIoT endpoints. Cryptographic functionality in secure MCUs still exists as primary hardware-based implementations, although vendors are very slowly beginning to offer cryptographic libraries for use on MCUs that do not have dedicated security hardware included. This serves to make those solutions more ideal for endpoint use as they are more easily upgradeable should cryptography algorithm standards change. Further work would need to be conducted to determine the impact that the use of vendor cryptographic libraries have on the MCU performance as compared to user-defined software implementation of cryptographic algorithms and secure MCUs with cryptographic functionality. The biggest gap prevalent is in the lack of open, standard security solutions for the IIoT. One example of this is in the available software TEEs for IoT devices, which is currently limited to ARM's TrustZone implementation. The use of vendor solutions means that configurations cannot be altered easily to suit design needs, or new solutions cannot be created based on existing design structures without a possible violation of the terms of use. It has also been previously identified that closed vendor solutions cannot be tested for compliance owing to non-disclosure agreements. This means that, even if a security device had been utilised on an IIoT endpoint, there is no guarantee that the level of security provided is at the level required for compliance with industrial regulations.

With the increasing concern towards securing the IoT, more MCUs and FPGAs are providing a basis of security from which additional changes can be made when designing a secure IIoT endpoint. Typically, this is achieved through the inclusion of a crypto engine and hardware accelerated cryptology within the MCU or open libraries of cryptographic standards, in addition to the inclusion of packaging and device sensors that provide for the detection of hardware tampering.

Table 2.12 Current secure MCUs for IIoT applications

Device Name	Manufacturer	CPU	Anti-Tamper	RoT	Digital Sign.	Data Integrity	HW Crypto.	Isolation	SW Crypto.
Smart Fusion 2 SoC FPGA [49]	Microsemi	32-bit ARM Cortex M3	Yes	Yes	?	Yes	Yes	?	Yes
A710x family [50]	NXP	MX51 security CPU	Yes	Yes	Yes	Yes	Yes	?	Yes
MAXQ 1852 [51]	Maxim	32-bit MAXQ 30 RISC Core	Yes	?	Yes	Yes	Yes	Yes	?
TM4C 12x [52]	TI	32-bit ARM® Cortex® -M4 with FPU	Yes	-	-	Yes	Yes	Yes	Yes
MAXQ 1050 [53]	Maxim	32-bit MAXQ 30 RISC Core	Yes	?	Yes	Yes	Yes	?	?
MAXQ 1061 [54]	Maxim	32-bit MAXQ 30 RISC Core	Yes	Yes	Yes	Yes	Yes	?	?
ST SAFE-A100 [55]	STM	Not Disclosed	?	?	Yes	Yes	Yes	?	Yes
ST SAFE-J100 [56]	STM	32-bit ARM SecurCore RISC	Yes	?	Yes	Yes	Yes	?	Yes

Zynq Ultra Scale MPSoC CG [57]	Xilinx	32-bit or 64-bit Dual-core ARM® Cortex A53 MPCore	-	Yes	Yes	Yes	Yes	Yes	?
AM335x (Sitara) [58]	TI	32-bit ARM Cortex-A8	-	Yes	-	Yes	Yes	Yes	-
PIC32 MZ-EF/EC [59]	Microchip	32-bit PIC	-	-	-	Yes	Yes	Yes	-

The symbols used in Table 2.12 serve to indicate the following; Yes = Security feature explicitly stated as being included by manufacturer, ? = Insufficient or contradictory information from manufacturer regarding the inclusion of this feature, - = Feature not included on device

Table 2.12 gives a brief list of MCUs that can be utilised in low power, IoT applications. These MCUs typically employ the use of 32-bit processors, an improvement on the 8-bit and 16-bit processors that had been used in the past. Those have had some combinations of security measures added as an effort to provide more secure device operation, such as cryptography, secure key storage, tamper resistance, tamper monitoring, authentication through hashing and, in the cases of the TrustZone-enabled ARM devices, a trusted environment. The implementations are done to minimise the overall delay and power consumption of the resultant IIoT endpoint. In some of the devices, not all the features identified for device security are implemented and in others, where it is unclear in the device datasheet whether a feature has been implemented on the device, a question mark is used as an indicator for that security feature.

Although a good number of solutions have been identified to create an implementation of the IIC and OpenFog security frameworks, there are still gaps which have yet to be filled. Looking at Table 2.12, with MCUs such as the NXP A710x [50], the Maxim MAXQ1852 [51] and hybrid implementations such as Microsemi's SmartFusion2 [49], one can see that

manufacturers are capable of providing many of the required security features; however, none of the secure MCUs have provided all the features required for a secure mote, highlighting the need for multi-layer security solutions. Some techniques require design and manufacturing in different engineering sectors from MCU design, such as PCB board and enclosure design for the inclusion of external tamper detection, and some features, such as data loss prevention techniques, hypervisor and container isolation techniques, remain largely focused on implementations for the traditional ICT devices such as PCs. As a result, existing solutions for these features are highly unsuitable for use on an IIoT device, and in-depth design and development is needed in order to push for the inclusion of these security features within the IoT. This requires increased collaboration across various fields in engineering and computer science along with increased collaborative development efforts between academia, as well as private and public sectors. Also linking to the increasing need for collaboration, device monitoring solutions for the IIoT, such as intrusion detection, are still largely academic and a larger push needs to be made towards the development and verification of usable, commercial, standard solutions based on research efforts already concluded. Cryptographic functionality in secure MCUs still exists as primary hardware-based implementations although vendors are very slowly beginning to offer cryptographic libraries for use on MCUs that do not have dedicated security hardware included. The software libraries offered by STMicroelectronics [60] and Texas Instruments [61] are two such examples. The release of verified, software cryptographic libraries serves to make the inclusion of encryption and decryption services easier and more ideal for endpoint use; as software has been seen to be more easily implemented and upgradable than hardware cryptography chip implementations in cases where algorithm standards change, extending the lifetime of network deployments. Further work would need to be conducted to determine the impact that the use of vendor cryptographic libraries have on the MCU performance as compared to a user-defined software implementation of cryptographic algorithms and secure MCUs with cryptographic functionality.

The biggest challenge facing the IIoT at the moment is in the lack of open, standard security solutions for the IIoT. One example of this is in the available software TEEs for IoT devices, which is currently limited to ARM's TrustZone implementation. The use of

vendor solutions means that configurations cannot be altered easily to suit design needs, or new solutions cannot be created based on existing design structures without a possible violation of the terms of use. It has also been previously identified that closed vendor solutions cannot be tested for compliance owing to non-disclosure agreements. This means that, even if a security device had been utilised on an IIoT endpoint, there is no guarantee that the level of security provided is at the level required for compliance with industrial regulations. Another instance in the limitations imposed by non-disclosure agreements and the lack of standard security offerings can be seen by considering Table 2.12 once more. Ambiguity is created in the true feature offerings for some of the MCUs, indicated with a question mark, where the information provided by manufacturers is vague or cannot be confirmed owing to provisions of non-disclosure. This hampers the ability for one to compare adequately solution offerings between manufacturers in order to select the device most appropriate for an application.

While there is some work currently being done to remedy the lack of standards for the IIoT, seen in the recent security frameworks developed by the IIC [2] and the OpenFog Consortium [8] and the standard for TPMs developed by the Trusted Computing Group [17], more standards need to be developed as a preventative and protective strategy for the IIoT sphere; allowing for accountability and verifiability to be placed on manufacturer designs and preventing a situation in which the IIoT could end up being flooded with designs that, ultimately, are incompatible with others and serve to open the already existing security holes further.

2.5 CHAPTER SUMMARY

In this chapter, a detailed introduction to the topic of security for the IIoT was presented, highlighting the characteristics of emerging and existing security frameworks and the previous work completed in trying to secure IIoT edge devices. Emphasis was placed on the specific security requirements proposed for devices at the edge of an IIoT network, as they are to form the basis of the design of a secure IIoT endpoint device.

In Section 2.2 an introduction to the open security architectures developed by the IIC and the OpenFog Consortium was given. The recommendations made by both organisations regarding the design of a secure IIoT endpoint device were highlighted.

In Section 2.3 a detailed look was taken into the recommendations made by the architectures; highlighting the technologies currently available for use in the IIoT sector and identifying areas in which further development is required.

In Section 2.4 the chapter was concluded and served to identify the trade-offs associated with the application of security on an IIoT endpoint node.

The following chapter serves to introduce the experiment design for this research, identifying the equipment chosen to conduct the testing and the experiment procedures used.

CHAPTER 3 METHODS

3.1 CHAPTER OBJECTIVES

This chapter serves to introduce the experimental setup and methodology used towards answering the research questions posed at the beginning of this work. The chapter gives a detailed insight into the experiment aims, equipment, tools, setup and methodology used during the testing of the identified cryptographic algorithms.

In Section 3.2 the aim of the experiments conducted in this research is given.

In Section 3.3 the cryptographic algorithms chosen for the experiments are identified with a brief motivation for their selection.

In Section 3.4 the equipment used to conduct the experiments is given along a brief motivation for their selection.

In Section 3.5 the experiment methodology is presented, detailing the steps taken towards determining the execution time, memory occupation and power consumption of each cryptographic algorithm running on the selected evaluation boards.

Section 3.6 provides a brief summary of the main topics covered and serves to conclude the chapter.

3.2 AIM

To date, studies regarding cryptography in the Internet of Things have been primarily focused on older generation platforms such as the Atmega128L. This has resulted in the prevailing view that IoT processors are incapable of supporting cryptographic operations in

software owing to the additional operational strain that cryptographic algorithms put on already resource-constrained devices. Over the last decade, however, processors for the IoT have progressed and expanded their resource offerings; subsequently moving from 8-bit and 16-bit processor architectures to 32-bit and 64-bit processor architectures. The aim of this experiment is to determine the performance of a new generation IoT microprocessor series when loading and running software-implemented cryptographic algorithms capable of encryption, decryption and key generation using pseudo random number generation.

The experiment considers three (3) main metrics:

- Algorithm Execution Time: defined as the time taken for key generation (if utilised by the algorithm), encryption, and decryption.
- Algorithm Power Consumption: defined as the power utilised by the MCU while executing the cryptographic algorithm.
- Memory Occupation: defined as the amount of MCU Flash and RAM memory utilised by the cryptographic implementation.

In the following sections, the equipment, tools and methodology used over the duration of the experiment are described.

3.3 CRYPTOGRAPHIC ALGORITHMS

The OpenFog security reference architecture [8] defines a list of cryptographic algorithms that are to be incorporated into the design of a secure mote for the IIoT. For the purposes of this research, three were chosen from the extensive list: the Advanced Encryption Standard (AES) as the symmetric algorithm, the Elliptic Curve Digital Signature Algorithm (ECDSA) as the public key algorithm, and the Secure Hashing Algorithm (SHA) as the hash generation algorithm. As published standards, the full implementation details of the algorithms are readily available from the NIST as FIPS PUB 197 for AES [42], FIPS PUB 186-4 for ECDSA [62], and FIPS PUB 180-4 for SHA [45]. As the chosen algorithms are also often to be found implemented as hardware accelerated chips, this allows for performance comparisons between the software and hardware implementations to be made in future studies.

From the chosen algorithms, a variety of key length and operation modes can be chosen, some of which are more appropriate for use in the IIoT than others, and some of which are purported to be more secure than others. The smallest secure key lengths for AES and SHA, as identified by the OpenFog architecture, were chosen in order to demonstrate the performance capabilities of the IoT processors as they implemented the minimum requirement for node security. This resulted in the use of AES128 and SHA256 for this research. In future studies, longer key expansions will be tested in order to determine the longest length supportable by the processors without significant impact on their processing resources. For AES, Counter Mode (CTR) was chosen for its simplicity, proven security and overall efficiency when running on software and hardware.

3.4 EQUIPMENT AND TOOLS

A wide variety of hardware and software tools were used over the course of this research. This section gives a brief description of the equipment used and the motivation behind their selection towards answering the research question proposed.

3.4.1 The ARM Cortex-M Processor Family

As some of the foremost modern processor series for the Internet of Things, the ARM Cortex-M family were the processors of choice for the testing cryptographic algorithms. The family currently consists of seven processors– M0, M0+, M3, M4, M7, M23 and M33– although the most recent processors, the M23 and M33, have yet to be available on development or prototyping boards. Within the processor family, M0, M0+ and M23 are intended for applications which require minimal monetary cost, power, and area [63]; the M3, M4, and M33 are intended as mid-range choices, balancing performance and energy efficiency [63]; and the M7 is intended to provide maximum performance, making it ideal for high processing applications [63].

A variety of ARM architectures have been implemented across the M-series. The M0+ and M3 have implemented the ARMv6 architecture; the M0, M4 and M7 have implemented

the ARMv7 architecture and the new M23 and M33 will implement ARM's latest ARMv8 architecture and will be ARM TrustZone capable; allowing for the establishment of a RoT and TEE in the processors [63]-[67].

As ARM does not physically produce silicon, there are a number of vendors from whom the processors can be purchased. As a result, variations in processing speed and available peripherals can occur. For the purposes of this research, the STM32 development boards were chosen after taking into consideration the availability of the boards in South Africa, cost, processing speeds, development tools and available peripherals on the evaluation boards. Figure 3.1 shows the STM32F0Discovery [68], STM32VLDiscovery [69], STM32F4Discovery [70] and STM32F767-Nucleo144 [71] evaluation boards used for this research, which are distributed with the M0 [64], M3 [65], M4 [66], [67] and M7 [67] processors. The processors chosen cover a wide spectrum of operating frequencies available in the processor series while two of the three architectures available across the series are represented. The operating frequency of the M3 distributed on the STM32VL comes close to the minimum available frequency at 24MHz [69] while the M7 distributed on the STM32767-Nucleo comes close to the maximum available frequency for the processor at 216MHz [71]. The processors were also able to represent the improvements made by ARM in the development of IoT processors in recent years, as the announcement dates for the chosen processors cover the ten (10) year period between 2004 and 2014. Figure 3.2 gives an illustration of the development timeline and the respective clocking frequencies of the STM32 development board processors as compared to the known general maximum operating frequencies for the specific M-series processor [72].

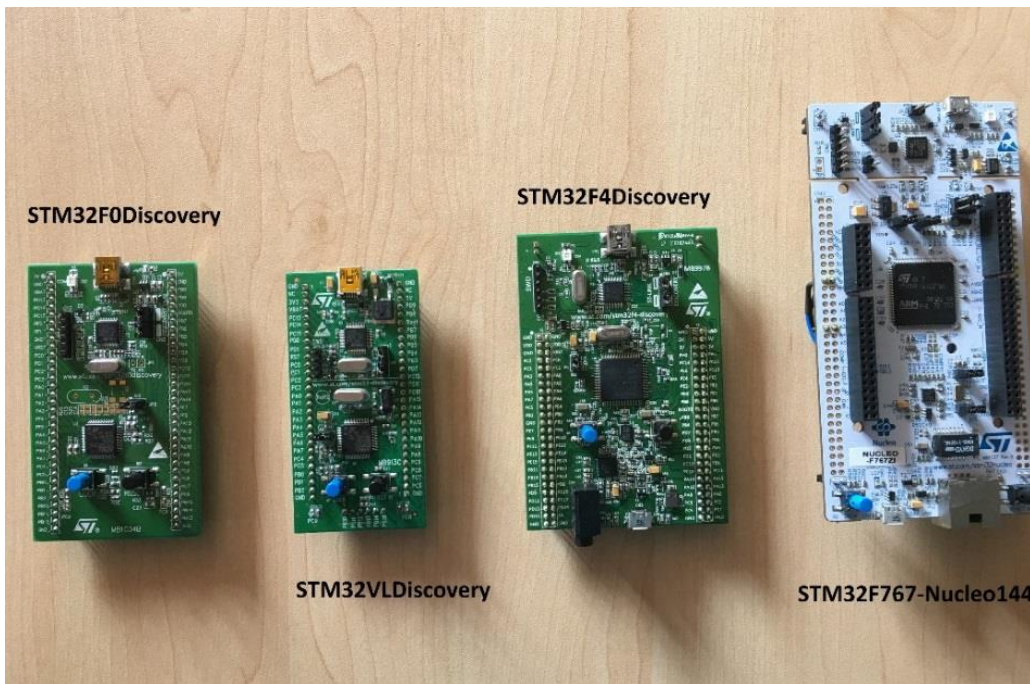


Figure 3.1 STM32 Development boards

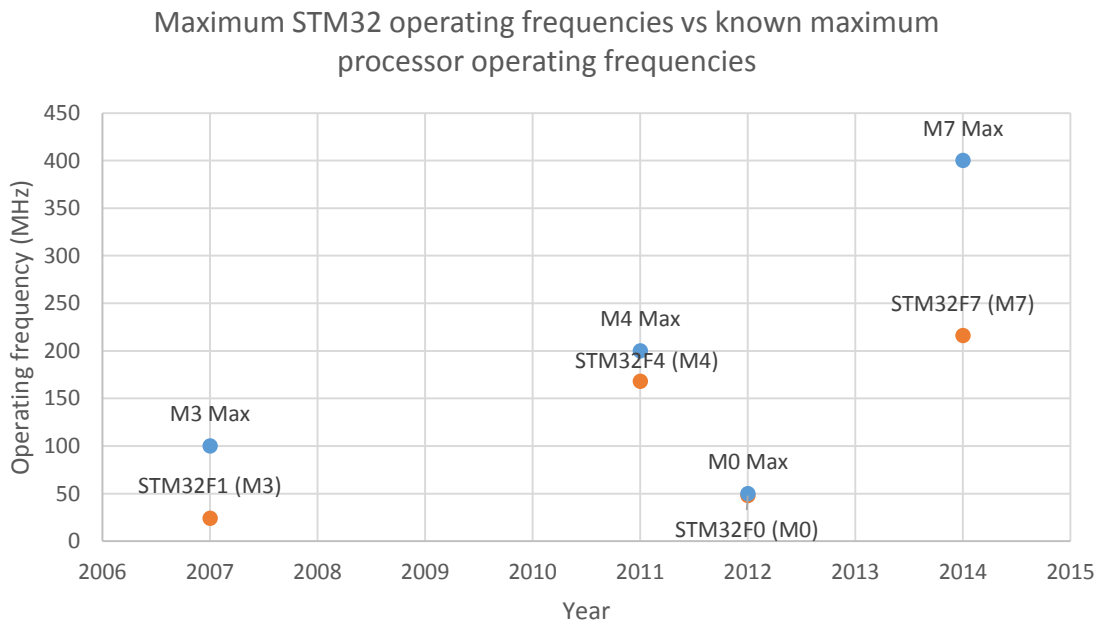


Figure 3.2 Release timeline and operating frequencies of Cortex-M series processors

3.4.2 STM CubeMX and X-Cube Cryptographic library

STMicroelectronics has provided a variety of software tools to aid in the faster development of projects for their solution chains. One such tool is the CubeMX graphical microcontroller configuration tool, seen in Figure 3.3.

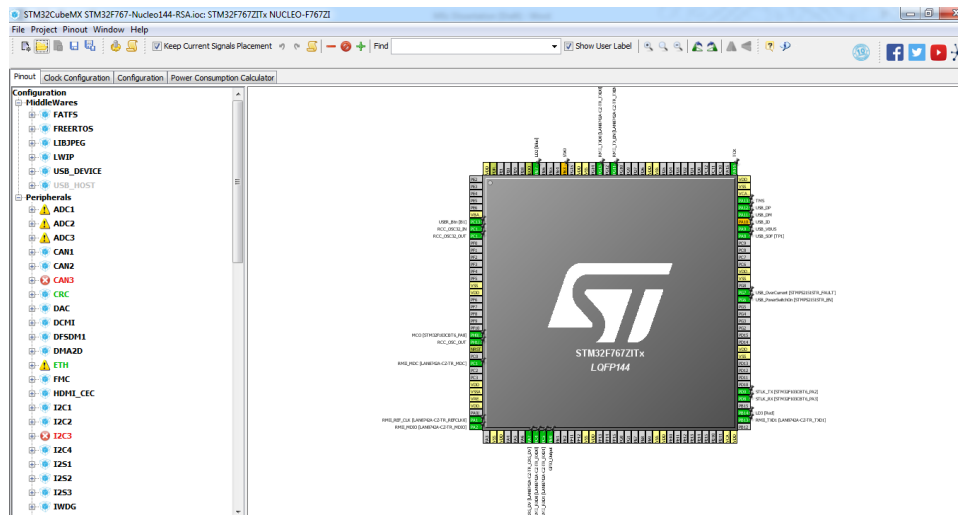


Figure 3.3 STM CubeMX graphic configuration for the STM32F767-Nucleo144

CubeMX provides a graphic interface that developers can use in order to select clocks, GPIO pins, peripherals and startup configurations easily for their MCU [73]. Conflict resolution for pin-outs, clock configuration validation, and power estimations are built within the configurator, helping to diminish the development time spent debugging incorrect user microcontroller configurations [73]. Once the developer is satisfied with the MCU configuration, the tool is then capable of generating the C initialisation code according to the specifications of one's preferred development environment, implementing the use of hardware abstraction layers to improve code portability [73]. CubeMX also allows for easier installation of various firmware libraries, leading to a shortened development lifecycle.

One such firmware library is the cryptographic library X-CUBE-CRYPTOLIB used over the course of this research. The library implements a variety of standard and non-standard cryptographic algorithms. Some of the algorithms within the library, which in this research

was AES 128-CTR, SHA-256 and ECDSA, have been certified for industrial use by the NIST Cryptographic Algorithm Validation Program [60]. The program provides independent, accredited “validation testing of FIPS approved and NIST recommended cryptographic algorithms” [74] with the aim of providing assurance to various agencies and industrial sectors that the cryptographic algorithms validated have been implemented correctly according to the official standards and of providing confidence that the algorithms do provide their claimed level of security [74]. The certification of the algorithms tested was the main decider in the use of the cryptographic library as opposed to a general C implementation of the algorithms as the code tested can be implemented directly in an industrial application without the need to go through an independent validation and certification process. The performance, memory occupation and consumption results seen within this research may be used as a guideline for product estimations to be used in industrial applications given that the algorithms tested have been pre-approved for use.

The X-Cube library is supported by the entire Cortex-M series; with implementations given in pure software and with support for hardware accelerated MCUs. Runnable implementations for specific MCU models from the series are provided for IAR Embedded Workbench, Keil and TrueStudio with templates available for easy porting to the other MCU models in the Cortex-M series. A drawback seen in the use of the cryptographic library with CubeMX is the current incompatibility of the library and the configuration tool. As at the writing of this dissertation, CubeMX was unable to load the X-CUBE-CRYPTOLIB firmware directly into the MCU configuration. A simple workaround, which is fully documented in [75], did, however, allow for the use of CubeMX and the cryptographic library.

3.4.3 Atollic TrueStudio

Atollic TrueStudio is an Eclipse-based integrated development environment (IDE) focused on the development of ARM-based projects [76]. The IDE provides a GCC compiler, C/C++ assembler and linker with support for debug probes such as ST-Link, SEGGER and

P&E micro [76]. Other features include support for integrated bug tracking, code editing, project management and version control tools [76]. The IDE is available in a Lite version, which is free to download, and a Pro version, which adds many additional features to the Lite version including a build/memory analyser, stack analyser and multi-core debugging [76]. For this research, the inbuilt ST-Link debugger, and the build/memory analyser were used, respectively, to enable MCU development and the determination of the memory occupation for each cryptographic algorithm.

3.5 EXPERIMENT METHODOLOGY

To determine the performance of the cryptographic algorithms on the Cortex-M processors, three (3) main metrics were considered: execution time, power consumption and memory occupation. Each experiment was repeated twenty (20) times in order to average out possible variations that may occur in single readings and to provide a more accurate estimation of execution time and power consumption. The following sections give a more detailed breakdown of the experiment setup followed for each metric.

3.5.1 Determining cryptography execution time

Execution time measurements were taken by toggling a chosen GPIO pin and using a Tektronix TDS3012B oscilloscope to measure the waveform width time between the falling and rising edges as the algorithm runs to completion. The physical experiment setup is seen in Figure 3.4 (a) and (b). The GPIO output pins of the blue LED were chosen for measurement in order to provide an additional visual indicator of successful execution. Table 3.1 gives a summary of the GPIO pin numbers that were used for timing as given for the different development boards.

Table 3.1 GPIO pins used for cryptographic algorithm timing

Development Board	Processor	GPIO Pin
STM32F0Discovery	Cortex M0	PC8
STM32VLDISCOVERY	Cortex M3	PC8
STM32F4DISCOVERY	Cortex M4	PD15
STM32F767Nucleo-144	Cortex M7	PB7



(a)



(b)

Figure 3.4 Execution time experiment setup

- (a) Oscilloscope setup shown using the STM32F0Discovery. (b) Close-up of probe connections.

Different pin toggling points were set within the cryptography code in order to determine the execution time for (a) the entire cryptographic algorithm, (b) the encryption portion of the algorithms, and (c) the decryption portion of the algorithm. During the default initialisation of the variables and methods used by the algorithm, the GPIO pin was set to high. Prior to the start of the cryptographic processes, the GPIO pin was pulled low. Once the algorithm had finished running, the GPIO pin was pulled high again. The subsequently square waveform was shown on the oscilloscope, which was set to trigger once the measured voltage on the probe surpassed 50% of the pin voltage. The width measurement

of the waveform, given by the oscilloscope, determined the execution time of the algorithm.

After having captured the waveform, the debugger was terminated and reset to ensure that the MCU was erased of the previous instance of the algorithm. The sequence of events was then repeated over twenty (20) runs, before changing the toggling points in order to measure the execution times of the next portions of the algorithm. The timestamps visible on each waveform were used to match the captured waveform, as well as the measured execution time, with its relevant run number. Figure 3.5 (a) - (c) gives the waveform captures seen for AES128-CTR when run on the STM32F0Discovery. These show the execution times measured during the first runs when toggling the pin for the entire algorithm, the encryption portion of the algorithm, and the decryption portion of the algorithm.

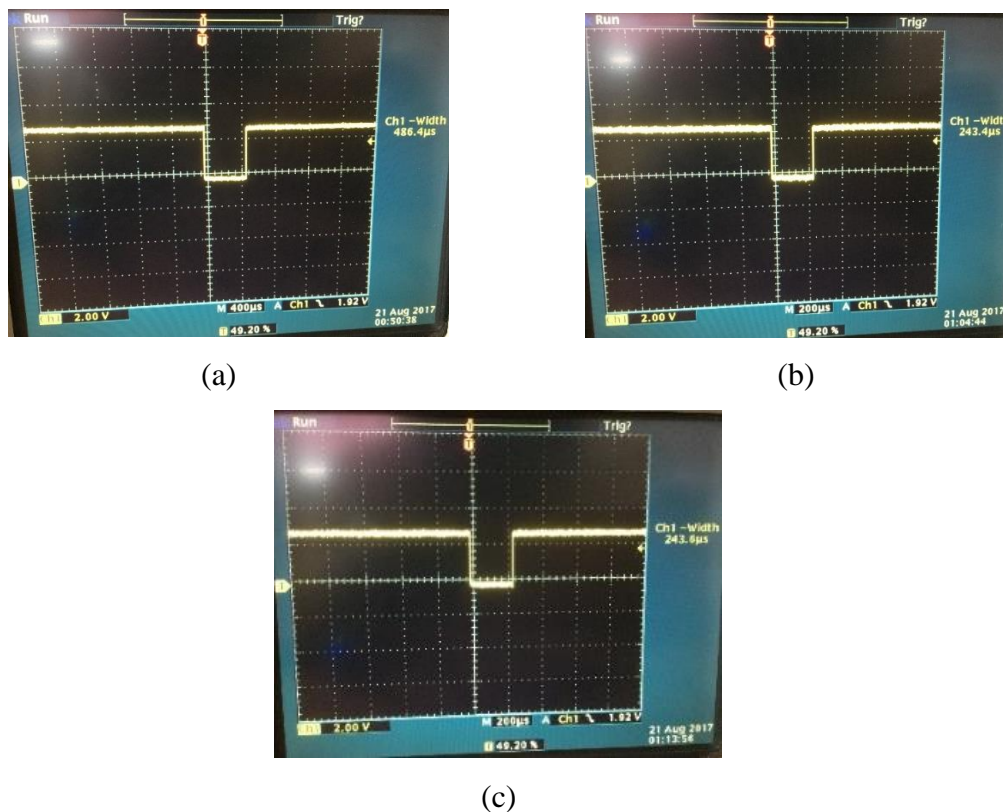


Figure 3.5 Execution time waveforms for a single run of AES128-CTR experiment on STM32F0Discovery

(a) Full algorithm, (b) Encryption only and (c) Decryption only

After concluding the necessary runs for the three algorithms under consideration, the boards were switched out and the test sequence started from the beginning using the new board.

3.5.2 Determining cryptography power consumption

To determine the power consumption of the cryptographic algorithms, the current consumption module (I_{DD}) found on the development boards was used in conjunction with a 1.2Ω shunt resistor in order to measure the voltage drop seen across the resistor during the execution of the cryptographic algorithms. With the resistance of the 1.2Ω resistor confirmed as 1.3Ω , when measured using a multi-meter; this was the resistance value used in the power calculations within this study. The measured voltage was then used in conjunction with Ohm's Law and the Power Equation to calculate the power consumption of the MCU seen when executing the identified algorithms.

The current consumption module is available on the Discovery and Nucleo boards and is activated by the removal of the I_{DD} jumper. Table 3.2 gives a summary of the different jumper numbers for I_{DD} across the different development boards, as given in the relevant user manuals.

Table 3.2 Summary of jumper pin numbers for the current consumption module

Development Board	Processor	I_{DD} Module Jumper No.
STM32F0Discovery	Cortex M0	JP2
STM32VLDISCOVERY	Cortex M3	JP1
STM32F4Discovery	Cortex M4	JP1
STM32F767Nucleo-144	Cortex M7	JP5

For the experiment, two channels were used with a Rigol DS1104Z oscilloscope. Channel one (1) was used to measure the voltage drop across the resistor while channel two (2) was used to visualise the execution time waveform generated when toggling the GPIO pins.

Having the two waveforms on-screen, the voltage drop waveform was matched with the execution time waveform, allowing for the identification of the portion of the MCU consumption that occurred as a result of the execution of the cryptographic algorithms. As only one power source may be used at a time with the development boards [70], power was provided to the boards using the USB connection, as opposed to an external power source, in order also to enable loading the cryptographic algorithms onto the MCUs through the debugger. The physical experiment setup for current measurement, following the measurement method described in [77], can be seen in Figure 3.6 (a) and (b).

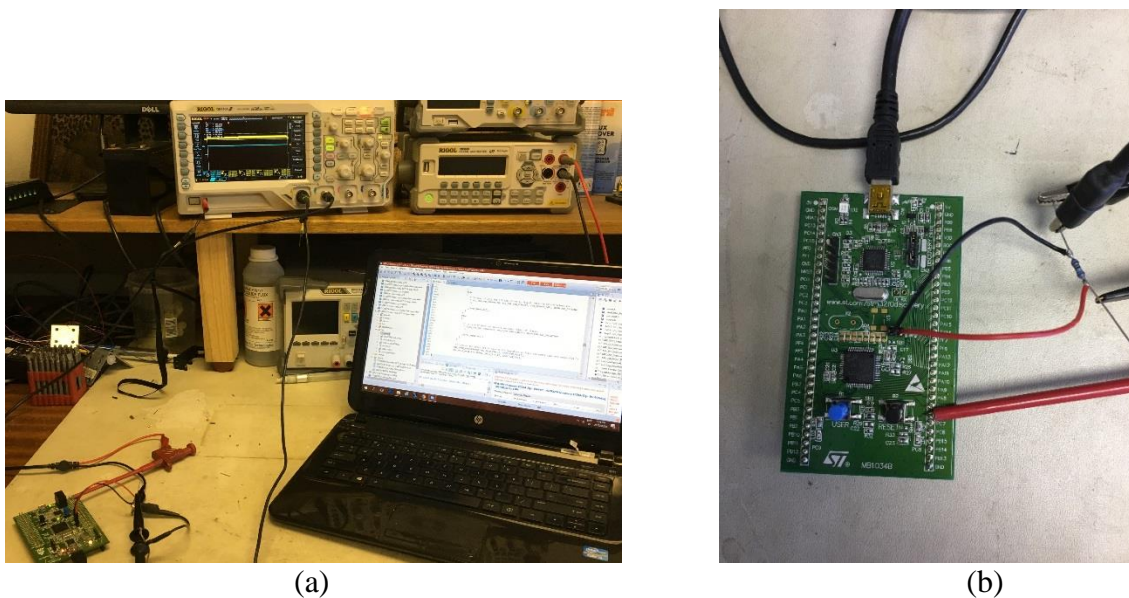


Figure 3.6 Physical setup of the power consumption experiment

(a) Entire setup and (b) Close-up of development board connections

As with the execution time experiments, the GPIO pin was toggled LOW during the instantiation of the variables and methods. Prior to the execution of the cryptography algorithm, the pin was toggled HIGH. Once the pin was pulled LOW again, the algorithm had successfully concluded. In order to isolate the voltage consumed during the execution of the cryptographic algorithms, measurement boundaries equal to the width of the execution time waveform were set using the cursor feature on the oscilloscope. The average voltage of the bounded portion of the waveform was then given by the

oscilloscope, which can be identified from the sample waveform in Figure 3.7 as the V_{PP} measurement.

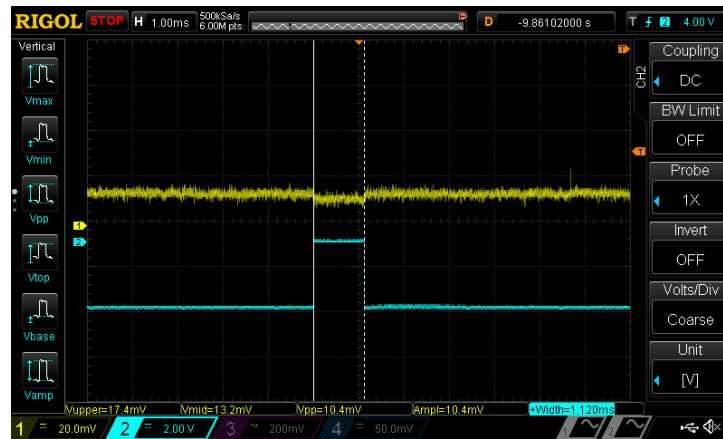
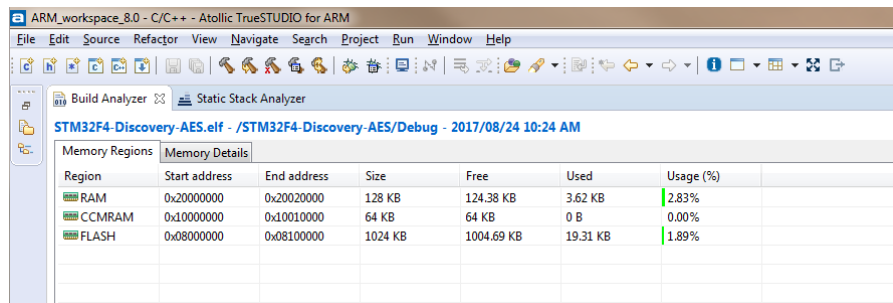


Figure 3.7 Execution time and bounded voltage waveform for SHA256 running on the STM32VLDISCOVERY

For this experiment, the power consumption was only measured for the whole cryptographic algorithm as a worst-case scenario. The experiment was again repeated twenty (20) times for each board, terminating and restarting the debugger to ensure the erasure of the MCU between runs.

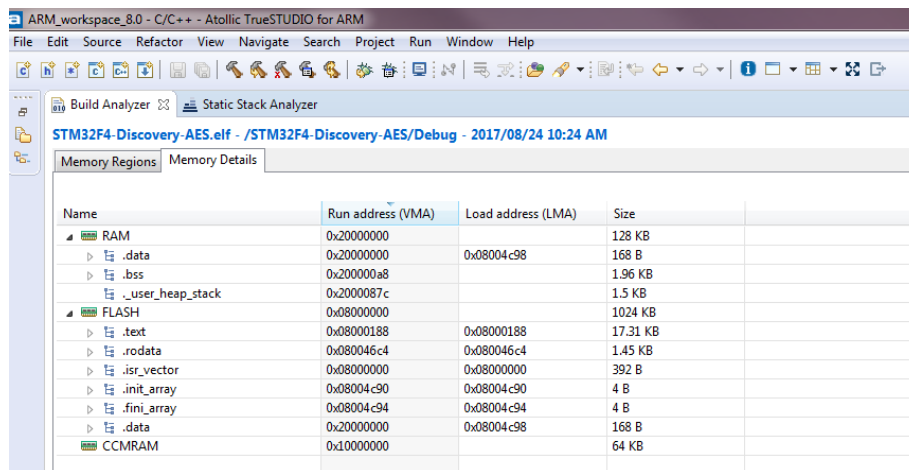
3.5.3 Determining memory occupation

The memory occupation of the cryptographic algorithms was determined using the Build/Memory analyser feature of Atollic TrueStudio Pro. The build/memory analyser utilised the elf file generated for the use of the debugger in order to give a detailed breakdown of the memory utilization for each algorithm build in RAM and Flash [78]. This can be seen in Figure 3.8 (a) and (b), which shows the memory occupation of AES128-CTR on the STM32F4DISCOVERY. The analyser can be switched also to view the stack utilisation of the algorithms; however, that was beyond the scope of this research.



Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20020000	128 KB	124.38 KB	3.62 KB	2.83%
CCMRAM	0x10000000	0x10010000	64 KB	64 KB	0 B	0.00%
FLASH	0x08000000	0x08100000	1024 KB	1004.69 KB	19.31 KB	1.89%

(a)



Name	Run address (VMA)	Load address (LMA)	Size
RAM	0x20000000		128 KB
. data	0x20000000	0x08004c98	168 B
. bss	0x200000a8		1.96 KB
. _user_heap_stack	0x2000087c		1.5 KB
FLASH	0x08000000		1024 KB
. text	0x08000188	0x08000188	17.31 KB
. rodata	0x080046c4	0x080046c4	1.45 KB
. isr_vector	0x08000000	0x08000000	392 B
. _init_array	0x08004c90	0x08004c90	4 B
. _fini_array	0x08004c94	0x08004c94	4 B
. data	0x20000000	0x08004c98	168 B
CCMRAM	0x10000000		64 KB

(b)

Figure 3.8 Memory occupation of AES128-CTR on the STM32F4Discovery

(a) Summary version (b) Detailed expansion version

While conducting the experiments to determine the memory occupation, it was discovered that a rebuild of the program had no effect on changing the memory occupation of the algorithms. The inclusion of the GPIO toggling instructions also had minimal effect on the resulting memory occupation of the algorithms.

3.6 CHAPTER SUMMARY

This chapter served to detail the hardware and software tools and experiment methodology used towards the completion of this research. The STMicroelectronics development boards were chosen in addition to the AES128-CTR, ECDSA and SHA256 algorithms from the

X-Cube cryptographic library. An oscilloscope was used in measuring the execution time and power consumption of the Cortex-M series MCUs, while the Atollic TrueStudio Build/Memory analyser was used to determine the memory occupation of the algorithms.

In Section 3.1 the chapter objectives and overview were presented.

In Section 3.2 the aim of the experiments conducted in this research were given.

In Section 3.3 the cryptographic algorithms chosen for the experiments were identified with a brief motivation for their selection.

In Section 3.4 the equipment used to conduct the experiments was given along a brief motivation for their selection.

In Section 3.5 the experiment methodology was presented, detailing the steps taken towards determining the execution time, memory occupation and power consumption of each cryptographic algorithm running on the selected evaluation boards.

The following chapter serves to report and give a brief analysis of the results from the experiments conducted in this chapter.

CHAPTER 4 RESULTS

4.1 CHAPTER OBJECTIVES

This chapter presents the graphical and statistical results of the execution performance, memory occupation and power consumption experiments conducted in Chapter 3. It begins to answer the research question proposed at the beginning of this study. The chapter is intended to serve as an introduction to the full discussion of the results presented in Chapter 5.

In Section 4.2 the results of the execution time experiments are given for AES128-CTR, ECDSA and SHA256; giving the mean execution time, standard deviation and standard mean error for the Cortex-M processors.

In Section 4.3 the results of the power consumption experiments are given for AES128-CTR, ECDSA and SHA256; giving the mean power consumption, standard deviation and standard mean error for the Cortex-M processors.

In Section 4.4 the results of the memory occupation experiments are given for AES128-CTR, ECDSA and SHA256; giving the RAM and Flash consumed by the algorithms on the Cortex-M processors.

Section 4.5 provides a brief summary of the main topics covered and serves to conclude the chapter.

4.2 EXECUTION PERFORMANCE

The experimental procedure calculating the execution time for each of the cryptographic algorithms across the Cortex-M family was presented in Section 3.5.1 of Chapter 3. Twenty (20) individual runs were conducted, with the previous instance of the program being erased from the MCU through the instantiation of a new debugger session between each run. The time taken to execute the entire algorithm was measured, with additional encryption and decryption execution times being measured as separate experiments for the symmetric algorithms, AES128-CTR and SHA256. The results of the experiments have been collated and presented in the following sections, rounded to three decimal points.

To determine the statistical error of the measured results over the number of runs N , the mean (\bar{X}), standard deviation (σ_X) and standard error of the mean ($\sigma_{\bar{X}}$) were determined using (4.1) to (4.3).

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N X_i \quad (4.1)$$

$$\sigma_X = \sqrt{\frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})^2} \quad (4.2)$$

$$\sigma_{\bar{X}} = \frac{1}{\sqrt{N}} \sigma_X \quad (4.3)$$

4.2.1 AES128-CTR

As a symmetric algorithm, three execution time measurements were taken during the AES128-CTR experiments: the execution time for the cryptographic algorithm as a whole, the time taken only for encryption operations, and the time taken only for decryption operations. Figure 4.1 gives the results of the execution of the full algorithm while Figures 4.2 and 4.3 give the results of the execution times for the encryption and decryption operations respectively.

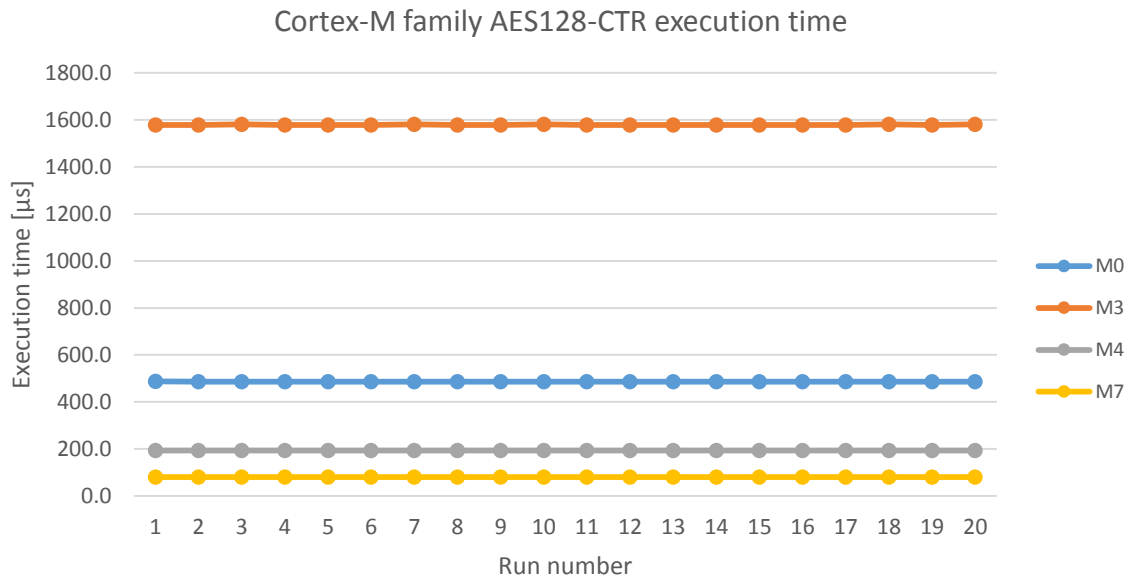


Figure 4.1 Execution time of Cortex-M series processors running AES128-CTR

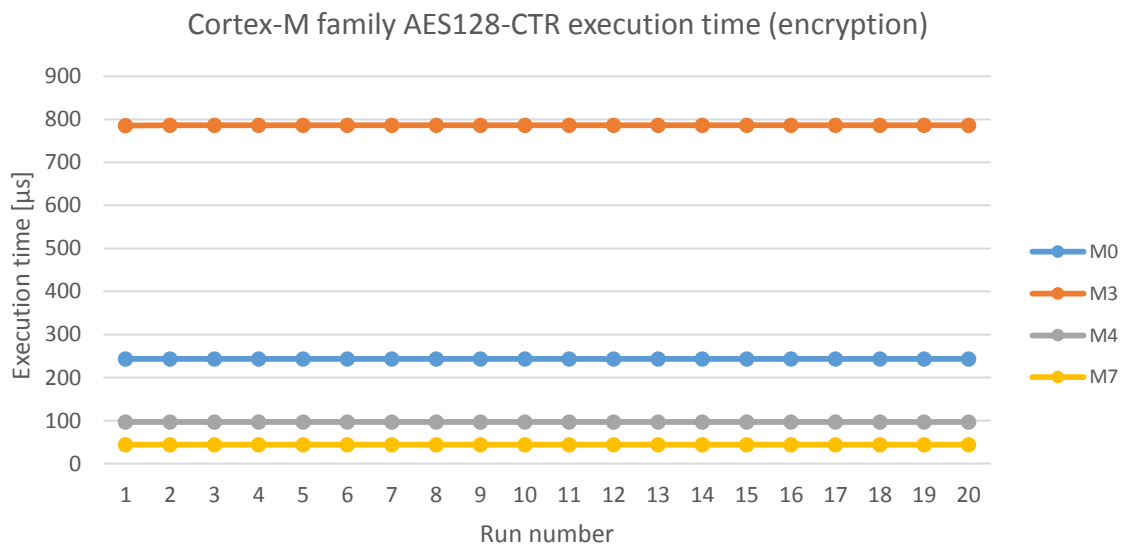


Figure 4.2 Execution time of Cortex-M series processors running AES128-CTR (encryption only)

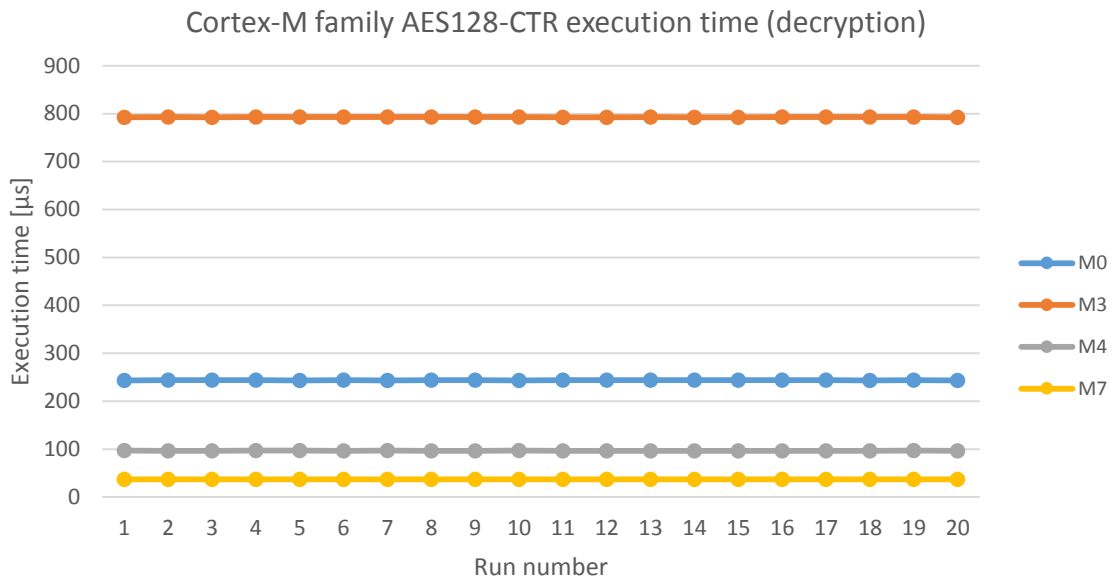


Figure 4.3 Execution time of Cortex-M series processors running AES128-CTR (decryption only)

Looking at the three figures, it could be seen that very little deviation occurred over the twenty (20) runs. With an operating frequency of 24 MHz, the M3 had the longest running times while the M7, with the largest operating frequency of 216 MHz, had the shortest running times. Surprisingly, the performance of the M4, with an operating frequency of 168 MHz, was close to the performance of the M7, which ran approximately 140.334 μ s faster than the M4 with a 22.222% speed increase over processor. Comparing the execution times seen in Figures 4.2 and 4.3, one could see that the time spent for encryption or decryption in AES128-CTR was nearly identical

To confirm mathematically the trends visible in the graphs, the results from the runs were used to calculate the mean, standard deviation and standard error for execution time estimates for each of the MCU, which are given in Tables 4.1 to 4.3:

Table 4.1 Execution Time Mean, Standard Deviation and Standard Error for MCUs running AES128-CTR

	\bar{X}	σ_x	$\sigma_{\bar{x}}$
	[μs]	[μs]	[μs]
M0	486.470	0.175	0.039
M3	1578.500	0.889	0.199
M4	193.870	0.208	0.047
M7	80.667	0.021	0.005

Using the result above, the total execution time for AES128-CTR on the Cortex-M processors was estimated as follows:

$$\text{M0: } 486.470 \mu\text{s} \pm 0.039 \mu\text{s}$$

$$\text{M3: } 1578.500 \mu\text{s} \pm 0.199 \mu\text{s}$$

$$\text{M4: } 193.870 \mu\text{s} \pm 0.047 \mu\text{s}$$

$$\text{M7: } 80.667 \mu\text{s} \pm 0.005 \mu\text{s}$$

Table 4.2 Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (encryption only)

	\bar{X}	σ_x	$\sigma_{\bar{x}}$
	[μs]	[μs]	[μs]
M0	243.385	0.075	0.017
M3	786.175	0.210	0.047
M4	96.984	0.068	0.015
M7	44.000	0.000	0.000

Similarly, using the results in Table 4.2, the execution time for the encryption operations could be estimated as:

M0: $243.385 \mu\text{s} \pm 0.017\mu\text{s}$

M3: $786.175 \mu\text{s} \pm 0.047 \mu\text{s}$

M4: $96.984 \mu\text{s} \pm 0.015 \mu\text{s}$

M7: $44.000 \mu\text{s} \pm 0.000 \mu\text{s}$

Table 4.3 Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (decryption only)

	\bar{X}	σ_X	$\sigma_{\bar{X}}$
	[μs]	[μs]	[μs]
M0	243.735	0.093	0.021
M3	792.670	0.187	0.042
M4	96.704	0.067	0.015
M7	37.071	0.014	0.003

The execution time for the decryption operations was estimated from the results in Table 4.3 as:

M0: $243.735 \mu\text{s} \pm 0.021\mu\text{s}$

M3: $792.670 \mu\text{s} \pm 0.042 \mu\text{s}$

M4: $96.704 \mu\text{s} \pm 0.015 \mu\text{s}$

M7: $37.071 \mu\text{s} \pm 0.003 \mu\text{s}$

From the estimations, it could be seen that, indeed, very little deviation occurred over the course of the execution time experiments and that equal periods of time were spent on the individual encryption and decryption operations; confirming the trends visible in the graphs. From the four (4) processors, the M3 presented the largest, observed deviation for the total execution time, encryption time and decryption time, while the M7 presented the least amount of deviation; with an instance of no deviation being observed over the twenty (20) runs measuring time spent on encryption operations.

It was also observed that, on three of the four (4) Cortex-M processors, AES128-CTR was capable of running within microseconds while the M3 ran AES128-CTR at an average

execution time of 1.578ms. Dependent on the allowed delay tolerance for a particular application, the observed execution times of AES128-CTR on the Cortex-M processors may be sufficiently fast for the algorithm to be considered for use, without introducing significant delay into the network, on IIoT nodes used in hard real-time operations.

4.2.2 ECDSA

The STM Cryptographic library implements two versions of ECDSA: one version that only implements signature generation and verification from pre-loaded public and private keys – identified within this study as ECDSA (Sign-Verify), and another version that utilises a pseudo random number generator (PRNG) for key generation – identified within this study as ECDSA (Key Gen- Sign-Verify). The execution times for both algorithms on the Cortex- M processors were determined over twenty (20) runs. Figure 4.4 gives the observed results of ECDSA without key generation while Figure 4.5 gives the results for ECDSA with key generation:

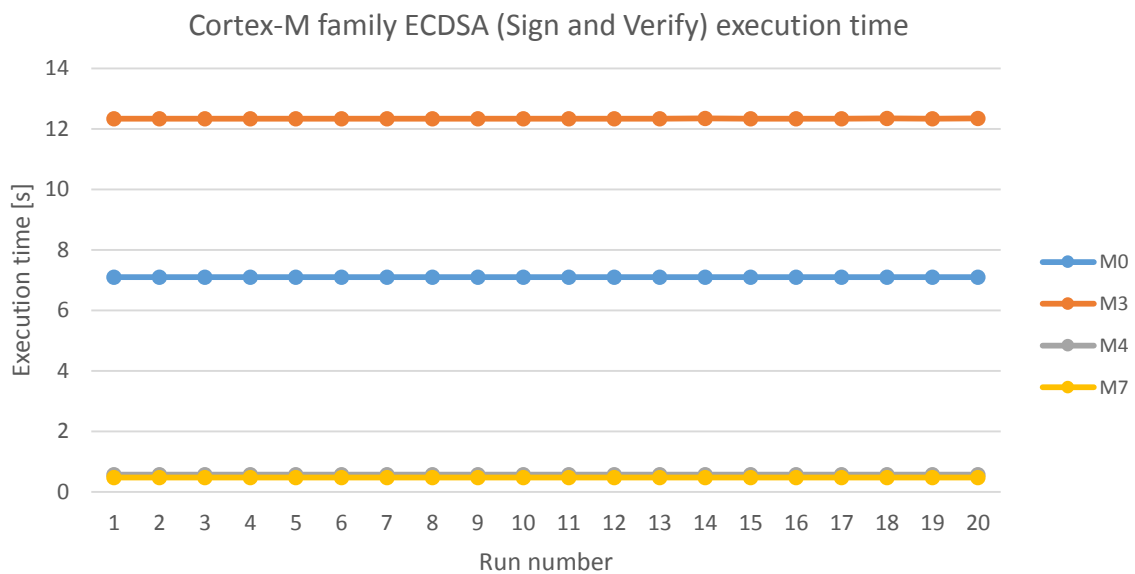


Figure 4.4 Execution time of Cortex-M series processors running ECDSA (Sign-Verify)

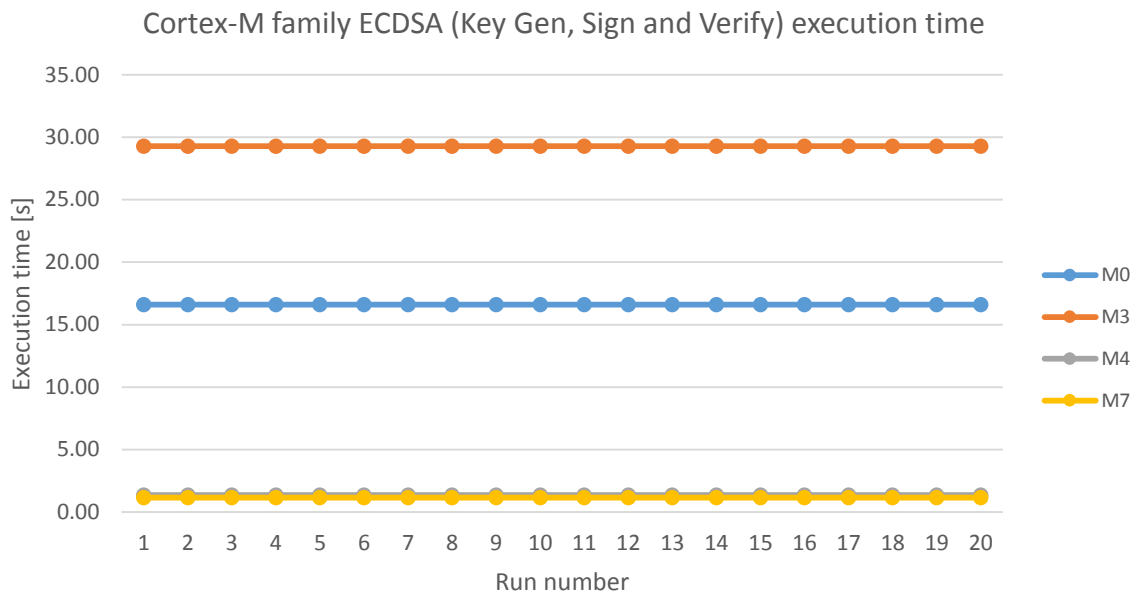


Figure 4.5 Execution time of Cortex-M series processors running ECDSA (Key Gen-Sign-Verify)

As with AES128-CTR, very little deviation was observed over the course of the execution time experiments. Interestingly, the performances of the M4 and M7 when running both versions of the public key algorithm were nearly identical. This was seen where the graph illustrating the performance of the M4 was almost completely obscured by the graph illustrating the performance of the M7. Also interesting to note was that the inclusion of key generation had, in essence, doubled the execution time of ECDSA on the processors as opposed to the use of pre-loaded keys. This resulted in delays that were seconds long; particularly in the case of the M3, where the execution time could add a delay that was nearly half a minute long.

Mathematically, the visible trends were confirmed using the results to calculate the mean, standard deviation and standard error for the execution time estimates for each of the MCU. These are given in Tables 4.4 to 4.7:

Table 4.4 Execution time mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify)

	\bar{X}	σ_x	$\sigma_{\bar{X}}$
	[s]	[s]	[s]
M0	7.101	0.002	0.000
M3	12.342	0.004	0.001
M4	0.572	0.000	0.000
M7	0.471	0.000	0.000

Using the mean and standard error the estimated execution time for ECDSA without key generation was given as follows:

$$\text{M0: } 7.101 \text{ s} \pm 0.000 \text{ s}$$

$$\text{M3: } 12.342 \text{ s} \pm 0.001 \text{ s}$$

$$\text{M4: } 0.572 \text{ s} \pm 0.000 \text{ s}$$

$$\text{M7: } 0.471 \text{ s} \pm 0.000 \text{ s}$$

Table 4.5 Execution time mean, standard deviation and standard error for MCUs running ECDSA (Key Gen- Sign-Verify)

	\bar{X}	σ_x	$\sigma_{\bar{X}}$
	[s]	[s]	[s]
M0	16.600	0.000	0.000
M3	29.284	0.005	0.001
M4	1.362	0.000	0.000
M7	1.141	0.000	0.000

Similarly, the estimated execution time for ECDSA including key generation was given as:

$$\text{M0: } 16.600 \text{ s} \pm 0.000 \text{ s}$$

$$\text{M3: } 29.284 \text{ s} \pm 0.001 \text{ s}$$

$$\text{M4: } 1.362 \text{ s} \pm 0.000 \text{ s}$$

$$\text{M7: } 1.141 \text{ s} \pm 0.000 \text{ s}$$

Apart from the very small deviation seen in the results for the M3, the estimated execution mean times for ECDSA on the Cortex-M series, both with and without key generation, show no deviation. This could allow for the employment of delay tolerance strategies using a pre-set delay estimate. However, with the longer estimated execution times, ECDSA may not be suited for hard real-time industrial applications. The use of the PRNG for key generation only served in nearly doubling the execution time of the algorithm. This observation did not provide much confidence in the performance of other public key cryptography algorithms or the inclusion of public key cryptography at the edge of the IIoT network. In this instance, should a public key algorithm need to be used on an edge node, the use of a hardware accelerator with true random number generation may be needed to be employed to avoid the introduction of long delays into the network.

4.2.3 SHA256

As in the experiment with AES128-CTR, three execution time measurements were taken during the SHA256 experiments: the execution time for the hashing algorithm as a whole, the time taken for only hash generation operations, and the time taken for operations confirming the validity of the message digest.

Figure 4.6 gives the results of the execution time for the full algorithm, while Figures 4.7 and 4.8 give the results of the execution time for hash generation and digest verification operations respectively.

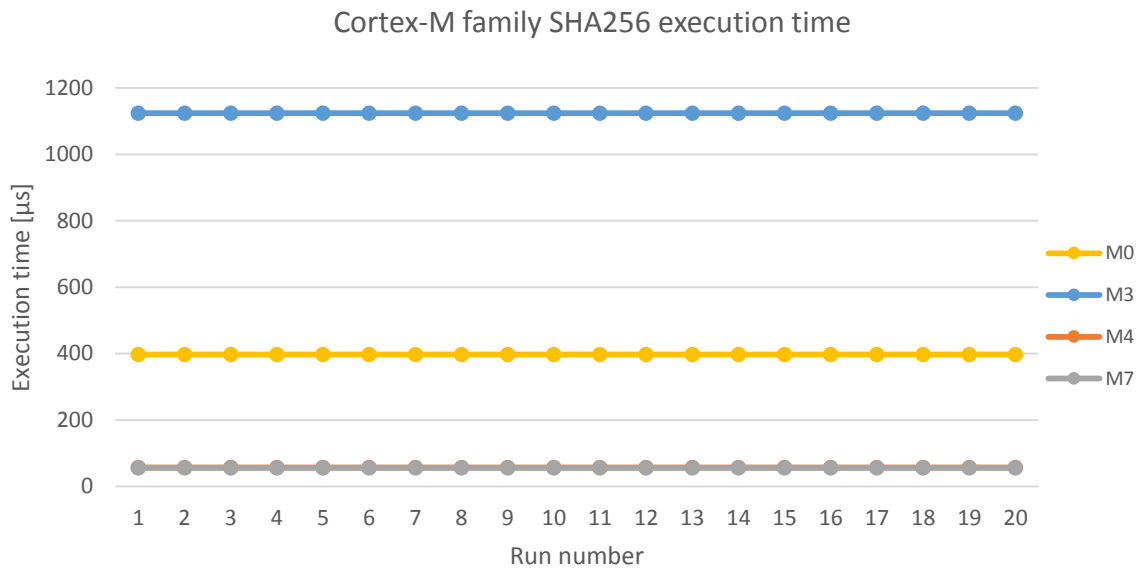


Figure 4.6 Execution time of Cortex-M series processors running SHA256

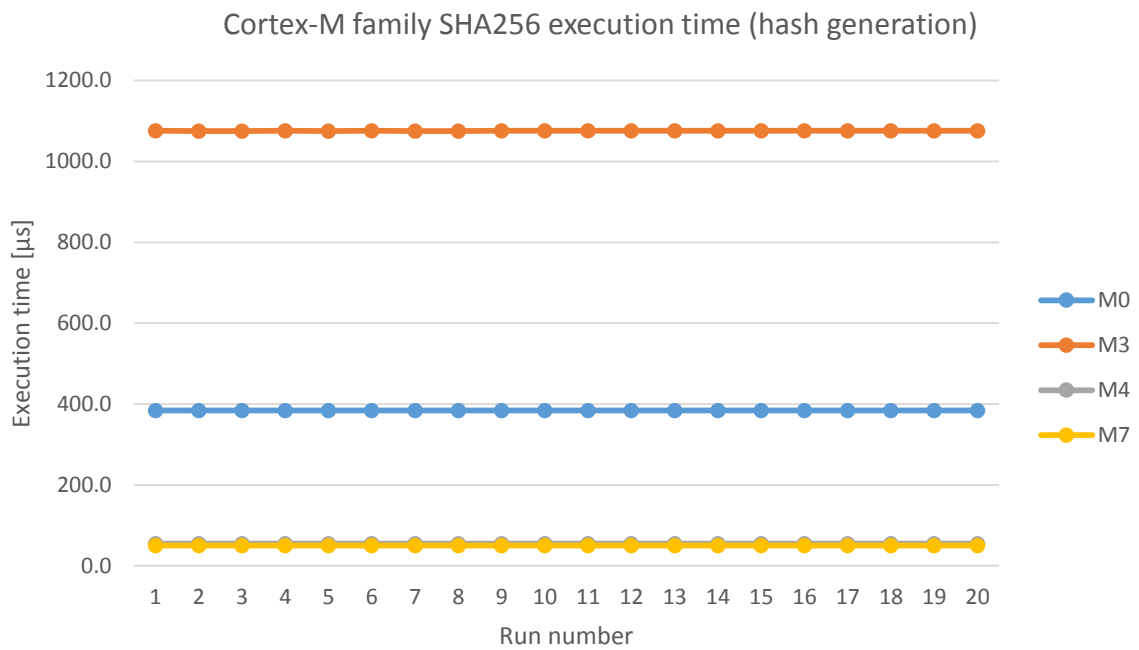


Figure 4.7 Execution time of Cortex-M series processors running SHA256 (hash generation only)

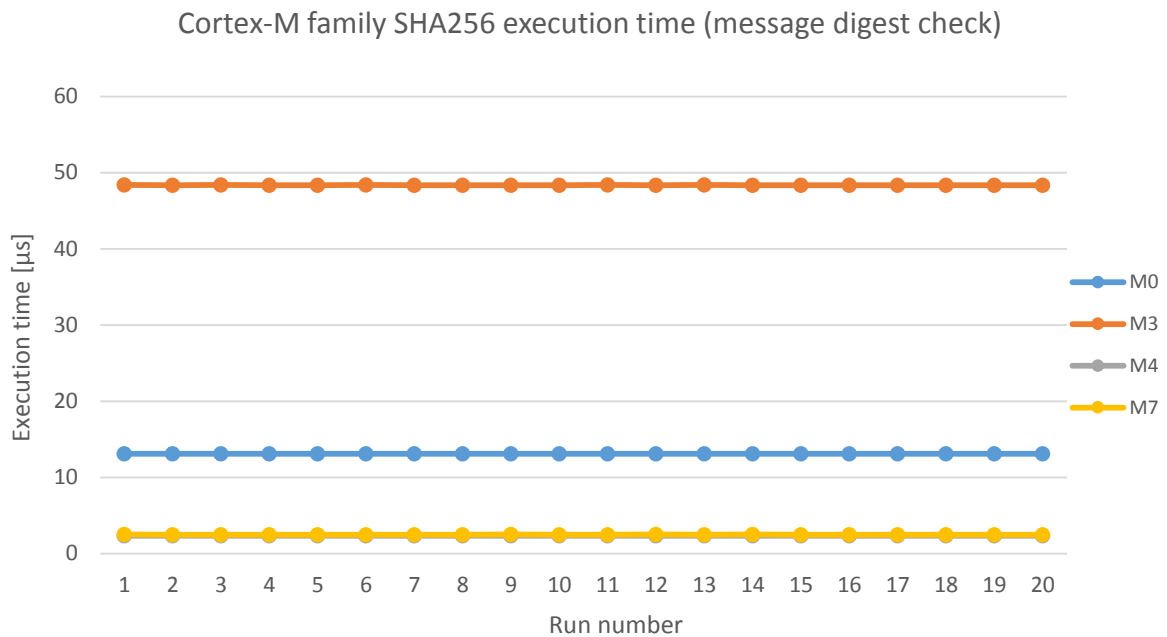


Figure 4.8 Execution time of Cortex-M series processors running SHA256 (message digest checking only)

Quite interestingly, the observed performances of the M4 and M7 for the total execution time, hash generation and digest verification were, once again, close to identical. In the three (3) figures, the graph illustrating the observed performance of the M4 had been obscured owing to an overlap with the graph illustrating the performance observed for the M7 processor. Another observation of interest was that the majority of the execution time observed for SHA256 was utilised in the generation of the hash function. In comparison, the time spent during the validity checking of the generated message digest was minimal.

Once again, in order to confirm mathematically the trends observed from the graphs, the results from the experiment runs were used to calculate the mean, standard deviation and standard error for the execution time estimates of each of the MCU. They are given in Tables 4.6 to 4.8:

Table 4.6 Execution time mean, standard deviation and standard error for MCUs running SHA256

	\bar{X}	σ_x	$\sigma_{\bar{x}}$
	[μs]	[μs]	[μs]
M0	397.000	0.205	0.046
M3	1124.000	0.000	0.000
M4	57.361	0.035	0.008
M7	56.029	0.018	0.004

Using the results above, the total execution times for SHA256 on the Cortex-M processors were estimated as follows:

$$\text{M0: } 397.000 \mu\text{s} \pm 0.046 \mu\text{s}$$

$$\text{M3: } 1124.000 \mu\text{s} \pm 0.000 \mu\text{s}$$

$$\text{M4: } 57.361 \mu\text{s} \pm 0.008 \mu\text{s}$$

$$\text{M7: } 56.029 \mu\text{s} \pm 0.004 \mu\text{s}$$

Table 4.7 Execution time mean, standard deviation and standard error for MCUs running SHA256 (hash generation only)

	\bar{X}	σ_x	$\sigma_{\bar{x}}$
	[μs]	[μs]	[μs]
M0	384.040	0.123	0.028
M3	1075.750	0.444	0.099
M4	55.102	0.017	0.004
M7	50.438	0.006	0.001

From the results given in Table 4.7, the execution times for the hash generation operations could be estimated as:

M0: $384.040 \mu\text{s} \pm 0.028 \mu\text{s}$

M3: $1075.750 \mu\text{s} \pm 0.099 \mu\text{s}$

M4: $55.102 \mu\text{s} \pm 0.004 \mu\text{s}$

M7: $50.438 \mu\text{s} \pm 0.001 \mu\text{s}$

Table 4.8 Execution time mean, standard deviation and standard error for MCUs running SHA256 (message digest checking only)

	\bar{X}	σ_x	$\sigma_{\bar{x}}$
	[μs]	[μs]	[μs]
M0	13.086	0.008	0.002
M3	48.371	0.016	0.003
M4	2.363	0.001	0.000
M7	2.491	0.006	0.001

Similarly, the execution times for the digest verification operations were estimated from the results in Table 4.8 as:

M0: $13.086 \mu\text{s} \pm 0.002 \mu\text{s}$

M3: $48.371 \mu\text{s} \pm 0.003 \mu\text{s}$

M4: $2.363 \mu\text{s} \pm 0.000 \mu\text{s}$

M7: $2.491 \mu\text{s} \pm 0.001 \mu\text{s}$

The foregoing estimations gave a better illustration of how close the observed execution times for the M4 and M7 were. With the estimated mean for the full execution time, only a difference of $1.332 \mu\text{s}$ was seen between the two processors while a difference of $0.128 \mu\text{s}$ was observed for the execution time of the digest check. The largest difference was observed for the execution time of the hash generation process at $4.664 \mu\text{s}$. As with the previous two cryptographic algorithms, very little deviation was observed in the execution times; illustrating that the processors were capable of running the cryptographic algorithms within a predictable time period. This could allow for easier planning and adjustments when determining the allowed delay tolerance in an IIoT network.

4.3 POWER CONSUMPTION

The power consumed by the MCUs was determined using the measured voltage drop, V_{MCU} , taken across a shunt resistor of resistance R . The current consumed by the MCU during the execution of the cryptographic algorithms, I_{MCU} , was calculated according to Ohm's Law using (4.4):

$$I_{MCU} = V_{MCU} \div R \quad \text{Amperes (A)} \quad (4.4),$$

where the resistance of the shunt resistor R was measured to equal 1.3 ohms (Ω).

Equation (4.5) was then used to calculate the power consumption of the MCU:

$$P_{MCU} = V_S \times I_{MCU} \quad \text{Watts (W)} \quad (4.5),$$

where the supply voltage from the ST-Link/USB connection, V_S , was given as 5.0 Volts (V).

The final consumption results were converted to milliwatts (mW) and rounded to three decimal places.

To determine the statistical error of the measured results over the number of runs N , the mean (\bar{X}), standard deviation (σ_X) and standard error of the mean ($\sigma_{\bar{X}}$) were determined using (4.1) to (4.3).

4.3.1 AES128-CTR

As in the execution time experiments, the power consumption experiments were conducted using the CTR mode of AES128 on the Cortex-M processors. The experiment measured the consumption of the algorithm from the start of the AES algorithm at encryption to its conclusion after a successful decryption. Twenty (20) runs were conducted on each development board, the results of which have been illustrated in Figure 4.9:

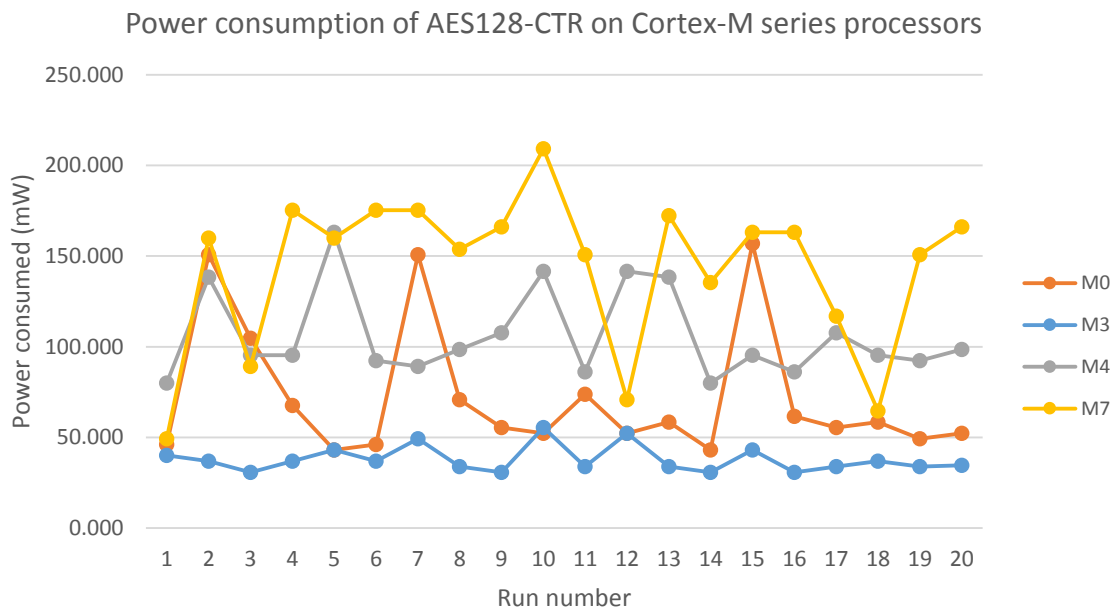


Figure 4.9 Power consumption of Cortex-M series processors running AES128-CTR

Looking at Figure 4.9, it could be seen that, apart from the M3, the consumption of the processors was capable of fluctuating widely. The M7 presented the largest fluctuation, with its lowest consumption at approximately 50mW and its highest consumption at just over 200mW. From the figure, it also appeared that the power consumption of the processors and the amount of fluctuation in consumption increased with the increase in operating frequency of the processors.

In order to determine mathematically the true extent of deviation illustrated in Figure 4.9, the mean, standard deviation, and standard error were calculated on the results from the twenty (20) runs.

Table 4.9 Power consumption mean, standard deviation and standard error for MCUs running AES128-CTR

	\bar{X}		σ_x		$\sigma_{\bar{X}}$	
	[mW]	[W]	[mW]	[W]	[mW]	[W]
M0	72.462	0.072	37.292	0.037	8.339	0.008
M3	37.885	0.038	7.254	0.007	1.622	0.002
M4	106.154	0.106	24.341	0.024	5.443	0.005
M7	143.385	0.143	42.838	0.043	9.579	0.010

Using the calculated mean and standard error presented in Table 4.9, the consumption of the processors running AES128-CTR could be estimated as follows:

$$M0: 72.462 \text{ mW} \pm 8.339 \text{ mW}$$

$$M3: 37.885 \text{ mW} \pm 1.622 \text{ mW}$$

$$M4: 106.154 \text{ mW} \pm 5.443 \text{ mW}$$

$$M7: 143.385 \text{ mW} \pm 9.579 \text{ mW}$$

It could be seen that while the M3 did consume the least power and offered the least deviation, the M4 presented a smaller deviation than the M0, despite having a larger operating frequency. The larger deviation seen in the M0, however, was insufficient for the upper bound power consumption estimate to match the lower bound consumption of the M4, which could be seen as one of the larger power consumers in the series. The M7 presented the largest consumption and the largest deviation from the processor series; with a mean consumption of 143.385mW and an estimated standard error of 9.579mW.

With the large deviation ranges presented by the processors, should software-implemented AES be required on an edge node, care would need to be taken during the design and consideration of external power sources to ensure that (a) the sources were capable of supporting the node operations at both the minimum estimated consumption and the maximum estimated consumption over the deployment lifecycle, and that (b) surge detection and protection circuitry was capable of distinguishing a legitimate, anomalous

power consumption deviation from the regular, large deviations seen in the power consumption profile of each particular MCU.

4.3.2 ECDSA

The power consumption of the two (2) versions of the public key cryptographic algorithm ECDSA was tested on the M-series processors. Figure 4.10 compares the power consumption of the MCUs running ECDSA (Sign-Verify) over the course of twenty (20), independent runs, while Figure 4.11 compares the power consumption of the MCUs running ECDSA (Key Gen-Sign-Verify) over twenty (20) runs.

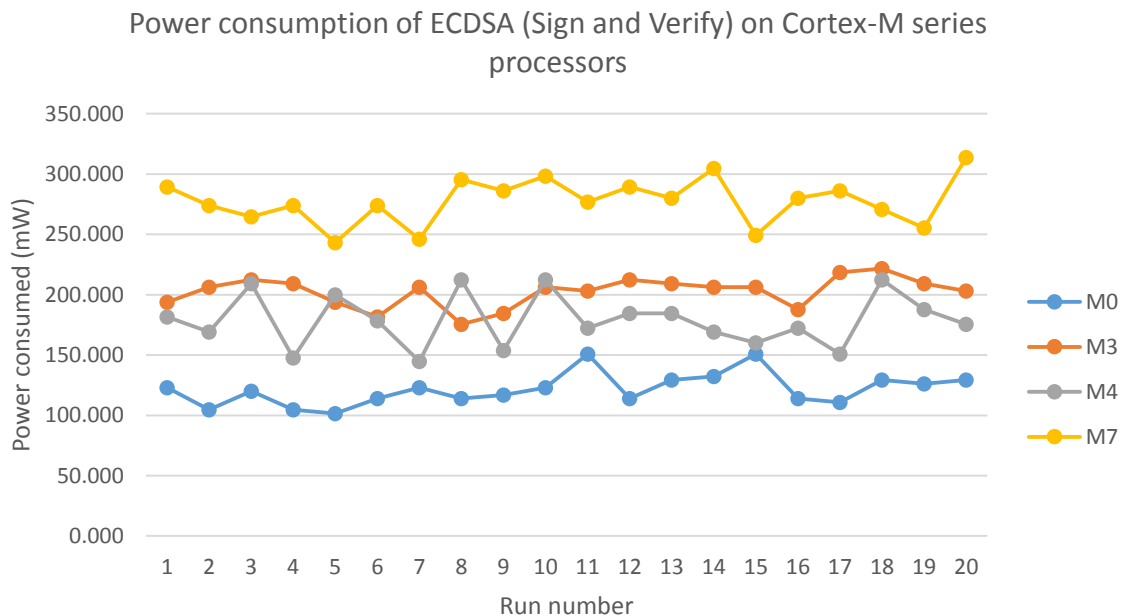


Figure 4.10 Power consumption of Cortex-M series processors running ECDSA (Sign-Verify)

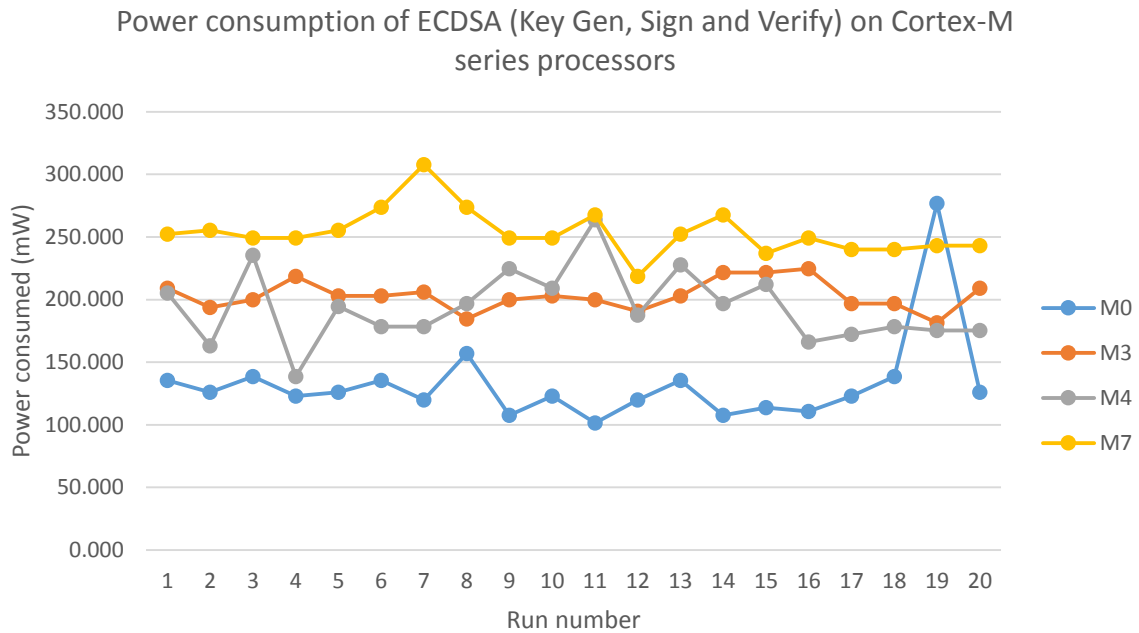


Figure 4.11 Power consumption of Cortex-M series processors running ECDSA (Key Gen- Sign-Verify)

It could be seen that, for ECDSA (Sign-Verify), the M0 and M4 MCUs consumed the least power. For ECDSA (Key Gen-Sign-Verify), the M0 was, once more, the least power consumption heavy MCU, whereas the performances of the M3 and M4 were very similar. In both instances, the M7 MCU consumed the most power.

Overall, the amount of deviation seen in the graph patterns for both Sign-Verify ECDSA and Key Gen-Sign-Verify ECDSA was relatively stable; however, ECDSA without key generation appeared to give a more stable power consumption profile for the MCUs than ECDSA with key generation.

In order to provide mathematical confirmation for the aforementioned observations, using the mean and standard error given in Table 4.10, the consumption of the processors running ECDSA (Sign-Verify) was estimated as follows:

Table 4.10 Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify)

	\bar{X}		σ_x		$\sigma_{\bar{x}}$	
	[mW]	[W]	[mW]	[W]	[mW]	[W]
M0	121.538	0.122	13.337	0.013	2.982	0.003
M3	202.308	0.202	12.278	0.012	2.746	0.003
M4	178.923	0.179	21.855	0.022	4.887	0.005
M7	277.538	0.278	19.140	0.019	4.280	0.004

M0: 121.538 mW \pm 2.982 mW

M3: 202.308 mW \pm 2.746 mW

M4: 178.923 mW \pm 4.887 mW

M7: 277.538 mW \pm 4.280 mW

Similarly, the consumptions of ECDSA (Key Gen- Sign-Verify) were estimated using the figures given in Table 4.11:

Table 4.11 Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Key Gen- Sign-Verify)

	\bar{X}		σ_x		$\sigma_{\bar{x}}$	
	[mW]	[W]	[mW]	[W]	[mW]	[W]
M0	132.308	0.132	36.461	0.036	8.153	0.008
M3	203.077	0.203	11.723	0.012	2.621	0.003
M4	194.038	0.194	29.129	0.029	6.513	0.007
M7	253.692	0.254	18.285	0.018	4.089	0.004

M0: 132.308 mW \pm 8.153 mW

M3: 203.077 mW \pm 2.621 mW

M4: 194.038 mW \pm 6.513 mW

M7: 253.692 mW \pm 4.089 mW

Looking at the standard errors, one could see that the observed deviation for ECDSA without key generation was contained within a smaller range than the deviations observed for ECDSA with key generation; with ECDSA (Sign-Verify) giving a range of 2.141mW and ECDSA (Key Gen-Sign-Verify) giving a range of 5.532mW. Between the two (2) versions of ECDSA, the M0 and M4 MCUs saw an increase slightly over 8% in power consumption with key generation. The M0 was observed to have had a 173.407% increase in deviation and the M4 was shown to have had a 33.271% increase in deviation with the inclusion of key generation. These figures showed that, in addition to an increase in power consumption, the inclusion of key generation had the effect of increasing the variability in the power consumed by the M0 and M4; highlighting that adequate surge detection trigger rules would need to be created and adjusted for any surge detection circuitry should it be required that key generation be utilised with the end nodes running ECDSA. The variability seen with the inclusion of key generation to ECDSA may also be indicative of the performance that may be observed from these two processors with other cryptographic algorithms requiring key generation using a PRNG.

The performance of the M3 and M7 were of particular interest. With the inclusion of key generation to the ECDSA algorithm, the M3 experienced only a 0.380% increase in power consumed and a 4.552% decrease in standard error. This showed that, while the power consumed between both algorithms was very similar, the inclusion of key generation gave a more stable power consumption profile in M3, over the twenty (20) run experiment. The M7 experienced decreases in both power consumed and deviation; with a decrease of 8.592% in power consumed and a decrease of 4.463% in observed deviation. This showed that the inclusion of key generation using a PRNG led to improved performance for ECDSA when run on the M7. The improvements observed with the inclusion of PRNG on these two processors raised the question of their possible performance when a PRNG is

used for key generation with other cryptographic algorithms. This will be considered for future work on this topic.

4.3.3 SHA256

Power consumption experiments for SHA256 were conducted over twenty (20) runs on each MCU. As in the consumption experiments for the other cryptographic algorithms, the power was measured from the start of the hashing algorithm to the conclusion of the algorithm. The graphical results of the power consumption observed in the MCUs over the course of the experiment are given in Figure 4.12.

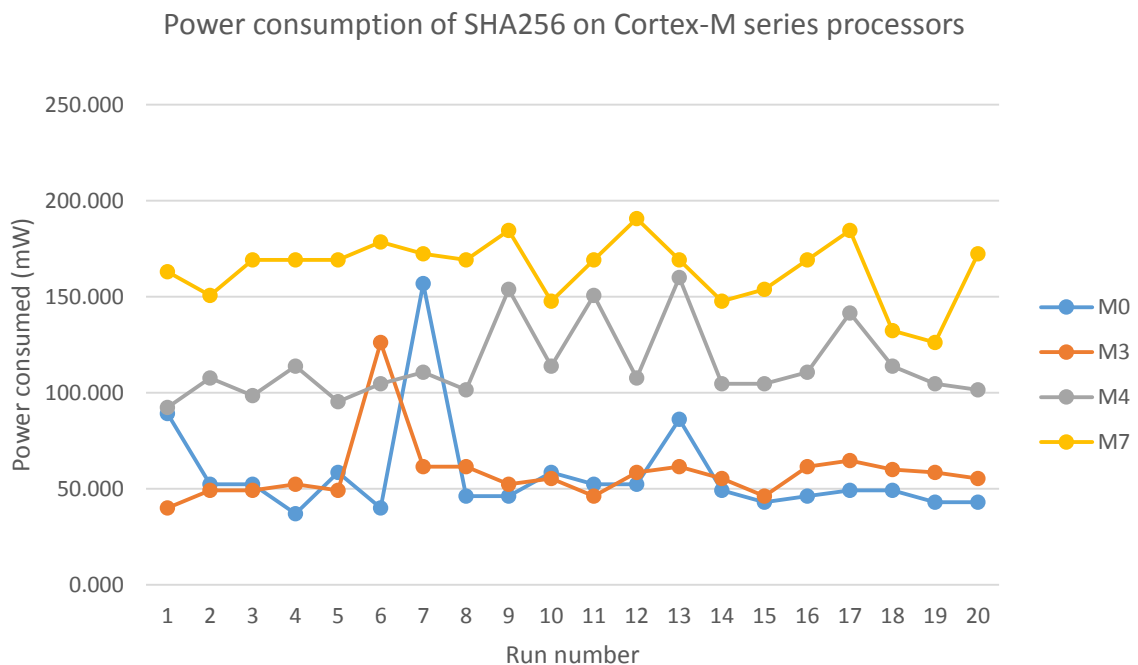


Figure 4.12 Power consumption of Cortex-M series processors running SHA256

The power consumption observed in the four (4) processors was surprisingly unstable, with the M3 showing the least amount of deviation. The power consumed by the M0 and M3 was seen to be very similar; however, the M0 displayed spikes in power consumption that would match, near match or exceed the observed consumption of the M4 processor.

Utilising the values given in Table 4.12, the power consumption of the M-series processors was estimated as follows:

Table 4.12 Power consumption mean, standard deviation and standard error for MCUs running SHA256

	\bar{X}		σ_x		$\sigma_{\bar{x}}$	
	[mW]	[W]	[mW]	[W]	[mW]	[W]
M0	57.538	0.058	26.919	0.027	6.019	0.006
M3	58.231	0.058	17.211	0.018	3.848	0.004
M4	114.615	0.115	20.037	0.020	4.480	0.004
M7	164.462	0.164	16.778	0.017	3.752	0.004

$$M0: 57.538 \text{ mW} \pm 6.019 \text{ mW}$$

$$M3: 58.231 \text{ mW} \pm 3.848 \text{ mW}$$

$$M4: 114.615 \text{ mW} \pm 4.480 \text{ mW}$$

$$M7: 164.462 \text{ mW} \pm 3.752 \text{ mW}$$

The results of the estimations showed that; while the M0 and M3 did display highly similar power consumption averages, the M0 had a larger, observed deviation than the M3 processor. More precisely, the M0 processor displayed the largest observed deviation from the processor series. The deviations observed from the remaining three processors were within range of the others while the M7 gave the largest, observed power consumption on average.

4.4 MEMORY OCCUPATION

The Build/Memory analyser tool built into Atollic TrueStudio allowed for a detailed analysis to be given of the RAM and Flash memory occupation for the debugger elf file which was generated for the selected processor. An analysis of the RAM and Flash occupation was conducted across the M-series processors for each of the cryptographic

algorithms in order to determine the extent to which memory resources are consumed by the security algorithms.

4.4.1 RAM

The detailed results of the RAM occupation analysis conducted for each cryptographic algorithm on the M-series processors are summarised in Table 4.13. A graphical representation of the percentage of RAM used by the algorithms is presented in Figure 4.13:

Table 4.13 RAM Occupation of cryptographic algorithms loaded onto Cortex-M series processors

	M0			M3			M4			M7		
	RAM			RAM			RAM			RAM		
	[8kB]			[8kB]			[128kB]			[512kB]		
	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)
AES128-CTR	6.33	1.67	20.85	6.36	1.64	20.51	124.38	3.62	2.83	509.15	2.85	0.56
ECDSA-S-V	6.43	1.57	19.63	6.46	1.54	19.29	124.48	3.52	2.75	510.43	1.57	0.31
ECDSA-G-S-V	6.43	1.57	19.63	6.46	1.54	19.29	124.48	3.52	2.75	510.43	1.57	0.31
SHA256	6.39	1.61	20.12	6.42	1.58	19.78	124.44	3.56	2.78	509.21	2.79	0.54

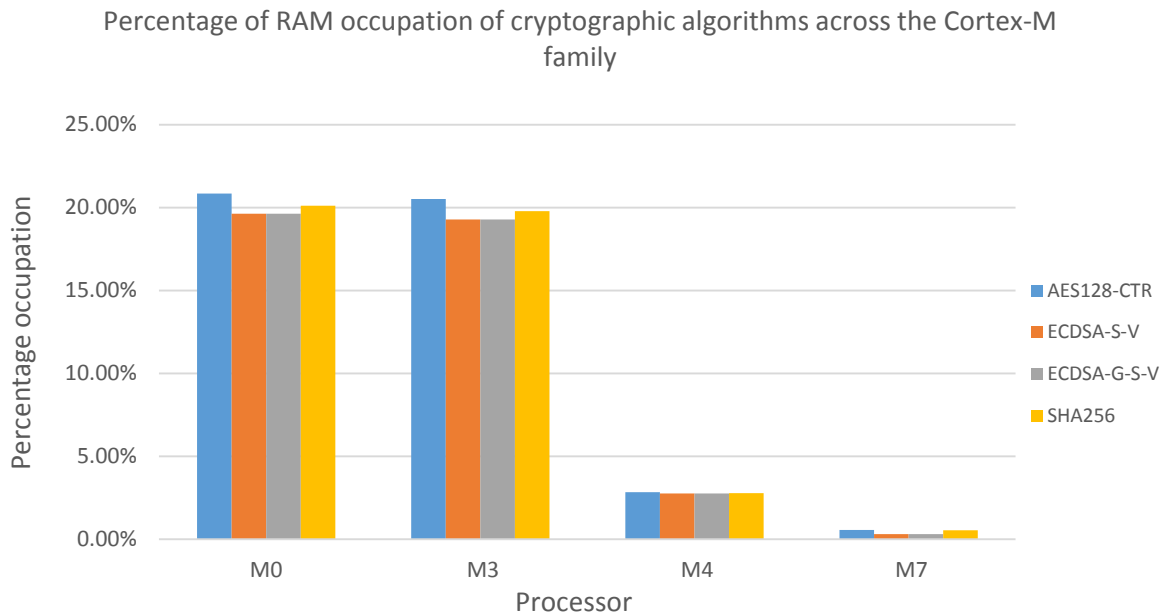


Figure 4.13 Comparison of the percentage RAM occupation of cryptographic algorithms loaded onto Cortex-M processors

It could be seen that, for the four (4) algorithms tested, the space occupied in RAM for the M0 and M3, both of which have 8kB of available RAM, was very similar; with AES128-CTR and SHA256 occupying the most RAM at slightly above 20% each, and ECDSA occupying the least RAM at slightly above 19%. Even with the cryptographic algorithms occupying a fair portion of RAM, approximately 80% of RAM was still available for the use of the MCUs in other applications and processes. This, however, would decrease relatively quickly as more algorithms are loaded onto the processors.

The observed RAM occupation significantly drops when considering the M4 and M7 processors, where the available RAM was 128kB and 512kB respectively. The M4 observed a very similar RAM occupation across the four (4) algorithms with AES128-CTR occupying slightly more RAM at 2.83%. The occupation of the cryptographic algorithms on the M7 could be considered almost insignificant, with not one of the algorithms occupying at least 1% of the available RAM. In this instance, the MCU had nearly all the RAM available to it for other applications and processes and could easily support the inclusion of multiple cryptographic algorithms. In all the cases, however, it was observed

that the inclusion of the cryptographic algorithms did not serve to deplete the available RAM resources to a point where further operations could be compromised.

4.4.2 Flash

As with the RAM occupation, analysis of the occupation of the cryptographic algorithms in Flash was conducted using the Atollic TrueStudio Build/Memory analyser. The detailed results of the analysis are presented in Table 4.14 with a graphical presentation of the percentage occupation given in Figure 4.14:

Table 4.14 Flash occupation of cryptographic algorithms loaded onto Cortex-M series processors

	M0			M3			M4			M7		
	Flash			Flash			Flash			Flash		
	[64kB]			[128kB]			[1024kB]			[2048kB]		
	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)	Free (kB)	Used (kB)	Used (%)
AES128-CTR	55.72	8.28	12.93	119.92	8.08	6.31	1004.69	19.31	1.89	2037.16	10.84	0.53
ECDSA-S-V	46.27	17.73	27.70	110.66	17.34	13.55	995.36	28.64	2.80	2028.46	19.54	0.95
ECDSA-G-S-V	38.96	25.04	39.13	103.59	24.41	19.07	988.31	35.69	3.49	2021.87	26.13	1.28
SHA256	57.68	6.32	9.88	121.85	6.15	4.80	1006.2	17.8	1.74	2039.43	8.57	0.42

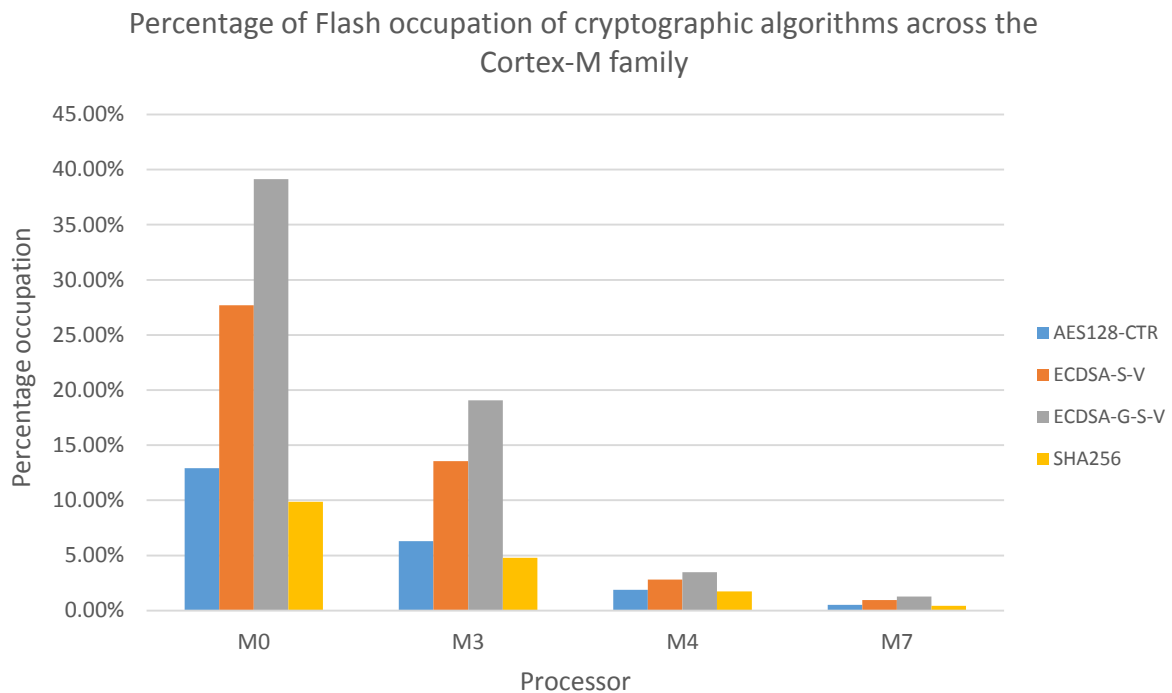


Figure 4.14 Comparison of the percentage Flash occupation of cryptographic algorithms loaded onto Cortex-M processors

Unlike the RAM, a larger variation in space occupation occurred in Flash. From the four (4) processors, ECDSA with key generation occupied the most Flash memory across the M-Series processors, with ECDSA sans key generation being the next largest algorithm. Of the four (4) processors, the cryptographic algorithms had the largest percentage occupation on the M0, which had the least amount of available Flash memory at 64kB. As the amount of available Flash in the processor increased, the percentage occupation of the cryptographic algorithms decreased, with the M7 displaying the smallest percentage occupation of its 2048kB Flash. It was noted, however, that the largest percentage occupation was observed from the smallest available Flash at just below 40%, leaving approximately 60% of the remaining Flash available. While this is a significantly larger occupation than that observed in RAM, a total depletion of resources had not occurred, and sufficient resources would still be available for other MCU applications and processes. Due care and planning may need to be taken when utilising larger algorithms with the M0 processor to ensure that any additional processes that may be required to run would have

sufficient Flash memory. In such instances, the use of an alternative but smaller algorithm providing a similar level of security and performance trade-off may prove to be beneficial.

4.5 CHAPTER SUMMARY

This chapter served to present the results observed from the execution time, power consumption and memory occupation experiments presented in Chapter 3. A brief analysis of the results was conducted after presenting the mean and average estimations of the execution time and power consumption alongside the estimated error observed from the mean results. The space occupation by the cryptographic algorithms was determined to allow the majority of the available memory resources to remain unoccupied for additional MCU processes and applications, unlike the total depletion of resources that was feared to occur in older generation processors.

In Section 4.1 the overview and objectives for the chapter were presented.

In Section 4.2 the results of the execution time experiments were given for AES128-CTR, ECDSA and SHA256; giving the mean execution time, standard deviation and standard mean error for the Cortex-M processors.

In Section 4.3 the results of the power consumption experiments were given for AES128-CTR, ECDSA and SHA256; giving the mean power consumption, standard deviation and standard mean error for the Cortex-M processors.

In Section 4.4 the results of the memory occupation experiments were given for AES128-CTR, ECDSA and SHA256; giving the RAM and Flash consumed by the algorithms on the Cortex-M processors.

The following chapter provides a detailed analysis of the overall performance of software-implemented cryptography on new generation processors. The analysis is based on the results presented in this chapter and highlights the viability of software solutions within the construction of a secure IIoT endpoint node.

CHAPTER 5 DISCUSSION

5.1 CHAPTER OBJECTIVES

Chapter 4 served to provide a report on the results of the experiments conducted and presented in Chapter 3 of this study; looking at the individual performances of the algorithms on each of the Cortex-M series processors. It was seen that the performance of the algorithms varied between the processors, dependent on the operating frequency. It was also seen that the degree of deviation that could be seen in the time and power estimates changed dependent on the algorithm; with some processors executing the algorithms more consistently than other processors in the series. This chapter serves to give a detailed analysis of the reported results and to answer the research question posed at the beginning of this study.

In Section 5.2 a detailed discussion on the performance of the processors running software-implemented cryptography is provided; comparing the performances seen between the four processors and comparing the performances of the new-generation Cortex-M processors against the performances seen in the older-generation Atmega128L processor.

In Section 5.3 the best performing of the four (4) Cortex-M processors is selected based on the overall performance given while running the cryptographic algorithms.

Section 5.4 provides a brief summary of the main topics covered and serves to conclude the chapter.

5.2 PERFORMANCE OF SOFTWARE-IMPLEMENTED CRYPTOGRAPHY

To try to determine the viability of software-implemented cryptography as a tool towards the design of a secure mote for the IIoT, one needs to consider the performance of the algorithms in terms of execution time, power consumption and memory resource consumption. An IIoT network has a variety of different operational requirements, one of which may be real-time operation. Real-time operation is highly dependent on the ability of the system to meet a pre-determined deadline while generating a correct response. Industrial control systems and safety or mission-critical systems typically utilise hard deadlines, where a missed deadline can constitute complete system failure, as predictability is a main requirement of a real-time system. Within these parameters, the addition of cryptographic operations should not impede upon the ability of the system to meet its deadlines. Considering the results given in Chapter 4, software-implemented cryptography appeared to be a good candidate for use in hard real-time operations. It was seen that, over the course of the twenty (20) runs, very little deviation in the execution time of the algorithms occurred. This showed that the Cortex-M processors were capable of running the cryptographic algorithms within a predictable time period and with a relatively low chance of a sudden, large jump in operating time between executions. Figure 5.1 gives the average execution time performances determined for the identified cryptographic algorithms.

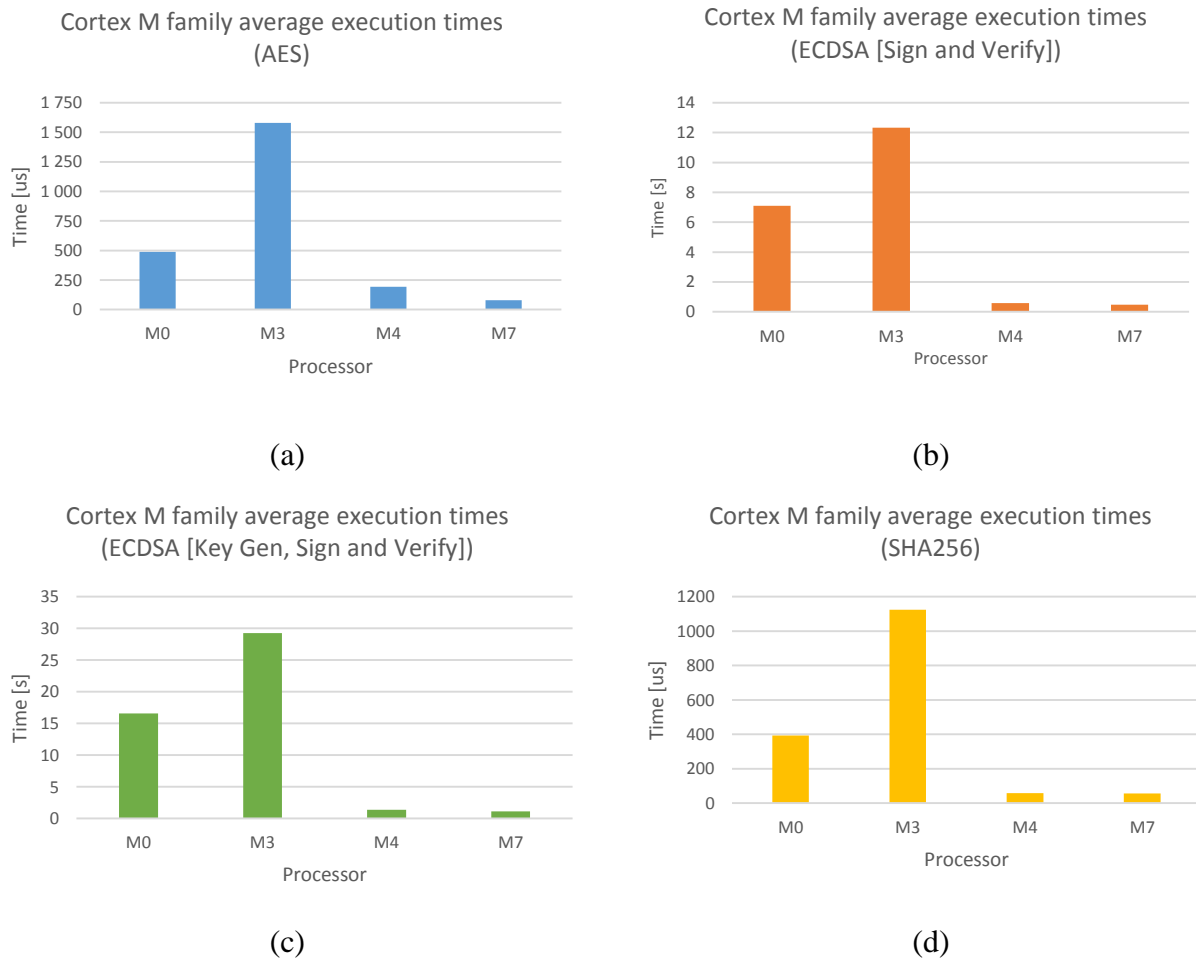


Figure 5.1 Average execution times of cryptographic algorithms on Cortex-M processors

(a) AES128-CTR, (b) ECDSA (Sign-Verify), (c) ECDSA (Key Gen-Sign-Verify) and (d) SHA256

Looking at the average execution times, one could see that, as could be expected, the performance of the processors was directly related to their operating frequency. The M3, with the smallest operating frequency, consistently gave the slowest execution time; followed by the M0, with the next smallest operating frequency. The performance of the M4 was surprising in that, apart from AES128-CTR, it showed a very similar execution time to the M7 processor, in spite of its slower operating frequency. For the four (4) algorithms tested, it could be seen that there was little benefit to the more powerful M7 processor when executing cryptographic algorithms, as the M4 was capable of delivering a similar performance.

The specific trends seen in the execution times of the processors were also interesting to note. The M0, M4, and M7 were capable of running the symmetric algorithms in a time period that may be considered sufficiently fast for use as part of hard real-time tasks; as their addition to the processing time would be within the realm of microseconds. In these cases, it appeared that the processors were capable enough themselves to run cryptographic algorithms without the need of adding a hardware crypto accelerator. The same could not be said for the public key cryptographic algorithm. The fastest execution times, as given by the M7 processor, were 471.02ms for ECDSA without key generation and 1.141s for ECDSA with key generation. These execution times, especially in the case where key generation is used, would be sufficiently long to increase the possibility of introducing cascading delay into the IIoT network and, with that, missed operation deadlines.

Depending on the deadline definitions for the IIoT network, some of the symmetric cryptography algorithms could be run on the M3 without the need for hardware acceleration; as AES128-CTR and SHA256 gave average execution times at 1.578ms and 1.124ms respectively. As with the more powerful processors, the added delay for the tested public key algorithm was sufficiently long that the addition of the algorithm to hard real-time tasks would possibly cause missed operation deadlines. Should public key cryptography be required as a part of the network security architecture, a number of alternative possibilities could be used in the place of software-implemented libraries. One option would be the use of a hardware crypto accelerator with the standard Cortex-M processors. In addition to providing acceleration in the execution of cryptographic processes, one might be able to establish a root of trust from which node operations are verified. Extra care would need to be taken, however, to ensure that security information was not leaked within the communications between the MCU and the hardware accelerator. Another option is the use of a security-enabled MCU, such as those given in Table 2.11 in Chapter 2. Security-enabled MCUs are specifically designed to provide a variety of security operations, quickly and efficiently, in addition to cryptography services. A comparison of the abilities of a security MCU and a standard MCU running a software cryptographic library will be conducted in the future in order to determine an exact speed-up factor that could be seen through the use of a security MCU.

In addition to the processors being able to meet the hard deadline requirements of real-time operation, the power consumption of the processors needed to be determined as part of maximising the operational lifetime of the node power supply. IIoT network deployments can be large and in areas where regular maintenance activities would be difficult and costly to complete. One would, therefore, want to maximise the time between maintenances and minimise as much power consumed during operational activities as possible, so as not to drain the power supply to the network endpoint too quickly. Figure 5.2 gives a comparison of the power consumption for the four (4) cryptographic algorithms run on the Cortex M processors.

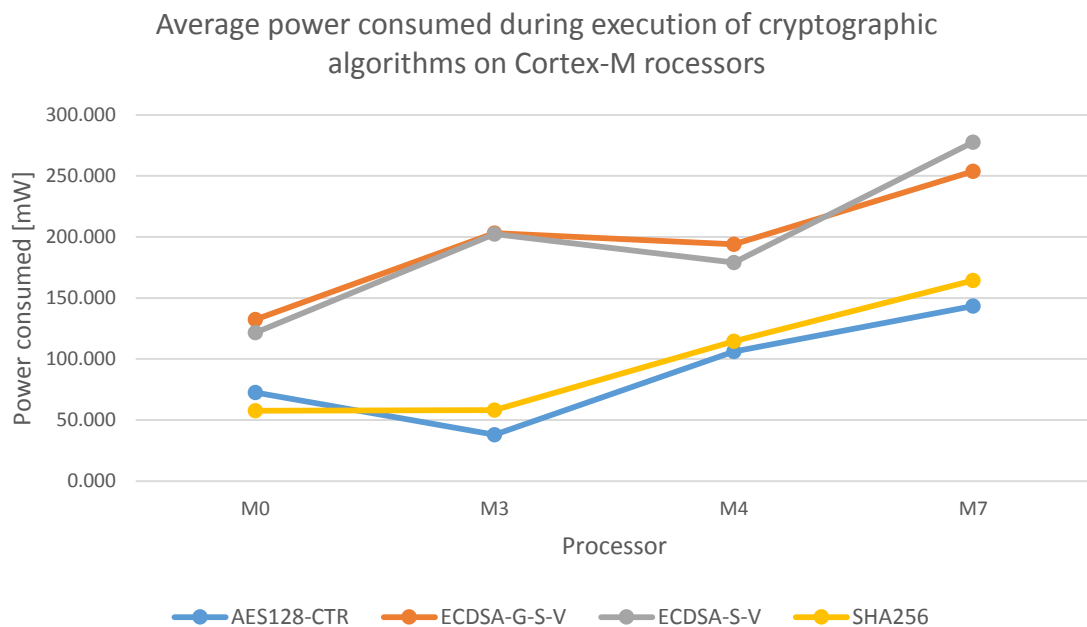


Figure 5.2 Average power consumption of cryptographic algorithms on Cortex-M Processors

Of the tested algorithms, AES128-CTR gave the lowest power consumption while ECDSA gave the highest consumption. It was interesting to note that the power consumption of the ECDSA algorithm was very similar whether key generation was utilised or not utilised. Comparing the four (4) MCUs, one could see that different algorithms performed better on the different processors. Looking at Figure 5.2, one could see that the M3 processor was the lowest consumer for AES128-CTR; the M0 gave the lowest power consumption for both versions of ECDSA; and the M0 and M3 gave similar power consumptions for

SHA256. The power consumptions from the M4 processor were varied; especially when compared to the consumptions of the other three (3) Cortex-M processors. With the symmetric algorithms, it gave the second largest power consumption, whereas when running the public key algorithm, it gave the second lowest power consumption, with the power consumption of the M3 preceding it. Throughout the experiments, the M7, as the most powerful processor, gave the largest power consumptions for the four (4) algorithms.

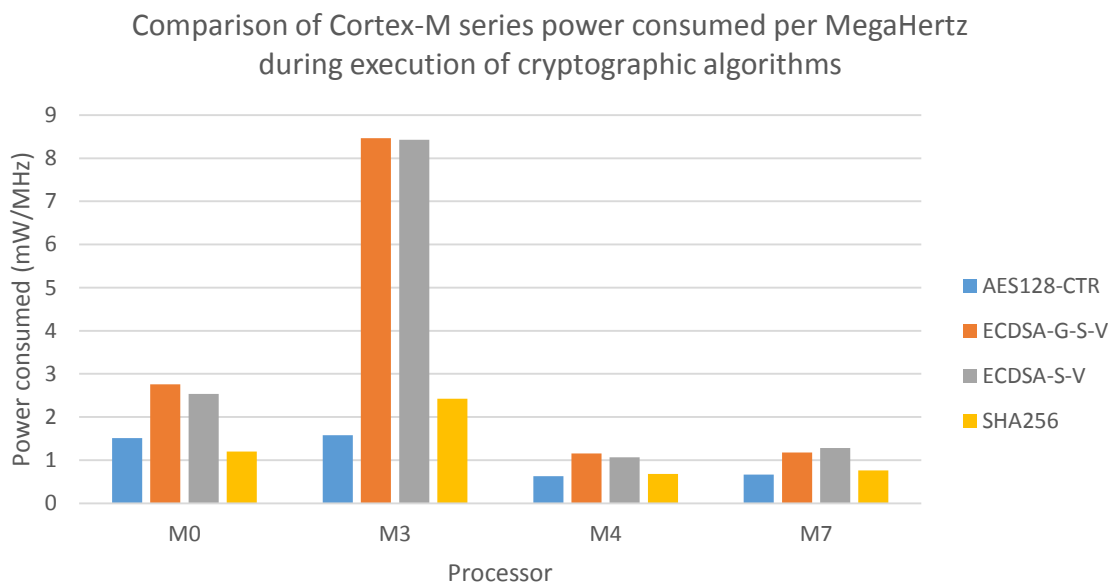


Figure 5.3 Average power consumption per MHz of cryptographic algorithms on Cortex-M processors

Should the operating temperature of the processor be a higher concern for the IIoT network than average power consumed, the identification of a more power-efficient processor may be of greater interest. To determine the power efficiency of the MCUs independent of the differing operating frequencies, the consumption per MHz for each MCU was calculated and is given in Figure 5.3. Opposed to the average consumptions seen in Figure 5.2, it could be seen that, with the normalisation of the power consumption, the Cortex-M3 was, on average, the least power efficient processor; giving highest power consumption per MHz when executing the cryptographic algorithms. The Cortex-M0 was seen to be the second highest consumer per MHz available to the processor. The high consumptions per MHz could result in warmer processors when running cryptographic processes; thus,

adequate cooling mechanisms in the enclosure design would need due consideration to prevent node failure owing to overheating.

Interestingly, the power consumption per MHz of the Cortex-M4 and the Cortex-M7 were very similar across the four cryptographic algorithms. This showed that, despite its higher clocking speed, the Cortex-M7 was capable of achieving similar power efficiency per MHz to its less powerful predecessor. This bodes well for IIoT edge applications, such as network gateways, which may require a more powerful processor for local processing activities, as upgrades to more intensive cooling mechanisms may not be required after the addition of software cryptographic operations.

The estimated consumption per MHz seen in Figure 5.3 could also be used as a guide towards generating a power consumption profile in cases where more powerful MCU versions of the Cortex processors may need to run cryptographic algorithms. Developers could extrapolate the results given in order to make the power consumption estimates for the more powerful processors running software cryptographic algorithms. This would allow for the incorporation of the necessary power allowances required for a security implementation during the initial design of the IIoT network as opposed to making support alterations for the additional power consumption after the deployment of the network.

To determine the best overall performer in the execution of the cryptographic algorithms, the average consumption and execution times seen for the processors needed to be considered as a single unit. Figure 5.4 compares the power consumption and execution times seen for the tested cryptographic algorithms using a combination of a split column and scatter plot in order to display the overall performance of the four (4) processors.

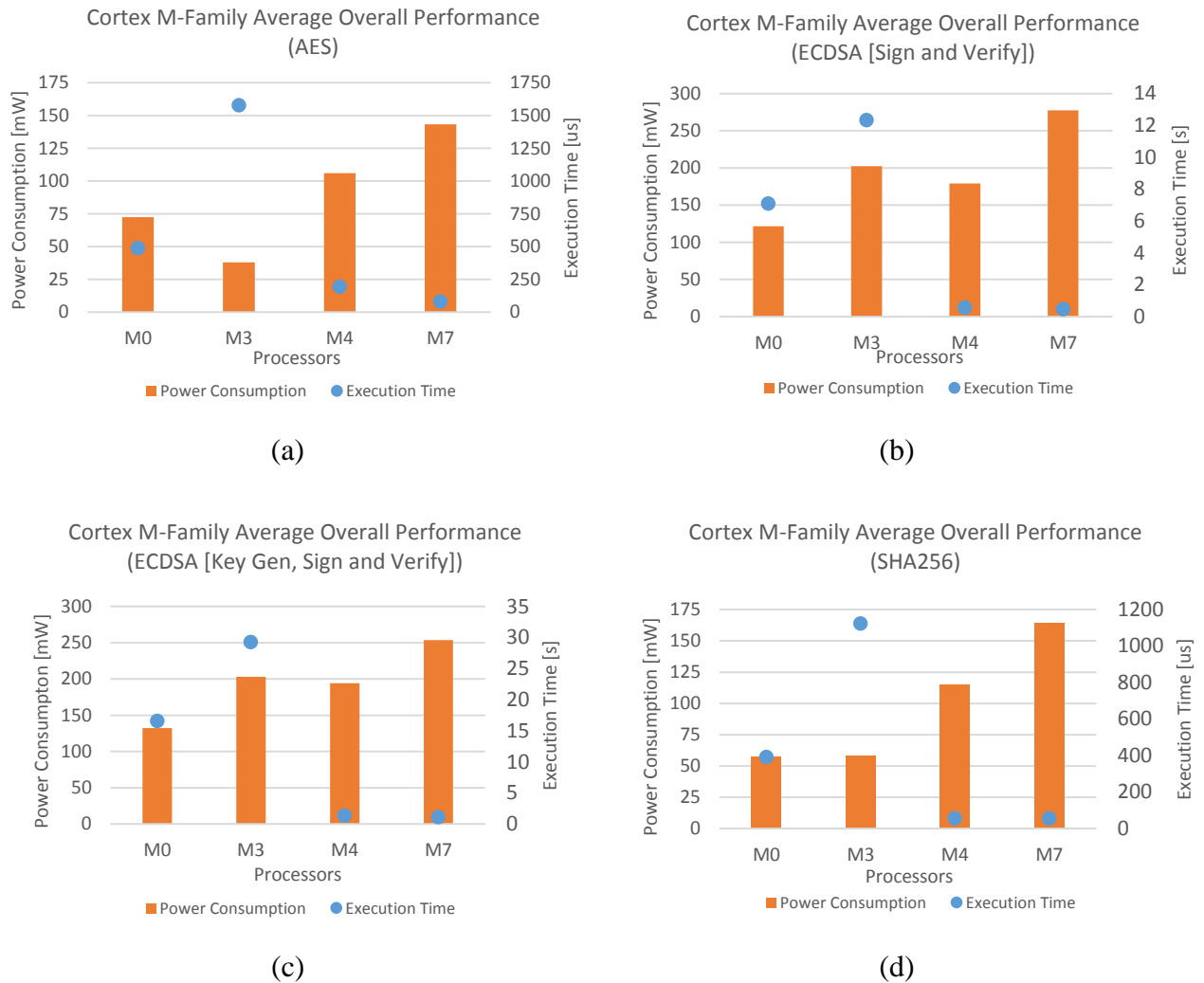


Figure 5.4 Overall performance of cryptographic algorithms on Cortex-M processors

(a) AES128-CTR, (b) ECDSA (Sign-Verify), (c) ECDSA (Key Gen-Sign-Verify) and (d) SHA256

Figures 5.1 and 5.2 showed that the M7’s performances in execution time and power consumption were extreme. It consistently gave the fastest execution time for the four (4) cryptographic algorithms; however, that came at the cost of giving the highest power consumptions. As power consumption was determined for the total execution time, the M7 gave a situation where, when implemented in a secure node, the processor would have a high power draw, but over a very short period of time. It also gave one of the best power consumptions per MHz, over the four algorithms tested. Looking at Figure 5.4, one could see that the combination of the metrics averaged out for the symmetric algorithms;

resulting in the processor producing a performance similar to that of the M4. For public key algorithms, the high power consumption could not be tempered by the fast execution time and good power efficiency; making the M7 the worst overall performing processor for ECDSA. The M3 processor provided another case of extremes as it consistently gave the longest execution time and worst power consumption per MHz of the four (4) processors; however, for the symmetric algorithms, the M3 gave the lowest power consumption. The extremes served to push the overall performance of the M3 up; making it the worst or second to worst performing processor in the execution of the cryptographic algorithms. The M0 and M4 changed in giving the best or second to best overall performances in running the cryptographic library, regardless of the power efficiency of the processor. The M4's best performances were given when running the symmetric algorithms, with a performance close to that given by the M7 processor. The M0 gave the best overall performance when running both versions of ECDSA, giving the second to best performance with the symmetric algorithms, however, as was previously mentioned, a hardware accelerator would need to be employed; as the M0 executed the algorithm within seconds as opposed to milliseconds or microseconds. This could potentially push the power consumption of the resultant node up but, given that the addition of the accelerator was recommended for all the Cortex-M processors, the resultant increased power consumption could still possibly give a better overall performance result than that of the M4 processor.

In addition to determining the best overall performer across the Cortex-M series, a comparison of the performance of the new generation processors against the old generation processors was needed to illustrate the improvements that had been made in IoT processors in the past decade. The results of attempts made at implementing software cryptography on the 8-bit Atmega128L processor, as found on the Mica2, were presented in [36], [37], [38], and [39]; however, of the many algorithms tested, only AES could still be used to secure an industrial network deployment. As a result, the comparison between the old and new generation processors was limited to the results observed for AES. The comparisons for execution time, energy consumption, energy consumption per megaHertz and memory

occupation are given in Figures 5.5 to 5.7. To provide an equivalent dataset for the comparison, the results for AES, as given in the previous works, were averaged and used to calculate an estimated energy consumption, execution time, and memory occupation. The Atmega128L was used to determine the processor's performance in encrypting and decrypting a 64-byte block size, as this was the block size used in STMicroelectronic's AES128 software implementation. The energy consumption of the Cortex-M processors was calculated from the measured power consumptions and execution times using (5.1):

$$\text{Energy (J)} = \text{Power (W)} \times \text{time (s)} \quad (5.1)$$

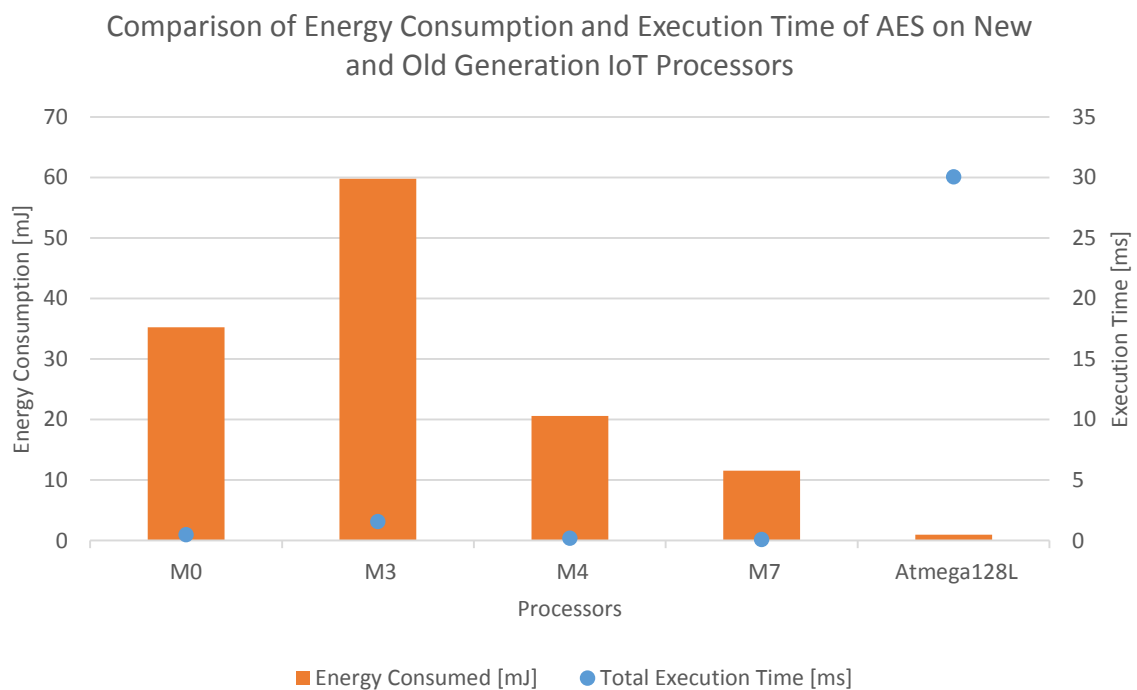


Figure 5.5 Energy consumption and execution time comparison for Atmega128L and Cortex-M processors

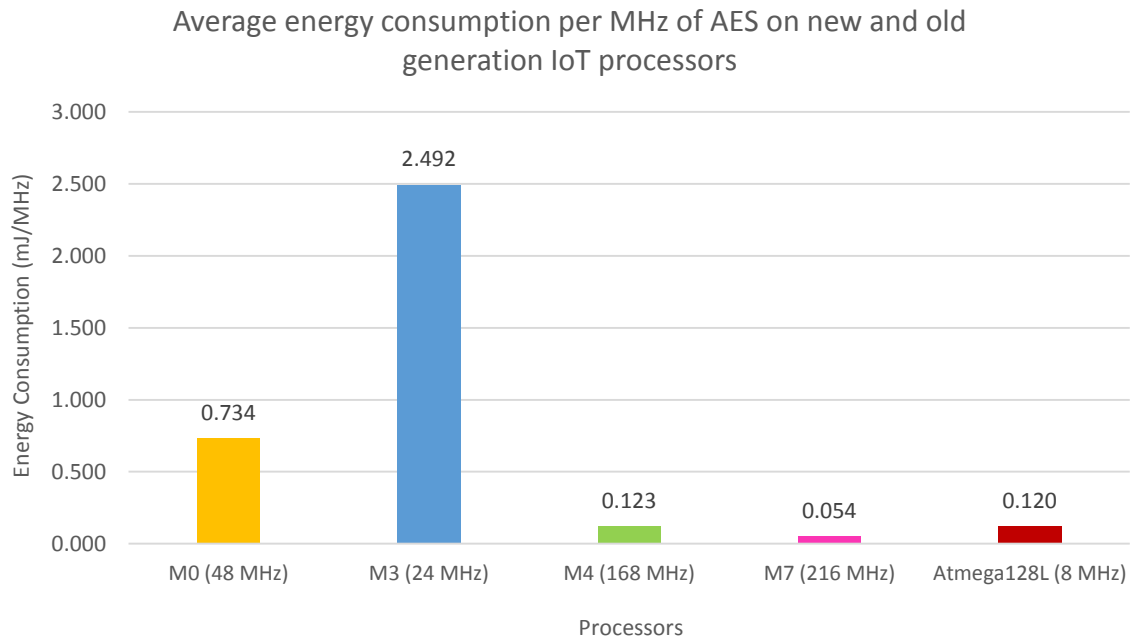


Figure 5.6 Energy consumption per MHz comparison for Atmega128L and Cortex-M processors

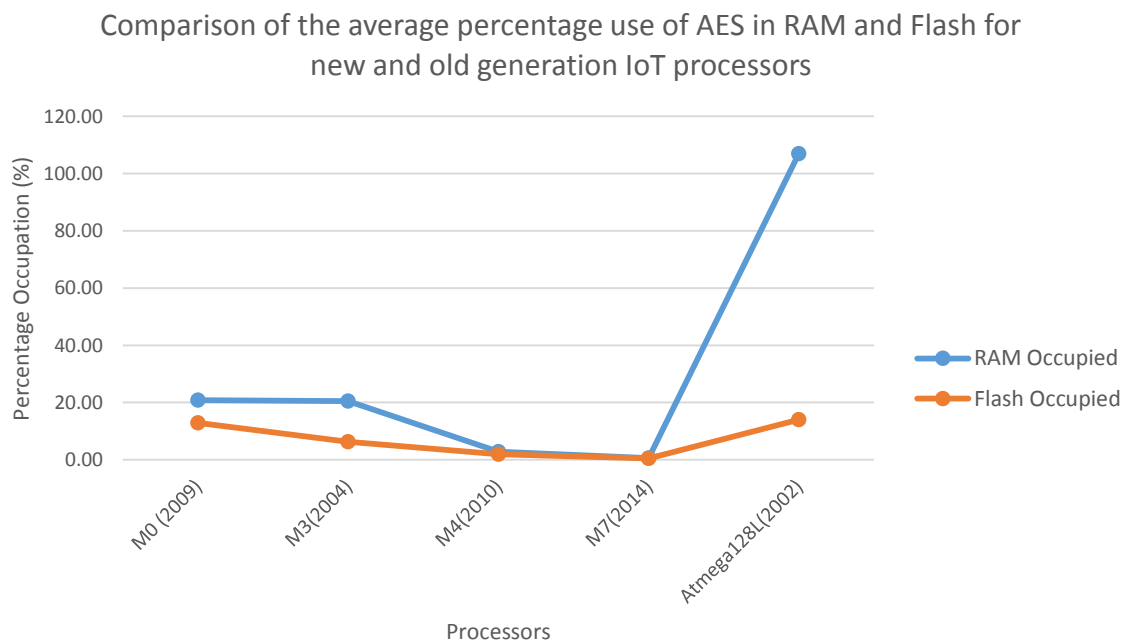


Figure 5.7 Memory occupation comparison for Atmega128L and Cortex-M processors

Comparing the energy consumption performance of the Atmega to those of the Cortex processors, as shown in Figure 5.5, one could see that the 8-bit Atmega gave a very low energy consumption compared to the 32-bit Cortex processors, consuming approximately twelve (12) times less energy than the best performing Cortex processor– the M7– at 0.962mJ. Looking at the overall energy efficiency of the processors in Figure 5.6, however, one could see that the more powerful Cortex-M7 consumed less energy per MHz amongst the processors under consideration – making it more energy efficient than the Atmega processor – while the M4 achieved an energy efficiency similar to that of the Atmega. The execution time of the Atmega was far greater than the execution times seen on any of the Cortex processors, executing AES approximately nineteen (19) times slower than the worst performing Cortex-M processor, with an average execution time of 30ms as compared to 1.579ms from the M3. In comparing the overall performance of the Atmega, one could see that it appeared to give a better performance than the M0 and M3 processors; however, with the length of the execution time and its heavy contribution towards the processor’s performance profile, a hardware accelerator would be required with the Atmega to ensure that the processor was capable of meeting the hard real-time deadline requirements of an IIoT network.

In spite of a fairly promising performance profile, when looking at the memory resource consumption of AES in Figure 5.7, one could see that, for a 64-byte packet, the Atmega would be incapable of running the algorithm as the memory requirement exceeds the available resources of the processor, requiring 4.28kB of RAM where only 4kB of RAM would be available. This complete depletion of the available RAM would mean that, even if the required RAM had been equivalent to the available RAM, the processor would only be able to execute the cryptographic algorithm and could not run other processor operations for the execution and transmission time duration. This is would result in a situation where the entire network essentially would have paused while the endpoint nodes completed their cryptographic operations. In comparison, the Cortex-M processors were capable of running AES efficiently for a 64-byte packet with very little memory resource consumption. The worst case consumptions, seen with the M0 and M3, still left the majority of the RAM resources available for the use of other processor operations.

The consumption of the non-volatile memory resources improved upon the consumption seen for the volatile memory resources. The Atmega gave a similar consumption of non-volatile memory as the Cortex M0, consuming 14.063% of its available ROM as compared to the M0's 12.93% consumption of its available Flash memory. In this respect, the Atmega gave a comparable performance to a processor seven (7) years newer.

Considering the foregoing results presented, despite the areas in which the Atmega was capable of performing as well as or better than the newer Cortex processors, its long execution time and lack of sufficient volatile memory made it unsuitable to run AES cryptographic processes without the inclusion of additional hardware, such as a hardware crypto accelerator and additional RAM. The Cortex processors, on the other hand, were capable of running the AES cryptographic services easily and quickly without additional hardware requirements or without requiring the algorithm to be optimised and scaled down to fit within their available memory. Their weakest point was in the processor energy consumption; where the Atmega was shown to be vastly superior. This, however, may be seen as a sufficient trade-off for the ability to be able to implement upgradable software cryptography services at the edge of the IIoT network without requiring additional hardware components and upgrades.

5.3 TOWARDS DESIGNING A SECURE MOTE FOR THE IIOT

Considering the performances seen for the four (4) processors, the M0 and M4 gave the best performances in running the cryptographic algorithms. A closer look at the performance profiles needed to be taken to identify the best, overall performer from the Cortex-M processors. Having removed the results for the M3 and the M7, Figure 5.8 gives a comparison of the performance profiles for the M0 and the M4.

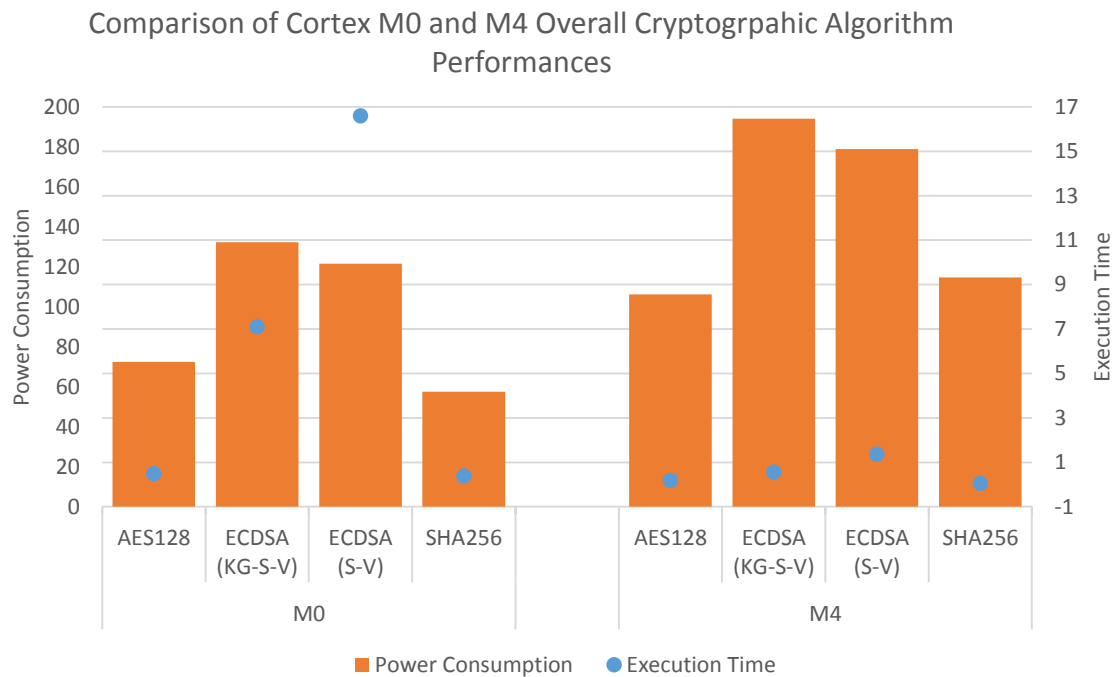


Figure 5.8 Energy consumption and execution time comparison for Cortex M0 and M4 processors

Comparing the performances of the M0 and M4, one could see that, even though the M0 performed better with the public key cryptographic algorithm, the jumps seen in its performance between the public key and symmetric algorithms were much larger than the jumps seen in the M4's performance with the public key and symmetric algorithms, highlighting the M4 as the more stable and consistent processor. Although the M4's main weakness was in its power consumption, as a processor for an edge node, it would spend approximately 99% of its time within sleep mode, where its power consumption would be minimal. The shorter execution times could also result in longer time periods that the processor is able to spend in sleep mode; as it would be capable of finishing the work required in active mode quickly and return to a low power consumption state faster than the M0 processor.

Although the M4 gave the best overall performance profile for use at the edge, depending on the requirements of the IIoT network, any one of the Cortex-M series processors would be suitable for use in a secure mote design. The versions tested within this work were

found to be capable of running software-implemented cryptography quickly at reasonably low power consumptions without the depletion of the processor resources, as opposed to the older Atmega128L which would require additional hardware resources to be able to support standard cryptographic operations for large packet sizes. In this respect, having seen the progress made over the last decade towards improving processors for the IoT, it would no longer be accurate to state that low power processors designed for the IoT and WSNs are incapable of supporting cryptographic processes, more specifically software cryptographic processes. This study has shown that even the oldest of the Cortex-M processors has been capable of running software-implemented cryptography without the depletion of its available resources. One may also find alternative versions of the processors tested within this study from different vendors who may produce designs which are faster, more power-efficient or which may be implemented with additional security features.

With regard to designing a secure endpoint device; depending on the application area, the expected lifetime of the network or the rate of deprecation of cryptographic algorithms for industrial use, verified software cryptography libraries may be a viable solution in addition to hardware cryptographic components. In cases where the security provided by the hardware components was compromised, one would be able to conduct update procedures that switch to the use of a software cryptography library to preserve the secure network state. In such cases, adjustments would need to be made within the network to compensate for the introduction of a new delay and the possible increase in power consumption. In other cases, where the size of the endpoint node may need to be minimised and if the inclusion of cryptography hardware were infeasible, software libraries could be used to provide cryptographic services on the endpoint as opposed to leaving them unprotected. Also, with the inclusion of TrustZone capability on the new M23 and M33 processors, software cryptographic operations could be conducted with the added security of having originated from and being processed within an isolated, trusted space.

5.4 CHAPTER SUMMARY

Previously, owing to the limited resources available, processors for WSNs and the IoT were left insecure and without security processes, resulting in large network deployments that were vulnerable to a wide variety of cyber-physical attacks. It could be seen, however, that with the improvements made in new generation processors over the last decade, low power processors designed for use with the IoT were easily able to run cryptographic operations quickly without the depletion of memory and power resources. This showed that processors for the IoT were no longer incapable of implementing security processes and that software cryptographic libraries could be a viable resource in designing a secure, endpoint node; either as a tool to extend the effective lifetime of a network deployment or as a security tool on nodes with very tight size restrictions.

In Section 5.1 the chapter objectives and overview were provided.

In Section 5.2 a detailed discussion on the performance of the processors running software-implemented cryptography was given; comparing the performances seen between the four (4) processors and comparing the performances of the new-generation Cortex-M processors against the performances seen in the older-generation Atmega128L.

In Section 5.3 the best performing of the four Cortex-M processors was selected based on the overall performance, as determined using the three (3) metrics tested, in running the cryptographic algorithms.

The following chapter serves to conclude this study and provides a summary of the research question and objectives posed, as well as the extent to which these have been addressed by this study and to highlight areas in the topic where additional work will be conducted in the future.

CHAPTER 6 CONCLUSION

Chapter 6 serves to summarise the ideas, results and recommendations presented as part of this research. The author presents the conclusions in terms of the original research objectives and in terms of the contributions made to the field of security for the IIoT. Future research directions regarding the incorporation of the research conducted into the broader IIoT security field are briefly highlighted, and additional fields in which the author intends to conduct further work are identified.

6.1 SUMMARY OF CONCLUSIONS

Prior to commencing this research, the author posed the problem asking whether cryptographic algorithms capable of providing encryption and decryption services could be implemented on an IIoT endpoint device without resulting in significant losses in device performance and longevity. Specifically, the research aimed to answer the following:

- What are the time costs associated with the application of encryption/decryption services on low power devices?
- What are the associated power consumption and memory utilisation costs for applying encryption/decryption services on IIoT endpoint devices?

The research objectives for this work were as follows:

1. Identify the security requirements of IIoT endpoint nodes.
2. Identify general purpose and security enabled new-generation IIoT platforms.
3. Identify open source, standard cryptographic algorithms best suited for application onto an IIoT endpoint node.

4. Determine possible performance trade-offs – e.g. power, memory, throughput, or cost – in applying cryptographic techniques, such as encryption and decryption, on an IIoT node.
5. Determine the best-suited cryptography scheme for securing a low power, IIoT node.
6. Identify the best method by which to integrate cryptography services as part of the construction of a secure IIoT mote.

In meeting objectives one (1) and two (2), a detailed literature review was conducted in Chapter 2; focusing on the identification of security standards for the IIoT and the relevant recommendations made for the security of an IIoT endpoint device. Technologies currently available for the IIoT and their associated trade-offs were highlighted alongside a lack of security application technologies able to implement and provide security attestation and isolation within an IIoT network. Work conducted towards the end of Chapter 2 and in Chapter 3 served towards meeting objective three (3). Table 2.12 introduced security enabled MCUs for the IIoT, highlighting the features provided on the MCUs and the areas still lacking in the full realisation of the security recommendations made by the IIC and OpenFog Consortium. Chapter 3 also served to detail the experimental setup and procedures used in the completion of this research; introducing the general purpose Cortex-M series processors and the STMicroelectronics cryptographic library. As the Cortex-M processors are commonly used for IoT applications and were without embedded security features, they made good candidates for testing the capabilities of new generation processors in running software cryptographic services. The detailed experimental procedures used for the measurement of the execution time of the algorithms, memory occupation and power consumptions were given; with relevant images showing an instance of one of the multiple results seen for each experiment. Chapters 4 and 5 served towards meeting objectives four (4) through six (6). Chapter 4 reported the results seen from the experiments conducted in Chapter 3, giving average estimations for execution time, memory occupation and power consumption. An estimate for the deviation that could be expected in each of the algorithms and processors was provided for each measurement. An in-depth analysis conducted in Chapter 5 illustrated the performance differences seen

across the M-series processors as they ran the four (4) cryptographic algorithms used in this work. It was seen that, despite having identified the Cortex M4 as the best-suited general purpose processors for running software cryptographic services, the new generation processors chosen were capable of running the algorithms without the depletion of their memory resources and excessive power consumption. It was also seen that the use of a hardware cryptographic module could be required for the Cortex-M series should an implementation of public key cryptography be needed in an IIoT network; as the resulting execution times for ECDSA were sufficiently long as to increase the probability of missed deadlines in a hard real-time network application. A comparison was made of the Cortex-M performance results seen in this research with the performance results seen on previous applications of AES on the Mica2 platform. This served to illustrate the improvements made in the capabilities of MCU platforms for the IoT over the past decade. At the conclusion of Chapter 5, it was noted that, dependent on the specific network application requirements, verified software cryptographic services were a more than adequate option to provide a security base when used on new generation IoT processors; allowing for the extension in the security lifetime of long-term node deployments or an alternative to the inclusion of hardware cryptographic modules when securing size-constrained edge mote designs.

6.2 RESEARCH CONTRIBUTION

By conducting this research, it was found that new generation IoT/WSN processors are more than capable of implementing software cryptographic services. efficiently Thus, the previous rhetoric that IoT devices are incapable of running cryptographic solutions is no longer universally true of technologies intended for the IoT and WSNs. This has various implications in how security in the IIoT would be established and regarding the expectations that could be made of IIoT network applications, some of which are briefly highlighted.

It was important to update the IoT/WSN knowledge to reflect the current state of the available technology as there is a fast-growing need for the adoption of security policies, standards and frameworks that protect the availability, integrity and confidentiality of IoT applications. The physical isolation of devices from Internet-capable areas of the network is no longer adequate as a protection strategy for industrial applications. By illustrating the capability of new generation processors, the inability to support cryptographic services can no longer be used as an excuse for the non-implementation of security services for IoT devices. Combining the updated viewpoint into a standard for device security would aid in establishing culpability and consequences for non-compliance at both a vendor and network design level for future IIoT applications that are found to be insecure. Hopefully, this will result in more secure IIoT networks and will pre-empt a large, dangerous security failure in current and future Industrie 4.0 deployments.

The improved capability seen in the processors means that verified software libraries can be utilised as tools to extend the secure lifetime of a network. This would result in shorter, zero-day exploitations and the introduction of multiple, overlapping points of redundancy in an IIoT network security scheme.

Finally, the basic security requirements for the IIoT can include the implementation of cryptographic services as mandatory in order to preserve integrity and confidentiality in the network devices. Software libraries can be used as an alternative solution to hardware acceleration devices providing symmetric cryptography in cases where the size of the mote device needs to be minimised. The combination of hardware and software solutions could greatly increase the number of cryptographic algorithms provided and supported by IIoT edge nodes; again providing multiple points of redundancy within a security scheme and improving the interoperability of devices without resulting in a compromise in network security.

6.3 FUTURE WORK

In the author's opinion, this research has met with all the stated objectives and has answered adequately the questions identified in the research problem statement. The research has provided a detailed analysis regarding the capabilities of new generation IoT processors when running software cryptographic services and has found that symmetric and hashing cryptographic services can be implemented on the processors with minimal costs to the processor performance. The inclusion of a hardware accelerator is recommended for the implementation of public key cryptographic services owing to the long execution times seen for the processors.

After concluding this research, it is the author's opinion that verified software cryptographic services could be used as a viable option towards securing new generation processors where the inclusion of hardware services may not be viable or where the network security may have been compromised.

As part of future studies, the author aims to continue work towards a general, implementable design for a secure endpoint device at the IIoT edge. In particular, the author aims to expand upon the research conducted within this study and to compare the performances of the software-implemented cryptography algorithms to their hardware-implemented counterparts; to revisit the experiments conducted in this study having expanded the list of cryptographic algorithms under consideration; to test the performance of the Cortex-M processors when running the algorithms used within this work with longer key expansions, and to test for the performance differences that could be seen when a TRNG is used for key generation as opposed to a PRNG. Finally, the author aims to conduct an evaluation comparing the performance of software-secured MCUs to the performance of security-enabled MCUs to the performance of a softcore secured FPGA/MCU hybrid platform in order to determine which configuration would provide the fastest, least power-intensive security services for the IIoT edge while maintaining good longevity and maintainability.

REFERENCES

- [1] Industrial Internet Consortium. (2015). *Industrial Internet Reference Architecture* [Online]. Available: <http://www.iiconsortium.org/IIRA-1-7-ajs.pdf> [Accessed: Jan. 30, 2017].
- [2] Industrial Internet Consortium. (2016). *Industrial Internet Security Framework Volume G4* [Online]. Available: http://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB-3.pdf [Accessed: Jan. 30, 2017].
- [3] A.R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," *Proc. of 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, 2015, pp. 1-6.
- [4] Unknown. (2011). *An Introduction to Cyber-Physical Systems* [Online]. Available: http://www.uio.no/studier/emner/matnat/ifi/INF5910CPS/h11/undervisningsmateriale/20110830_CPS-WSN-Overview.pdf [Accessed: Jan. 30, 2017].
- [5] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," *Proc. of Design Automation Conference*, Anaheim, 2010, pp. 731-736.
- [6] D. Gollmann and M. Krotofil, "Cyber Physical System Security," in *The New Codebreakers*, 1st ed., P. Ryan, D. Naccache and J. J. Quisquater, Ed. Berlin, Heidelberg: Springer, 2016, pp. 195-204.
- [7] *Security Requirements for Cryptographic Modules*, FIPS Standard 140-2, 2001.
- [8] OpenFog Consortium. (2017). *OpenFog Reference Architecture for Fog Computing* [Online]. Available: https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf [Accessed: Jul. 5, 2017].
- [9] B. Nisarga and E. Peeters. (2016). *System-Level Tamper Protection Using MSP MCUs* [Online]. Available: <http://www.ti.com/lit/an/slaa715/slaa715.pdf> [Accessed: May 11, 2017].

REFERENCES

- [10] Y.M. Yussoff, H. Hashim, R. Rosli, and M.D. Baba, "A Review of Physical Attacks and Trusted Platforms in Wireless Sensor Networks," *Procedia Engineering*, vol. 41, pp.580-587, Jan. 2012.
- [11] Digi-Key European Editors. (2015). *Developing Anti-Tamper Protection for Wireless Hardware* [Online]. Available: <https://www.digikey.com/en/articles/techzone/2015/may/developing-anti-tamper-protection-for-wireless-hardware> [Accessed: May 12, 2017].
- [12] S. Skorobogatov, "Physical Attacks and Tamper Resistance," in *Introduction to Hardware Security and Trust*, 1st ed., M. Tehranipoor and Cliff Wang, Ed. New York, NY: Springer Verlag, 2012, pp. 143-173.
- [13] Rambus Incorporated. (2017). *Security Licensed Countermeasures* [Online]. Available: <https://www.rambus.com/security/dpa-countermeasures/licensed-countermeasures/> [Accessed: Jun. 6, 2017].
- [14] Rambus Incorporated. (2017). *DPA Countermeasures Validation Program* [Online]. Available: <https://www.rambus.com/security/dpa-countermeasures/dpa-countermeasures-validation-program/> [Accessed: Jun. 6, 2017].
- [15] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," *Proc. of IEEE Trustcom/BigDataSE/ISPA*, Helsinki, 2015, pp. 57-64.
- [16] J. Attridge. (2002). *An Overview of Hardware Security Modules* [Online]. Available: <https://www.sans.org/reading-room/whitepapers/vpns/overview-hardware-security-modules-757> [Accessed: May 11, 2017].
- [17] Trusted Computing Group. (2008). *Trusted Platform Module (TPM) Summary* [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-Summary_04292008.pdf [Accessed: May 11, 2017].
- [18] Atmel Corporation. (2014). *Atmel Trusted Platform Module AT97SC3204 / AT97SC3205* [Online]. Available: <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp2014.pdf> [Accessed: May 11, 2017].
- [19] Infineon Technologies. (2015). *Trusted Platform Module TPM SLB 9665 TCG Family 2 Level 00 Rev. 01.16 Datasheet* [Online]. Available: http://www.infineon.com/dgdl/Infineon-TPM+SLB+9665-DS-v10_15-EN.pdf?fileId=5546d4625185e0e201518b83d9273d87 [Accessed: May 11, 2017].
- [20] STMicroelectronics. (2013). *ST33TPM12LPC - Trusted Platform Module with LPC interface based on 32-bit ARM SecurCore SC300 CPU* [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/data_brief/6b/d2/76/50/a4/46/

REFERENCES

<44/86/DM00037936.pdf/files/DM00037936.pdf/jcr:content/translations/en.DM00037936.pdf> [Accessed: May 11, 2017].

[21] SANS ISO/IEC 24760-1:2011: *Information technology. Security techniques. A framework for identity management. Terminology and concepts*, ISO/IEC 24760-1-2011, 2011.

[22] Plattform Industrie4.0. (2016). *Technical Overview: Secure Identities* [Online]. Available: https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/secure-identities.pdf?__blob=publicationFile&v=7 [Accessed: Jan. 30, 2017].

[23] Liwenhao. (2014). *A Technical Report on TEE and ARM TrustZone* [Online]. Available: <https://community.arm.com/processors/b/blog/posts/a-technical-report-on-tee-and-arm-trustzone> [Accessed: May 11, 2017].

[24] S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares, "IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices," *IEEE Internet Computing*, vol. 21, no. 1, pp.40-47, Jan. 2017.

[25] N. Asokan, F. Brassler, A. Ibrahim, A. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable Embedded Device Attestation," *Proc. of 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, 2015, pp. 964–975.

[26] J. Valente, C. Barreto, and A.A. Cárdenas, "Cyber-Physical Systems Attestation," *Proc. of IEEE International Conference on Distributed Computing in Sensor Systems*, Marina Del Rey, 2014, pp. 354-357.

[27] A. Fongen and F. Mancini, "Integrity attestation in military IoT," *Proc. of IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, 2015, pp. 484-489.

[28] A. Ibrahim, A. Sadeghi, G. Tsudik, and S. Zeitouni, "DARPA: Device Attestation Resilient to Physical Attacks," *Proc. of 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, Darmstadt, 2016, pp. 171–182.

[29] A. Seshadri, A. Perrig, L.v. Doorn, and P. Khosla, "SWATT: softWare-based attestation for embedded devices," *Proc. of IEEE Symposium on Security and Privacy*, Berkeley, 2004, pp. 272-282.

[30] K. El Defrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART: Secure and minimal architecture for (establishing a dynamic) root of trust," *Proc. of 19th Annual Network and Distributed Systems Security Symposium*, San Diego, 2012, pp. 1-15.

[31] P. Koeberl, S. Schulz, A. Sadeghi, and V. Varadharajan, "TrustLite: A Security Architecture for Tiny Embedded Devices," *Proc. of 9th European Conference on Computer Systems*, Amsterdam, 2014, pp. 10:1–10:14.

REFERENCES

- [32] Moteiv Corporation. (2004). *Telos- Ultra low power IEEE 802.15.4 compliant wireless sensor module Revision B : Humidity, Light, and Temperature sensors with USB* [Online]. Available: <http://www4.ncsu.edu/~kkolla/CSC714/datasheet.pdf> [Accessed: May 11, 2017].
- [33] Crossbow Technology Incorporated. (2004). *Mica2: Wireless Measurement System* [Online]. Available: <https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf> [Accessed: May 11, 2017].
- [34] Crossbow Technology Incorporated. (2004). *MicaZ: Wireless Measurement System* [Online]. Available: http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf [Accessed: May 11, 2017].
- [35] Ember Corporation. (2004). *EM2420: 2.4 GHz IEEE 802.15.4 / ZigBee RF Transceiver* [Online]. Available: <http://media.digikey.com/pdf/Data%20Sheets/Ember%20PDF%27s/EM2420.pdf> [Accessed: May 11, 2017].
- [36] C.P. Antonopoulos, C. Petropoulos, K. Antonopoulos, V. Triantafyllou, and N.S. Voros, "The effect of symmetric block ciphers on WSN performance and behaviour," *Proc. of IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Barcelona, 2012, pp. 799-806.
- [37] C.C. Chang, S. Muftic, and D.J. Nagel, "Measurement of Energy Costs of Security in Wireless Sensor Nodes," *Proc. of 16th International Conference on Computer Communications and Networks*, Honolulu, 2007, pp. 95-102.
- [38] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, "Evaluation of security mechanisms in wireless sensor networks," *Proc. of Systems Communications (ICW'05, ICHSN'05, ICMCS'05, SENET'05)*, Montreal, 2005, pp. 428-433.
- [39] A. Trad, A.A. Bahattab, and S.B. Othman, "Performance trade-offs of encryption algorithms for Wireless Sensor Networks," *Proc. of World Congress on Computer Applications and Information Systems (WCCAIS)*, Hammamet, 2014, pp. 1-6.
- [40] R.L. Rivest, "The RC5 Encryption Algorithm," in *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings*, B. Preneel. , Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 86-96.
- [41] R.L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin. (1998). *The RC6 Block Cipher* [Online]. Available: <https://people.csail.mit.edu/rivest/pubs/RRSY98.pdf> [Accessed: Sep. 5, 2015].
- [42] *Advanced Encryption Standard (AES)*, FIPS Standard 197, 2001.

REFERENCES

- [43] *Data Encryption Standard (DES)*, FIPS Standard 46-3, 1999.
- [44] *SkipJack and KEA Algorithm Specification*, FIPS Standard 185, 1998.
- [45] *Secure Hash Standard (SHS)*, FIPS Standard 180-4, 2015.
- [46] D.J. Wheeler and R.M. Needham, "TEA, a tiny encryption algorithm," *Proc. of Fast Software Encryption*, Berlin, 1994, pp. 363-366.
- [47] E. Barker. (2016). *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms* [Online]. Available: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175B.pdf> [Accessed: Sep. 4, 2017].
- [48] EMC Corporation. (2017). *RSA Laboratories - RC6® Block Cipher* [Online]. Available: <https://www.emc.com/emc-plus/rsa-labs/historical/rc6-block-cipher.htm> [Accessed: Sep, 4, 2017].
- [49] Microsemi Corporation. (2017). *SmartFusion2 SoC FPGA Family* [Online]. Available: <https://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2#overview> [Accessed: May 11, 2017].
- [50] NXP Semiconductors. (2017). *A7101CGT1: Secure authentication microcontroller* [Online]. Available: <http://www.nxp.com/products/identification-and-security/secure-authentication-and-anti-counterfeit-technology/secure-authentication-microcontroller:A7101CGT1> [Accessed: May 11, 2017].
- [51] Maxim Integrated. (2017). *MAXQ1852 DeepCover Secure Microcontroller with Fast Wipe Technology and Cryptography* [Online]. Available: <https://www.maximintegrated.com/en/products/digital/microcontrollers/MAXQ1852.html> [Accessed: May 11, 2017].
- [52] Texas Instruments Incorporated. (2016). *ARM Cortex-M4F Microcontroller - Overview - TM4C12x* [Online]. Available: <http://www.ti.com/lscds/ti/microcontrollers-16-bit-32-bit/c2000-performance/control-automation/tm4c12x/overview.page> [Accessed: May 11, 2017].
- [53] Maxim Integrated. (2017). *MAXQ1050 DeepCover Secure Microcontroller with USB and Hardware Cryptography* [Online]. Available: <https://www.maximintegrated.com/en/products/digital/microcontrollers/MAXQ1050.html> [Accessed: May 11, 2017].
- [54] Maxim Integrated. (2017). *MAXQ1061 DeepCover Cryptographic Controller for Embedded Devices* [Online]. Available: <https://www.maximintegrated.com/en/products/digital/microcontrollers/MAXQ1061.html> [Accessed: May 11, 2017].

REFERENCES

- [55] STMicroelectronics. (2017). *STSAFE-A100 - Authentication and Brand protection secure solution* [Online]. Available: http://www.st.com/content/st_com/en/products/secure-mcus/authentication-secure-iot/stsafe-a100.html [Accessed: May 11, 2017].
- [56] STMicroelectronics. (2017). *STSAFE-J100 - Flexible secure solution for Gateway* [Online]. Available: http://www.st.com/content/st_com/en/products/secure-mcus/authentication-secure-iot/stsafe-j100.html [Accessed: May 11, 2017].
- [57] XILINX INC. (2017). *Zynq UltraScale+ MPSoC* [Online]. Available: <https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html> [Accessed: May 11, 2017].
- [58] Texas Instruments Incorporated. (2016). *AM335x | Cortex-A8 | Overview | Sitara* [Online]. Available: http://www.ti.com/lscds/ti/processors/sitara/arm_cortex-a8/am335x/overview.page [Accessed: May 11, 2017].
- [59] Microchip Technology Inc. (2017). *Hardware Crypto Engine - Technology / Embedded Security* [Online]. Available: <http://www.microchip.com/design-centers/embedded-security/reference-designs/hardware-crypto-engine> [Accessed: May 11, 2017].
- [60] STMicroelectronics. (2017). *X-CUBE-CRYPTOLIB - STM32 cryptographic firmware library software expansion for STM32Cube (UM1924)* [Online]. Available: <http://www.st.com/en/embedded-software/x-cube-cryptolib.html> [Accessed: Sep. 5, 2017].
- [61] Texas Instruments. (2017). *CRYPTO Cryptography for TI Devices* [Online]. Available: <http://www.ti.com/tool/CRYPTO> [Accessed: Sep. 8, 2017].
- [62] *Digital Signature Standard (DSS)*, FIPS Standard 186-4, 2013.
- [63] Arm Limited. (2017). *Cortex-M* [Online]. Available: <https://www.arm.com/products/processors/cortex-m> [Accessed: Aug. 30, 2017].
- [64] Arm Limited. (2017). *Cortex-M0* [Online]. Available: <https://developer.arm.com/products/processors/cortex-m/cortex-m0> [Accessed: Aug. 30, 2017].
- [65] Arm Limited. (2017). *Cortex-M3* [Online]. Available: <https://developer.arm.com/products/processors/cortex-m/cortex-m3> [Accessed: Aug. 30, 2017].
- [66] Arm Limited. (2017). *Cortex-M4* [Online]. Available: <https://developer.arm.com/products/processors/cortex-m/cortex-m4> [Accessed: Aug. 30, 2017].

REFERENCES

- [67] Arm Limited. (2017). *Cortex-M7* [Online]. Available: <https://developer.arm.com/products/processors/cortex-m/cortex-m7> [Accessed: Aug. 30, 2017].
- [68] STMicroelectronics. (2017). *STM32F0 - ARM Cortex-M0 Microcontrollers* [Online]. Available: <http://www.st.com/en/microcontrollers/stm32f0-series.html?querycriteria=productId=SS1574> [Accessed: Sep. 4, 2017].
- [69] STMicroelectronics. (2017). *STM32F1 - ARM Cortex-M3 Microcontrollers* [Online]. Available: <http://www.st.com/en/microcontrollers/stm32f1-series.html?querycriteria=productId=SS1031> [Accessed: Sep. 4, 2017].
- [70] STMicroelectronics. (2017). *STM32F4 - ARM Cortex-M4 High-Performance MCUs* [Online]. Available: <http://www.st.com/en/microcontrollers/stm32f4-series.html?querycriteria=productId=SS1577> [Accessed: Sep.4, 2017].
- [71] STMicroelectronics. (2017). *STM32F7 - ARM Cortex-M7 Microcontrollers* [Online]. Available: <http://www.st.com/en/microcontrollers/stm32f7-series.html?querycriteria=productId=SS1858> [Accessed: Sep. 4, 2017].
- [72] I. Johnson. (2015). *ARM's Cortex-M and Cortex-R Embedded Processors* [Online]. Available: https://www.arm.com/zh/files/event/2_2015_ARM_Embedded_Seminar_Ian_Johnson.pdf [Accessed: Aug. 30, 2017].
- [73] STMicroelectronics. (2017). *STM32CubeMX - STM32Cube initialization code generator* [Online]. Available: <http://www.st.com/en/development-tools/stm32cubemx.html> [Accessed: Sep. 5, 2017].
- [74] S. Keller. (2009). *Cryptographic Algorithm Validation Program* [Online]. Available: <https://www.nist.gov/programs-projects/cryptographic-algorithm-validation-program> [Accessed: Sep. 5, 2017].
- [75] E. Granados. (2017). *One way to use the X-CUBE-CRYPTOLIB with CubeMX* [Online]. Available: <https://community.st.com/docs/DOC-1557-one-way-to-use-the-x-cube-cryptolib-with-cubemx-and-system-workbench> [Accessed: Sep. 5, 2017].
- [76] Atollic. (2017). *TrueSTUDIO* [Online]. Available: <https://atollic.com/truestudio/> [Accessed: Sep. 4, 2017].
- [77] M. Mielke. (2017). *Using the IDD Current Measurement Feature on the STM32L053 Discovery Board* [Online]. Available: <https://eewiki.net/display/microcontroller/Using+the+IDD+Current+Measurement+Feature+on+the+STM32L053+Discovery+Board> [Accessed: Sep. 5, 2015].
- [78] Atollic. (2017). *TrueSTUDIO Pro - ARM Development Tools - Subscription FAQ* [Online]. Available: <http://info.atollic.com/pro-upgrade-faq> [Accessed: Sep. 4, 2017].