IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 20, 2017, accepted December 23, 2017, date of publication January 15, 2018, date of current version March 13, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2793301

Performance Costs of Software Cryptography in Securing New-Generation Internet of Energy Endpoint Devices

LEHLOGONOLO P. I. LEDWABA[1,2], GERHARD P. HANCKE[2,3], (Senior Member, IEEE), HEIN S. VENTER[4], (Member, IEEE), AND SHERRIN J. ISAAC[1], (Member, IEEE)

[1]Meraka Institute, Council for Scientific and Industrial Research, Pretoria 0001, South Africa
[2]Departments of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa
[3]Department of Computer Science, City University of Hong Kong, Hong Kong
[4]Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa

Corresponding author: Lehlogonolo P. I. Ledwaba (lledwaba@csir.co.za)

This work was supported in part by the Council for Scientific and Industrial Research under Project KR7SSMC and in part by the South African Mineral Extraction Research Development and Innovation Strategy Programme's thematic area of Real Time Information Management Systems.

ABSTRACT In past years, cryptography has been considered a difficult task to achieve on sensor nodes for the Internet of Energy (IoE) owing to the resource-constrained nature of 8- and 16-bit microcontroller units (MCUs). Previous attempts at implementing cryptographic services on wireless sensor nodes have resulted in high power consumptions, long operating times, and the depletion of memory resources. Over the last decade, however, processors for the IoT and IoE have improved; with increased operating power and memory resources, longer data bus widths and low-power consumption. With the improvements made to processors suitable for building IoT devices, the question remains whether endpoint nodes should still be considered capable of only supporting the most lightweight of cryptographic mechanisms. We evaluate the capabilities of a device family (Cortex-M series processors) commonly found in programmable logic controllers to implement standard, verified software cryptographic libraries in terms of execution times, memory occupation, and power consumption in order to determine their adequacy for use in smart grid applications. It was seen that the MCUs were easily capable of running standard cryptographic algorithms. However, the use of public key cryptography may still require the inclusion of a hardware crypto accelerator or the use of a secure MCU implementing public key cryptography; as the relatively long execution times seen during the operation of ECDSA, for example, could be intolerable within a real time IoE application.

INDEX TERMS Industrial Internet of Things, Internet of Energy, Embedded Security, Smart Grid.

I. INTRODUCTION

Smart grid applications come as a response to changing market structures and the increasing need for new energy sources and more efficient energy distribution structures. The electrical grid in its current incarnation is unable to cope with the rising demands for electrical energy as a result of the growing populations, the decrease in conventional energy sources and the inefficiencies in distribution systems which result high losses of energy in the form of heat dissipation [1], [2]. The Internet of Energy (IoE) is intended to allow energy utilities to better meet the rising demand for energy resources, improved reliability in distributed resources and lower carbon emissions from generation processes [3]. The introduction of ICT systems allow for various, distributed energy sources to be integrated into a network configuration similar to the Internet in order to provide high grid flexibility, energy packaging and routing functions on a wider scale of distribution [1]. The main aims of smart grid applications is to reduce the capital expenditure of energy utilities by allowing for precise matching of supply and demand across the energy grid, better manage demand on the grid through the analysis of consumption patterns and the promotion of energy conservation initiative, increase renewable capacity with existing grid generation structures and technologies, lower maintenance costs through remote fault diagnosis and analysis of the operational activity of generation, transmission and distribution assets,

promote better compliance with carbon emission regulations and enable better customer engagement through the sharing of predicted load patterns and pricing schemes at peak and non-peak hours [3].

Currently, the electrical energy supply chain consists of four main activities: generation, transmission, distribution and consumption [4]. Generation looks at processes that result in the production of electrical energy, transmission looks at processes which distribute electrical resources from the generation point, over long distances to distribution points where electrical resources are then provided to end-users who then consume the resource and become a load on the electrical grid [4]. Billing in the current incarnation of the electrical grid is typically conducted and forwarded to consumers on a monthly basis [1].

The IoE results in the incorporation of cyber processes into the generation, transmission, distribution and consumption of electrical resources and the accommodation of multiple forms of energy generation, integrate energy and information exchange into a single infrastructure based on an internet-enabled energy router paradigm, and achieve balance between supply and demand using Internet, distributed intelligence and big data applications [1], [4].

Within these processes, as renewable generation sources become more prevalent in the grid space, smart grids aim to provide a variety of services which improve the quality of service received from energy utilities such as allowing for a bidirectional energy chain with provision for the sale of excess energy back into the main grid, advanced metering, dynamic pricing, fault detection and diagnostics and remote monitoring and control over transmission systems, and predictive maintenance and self-healing in the event of failure conditions [2].

In general, a smart grid network may consist of multiple domains from which data can be collected and aggregated. At the consumer end, smart meters are utilised to collect real-time consumption information [5], [6]. Total energy consumption data for residential neighbourhoods are gathered by a collector device for each neighbourhood which then distributes the information to the distribution and operation centres through the use of a wireless mesh network [5], [6]. Similar operations are conducted for commercial and industrial consumers [5]. All collector devices generate a separate network and forward the load information to the central energy utility units where grid load analysis, energy transmission and billing is conducted [5], [6]. Utilities can also forward grid load forecasts, pricing for peak load hours and general control commands back to smart grid devices which would then forward the relevant information on to the end users in order to allow them to adjust their energy consumption in accordance to peak load times and prices [5], [6]. The numerous domains from which data is collected in the smart grid brings value into the customer profiling, demand response, network planning and pricing profiles that can be conducted by utilities and energy providers [5]. Wireless technologies are often chosen for data gathering owing to their high data rates and reliability while big data computing mechanisms are required to transform the vast amounts of gathered data into useful information [5].

Data gathered in a smart grid application typically exhibits unique characteristics in terms of the volume of data, the velocity and variety of the data gathered, the inherent value of the data and the veracity of the data coming from the grid [5]. As the number of collection points in the grid and the frequency at which these points are sampled has increased, the large amounts of data gathered from the smart grid brings with it issues in terms of data storage, transmission and data analysis. The rate at which the data requires processing serves to add additional challenges as real-time processing of data is required to provide a complete record of the grid's operation and in order to support decision making processes [5]. The authenticity of the data gathered needs to be guaranteed as this has a large impact on decision making and the quality of the grid profiles generated, whether they are customer consumption profiles, peak load prediction tables or pricing profiles [5].

The realisation of a secure IoE comes with a variety of problems. The cyber-physical nature of smart grid implementations means that the grid is exposed to a variety of security vulnerabilities such as equipment and network failures, environmental erosion and disasters, industrial espionage, physical tampering and malicious cyber-attacks [2]. The use of traditional ICT mechanisms of cyber security are inadequate and inappropriate towards the overall security of the grid. Security solutions for smart grid applications are required to consider the distributed nature of the network devices and communications, the resource-constrained nature of the communicating devices and the vastness of the network deployment. Adding provision for a bidirectional supply/demand chain and incorporating cyber processes into previously isolated physical process introduces a wide variety of security vulnerabilities that may result in ''either abandoning an exceptionally promising solution for energy issues or deploying a system that could be the Achilles heel of any industrialised nation's critical infrastructure'' [7]. Attacks on a smart grid could result in the widespread loss of system components owing to the high interconnectivity of the network, which could lead to physical damages to the plant or end user assets, loss of generation efficiency resulting in rolling blackouts that last days, weeks or months [4].

The remainder of the paper is organised as follows: Section II provides a brief look into previous work conducted towards addressing the security challenges faced in the IoE and IoT. In Section III, the equipment and tools used in conducting this research are listed and the main research methodology is introduced and discussed in Section IV. Section V presents the results obtained for the execution time, power consumption and memory occupation on the software-secured Cortex-M series processors while Section VI gives a more detailed analysis of the results alongside the main observations and recommendations made by the authors regarding the use of software cryptographic services with the Cortex-M

series. Section VII serves to conclude the paper and gives insight into the research directions in which additional work shall be conducted by the authors.

II. RELATED WORK

A smart grid implementation can be formed from a wide variety of device types, each with differing resources available to them. These technologies could affect supply, load or grid conditions dependent on the current grid consumption profile [8]. The devices would typically form part of the advanced metering infrastructure (AMI), used in order to provide information regarding consumption and pricing profiles within the smart grid, an energy management system (EMS), a distribution management system (DMS), wide area measurement system (WAMS) and network communication that make up the smart grid configuration [1], [9].

In RFC 7228, Keranen *et al.* [10] developed a classification scheme of the devices that could typically be found in internet of energy and sensor network schemes. Devices classified as Class 0 devices are typically sensor-like endpoints which would, ordinarily, have less than 10kB of RAM and 100kB of Flash memory available to them and would not be able to support secure communications [10]. Class 1 devices, with approximately 10kB of RAM and 100kB of Flash memory available to them, would be the devices that would typically handle basic communications within the network with the ability to support protocol stacks, such as CoAP, for constrained devices [10]. Class 2 devices could typically be used as gateway or as part of smart meter devices given that they would have approximately 50kB of RAM and 250kB of Flash available to them. These devices would be capable of supporting more secure communications and protocol stacks however they would perform best when running lightweight and energy-efficient protocols [10].

While it does provide a good system of classification for devices that could be found within an IoE application, the RFC does not take into consideration the advancements and improvements that have been made within Internet of Things technologies and within the fog computing sector. Class 0 devices could be combined with Class 2 devices to form a more powerful, less constrained endpoint device for the Smart grid, such as a smart meter or data concentrator unit [1]. As such, it would no longer be appropriate to say that the Class 0 device is incapable of supporting secure communications, given that the resources available to it have no increased. Also, devices that could be considered class 0 devices are getting more powerful processors and are being made available with more memory resources. As such, one would need to conduct an evaluation on the new-generation device before concluding that it is incapable of supporting secure communications or security processes.

Smart grids are subjects of a wide variety of vulnerabilities that could lead to a number of cyber-physical attacks by malicious actors. The distributed nature of a smart grid means that it is difficult to implement access control mechanisms to the grid. A lack of protocols and hardware standards makes it harder to hold accountable third party vendors who supply insecure technologies intended for smart grid operations [4], [11]. The interconnection of equipment, applications, utilities and consumers open the grid up to the possibility of cascading failure from the various subsystems within the grid and information leaks as a result of insufficient mechanisms designed to protect the confidentiality of network and customer data [1]. The vast amounts of data generated by the smart grid and the variety of sources from which the data is gathered introduce issues of trust, data authenticity and the use of secure storage facilities [1], [5]. Smart meters vulnerabilities are easily monetised and as a result these devices are often subject to a variety of customer fraud attacks, such as the falsification of meter usage readings whether to lower the readings of an individual or to increase the readings of an unsuspecting target [6]. Smart meters can also be used as part of large scale denial of service (DoS) attacks on the smart grid, through the flooding of the grid servers with numerous *keepalive* messages or falsified meter readings [6], [11]. In the larger grid, alteration type attacks could create false links to change the server view of the grid network, thus changing the grid topology [11], e.g. simple wormhole attacks can cause data to be routed through attack nodes [12]. This can also be used in conjunction with black-hole attacks where data is routed to a compromised router and dropped or packets of a certain type are dropped in the false network topology. This could severely hamper the smart grid network function [11].

With consideration of the security challenges faced by smart grid applications, the goal of any security scheme intended for the smart grid is to ensure that the grid is able to maintain the integrity, confidentiality, availability and authenticity of the data, detect and prevent attacks on the grid, maintain end user privacy by protecting usage information collected on smart meters, establish trust in the data sent and received within the grid, provide attestation for devices within the smart grid, and provide adequate key management and cryptographic services on grid devices [4], [6], [8], [11]. With the large scale of work required until a complete security solution is developed, this research shall focus on the establishment of cryptographic services for the smart grid edge network.

Cryptographic techniques are intended for the establishment and maintenance of integrity and confidentiality in the IoE however, with the resource-constraints found on old-generation, Class 0 and Class 1 microcontrollers, implementing cryptographic operations on IoT endpoint devices without algorithm optimisation and minimisation techniques remained a challenge. Khurana *et al.* [8] note that any cryptographic services intended for use in smart grid applications need to include a key management system that is able to update or revoke security keys. However, the authors also note that devices intended for the smart grid do not have the processing and memory resources required to support real time, high-volume cryptographic services [8]. Previous attempts at implementing cryptographic services on devices

intended for sensor network or IoT applications have aided in supporting this viewpoint.

Antonopoulos *et al.* [13] utilised the Advanced Encryption Standard (AES), Rivest Cipher 5 algorithm (RC5) and SkipJack as their cryptographic algorithms of choice in order to determine the effect of security processes on the ATmega128L processor found on the MicaZ. 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 [13]. 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 [13].

The study conducted by Chang *et al.* [14] determined the energy consumption of RC5, the Data Encryption Standard (DES) with cipher block chaining (CBC), AES and the Secure Hashing Algorithm 1 (SHA1). Operation times were only determined for 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 [14]. 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 read-only memory (ROM) [14]. 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 [14].

Guimaraes *et al.* [15] tested the energy utilisation and execution time of SkipJack, RC5, the Rivest Cipher 6 algorithm (RC6), the Tiny Encryption Algorithm (TEA) and DES. Measurement of the ATmega128L processor, on the Mica2, in active mode without the inclusion of security processes gave the authors a central processing unit (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 [15]. After having determined the base energy consumption measurements for the processor prior to the inclusion of the cryptographic algorithms, the authors note that the increase in energy consumption would be determined by the added execution and transmission time as a result of the cryptographic processes [15]. 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 [15].

Trad *et al.* [16] analysed AES, RC5 and RC6 in terms of the energy consumption, operational time and memory occupation on the ATmega128L processor. As in [14] and [15]; to calculate the energy consumption of the cryptographic algorithms, the authors measured the voltage drop across two resistors using an oscilloscope [16]. 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 [16].

The results from the studies provided a good indication of the capability of older generation sensor nodes to handle cryptographic algorithms and their inadequacy for use in real time, smart grid applications. The knowledge base however, requires updating owing to changes that have occurred in the years past. Of the six (6) algorithms tested in the studies above, four (4) are not appropriate for use in an industrial setting; leaving a gap regarding the capabilities of IoT processors in running cryptographic algorithms that are approved for use in industrial applications. Both SkipJack and DES were deprecated by the National Institute of Standards and Technology (NIST) in 2016 [17] and in 2005 [18], respectively. RC6 was developed as a candidate algorithms for AES and is subsequently non-standard [19] whereas RC5 is a proprietary algorithm of RSA Data Security Laboratories and may not be as easily modifiable and usable as an open standard algorithm [20].

More recently, studies on cryptography for the IoT have looked into the use of elliptic curve cryptography (ECC) as a mechanism to provide security on network endpoints. Liu *et al.* [21] noted that ECC is a more applicable cryptographic algorithm for low-end IoT devices owing to the algorithm's relatively small key sizes and operand lengths however that the ECDSA signature algorithm, which is capable of providing security equivalent to AES 128 at 255-bits, is the verification of the signature is a computationally intensive process which has the potential of introducing delays into a real-time IoT network [21]. In an effort to speed up the verification process of ECDSA, the authors utilise twisted Edwards curves with a computable endomorphism. Two versions of the optimisation were proposed, one which allowed for savings in memory occupation but at the cost of execution performance and another which allowed for savings in execution time at the cost of the amount of memory occupied on the processor [21]. Additional details regarding the design of the hardware architectures utilised in the implementation of the optimised verification engines were given in [21].

A second study by Liu *et al.* [22] also notes that despite the relatively small key sizes, the computational requirements of elliptic curve algorithms may make them unsuitable for

real IoT implementations however, unlike the previous study which was concerned about the potential processing delay that could be incurred, the authors indicate that the elliptic curve algorithms may have too high an energy consumption for use in the IoT [22]. In this work, the authors implement a hybrid class of elliptic curves, called MoTE curves after the Montgomery and twisted Edwards curves, on the 8-bit ATmega128 processor as implemented on the MicaZ and on the MSP430 as implemented on the Tmote Sky while optimising for execution speed and memory occupation [22].

Owing to the limited resources available on the MicaZ, the more memory efficient ECC algorithm was implemented. The authors found that 11.5% of the total RAM and 10.2% of the total ROM was occupied by the optimised ECC algorithms. The modified double-base scalar multiplication which would be utilised in ECDSA achieved an execution time of 0.841s for a 159-bit prime; with a 191-bit prime it achieved an execution time of 1.339s; the 223-bit prime gave an execution time of 2.001s and an execution time of 2.848s was achieved using a 255-bit prime [22]. Energy consumption of the cryptographic operation was estimated to be 32.58mJ for 159-bit ECC [22].

Both the execution time and memory occupation optimisations were implemented on the Tmote Sky. The authors show that the MSP430 was able to give execution times of 0.581s for 159-bit, 0.883s for 191-bit, 1.278s for 223-bit and 1.772s for 255-bit primes while running the execution time optimisation and execution times of 0.701s for 159-bit, 1.080s for 191-bit, 1.579s for 223-bit and 2.209s for 255-bit primes while running the memory efficient optimisation [22].

While more recent studies are looking into the use of newer, standard cryptography algorithms in the IoT, with research still being conducted using older generation platforms, the cryptographic algorithms are still being labelled as unsuitable for use in the IoT. This however does not give consideration to the increases in available resources and improvements in energy efficient processing that is seen on new generation processors for the IoT. As a result, this work aims to determine the performance of new generation IoT microprocessors in loading and running software-implemented cryptographic algorithms capable of encryption, decryption and key generation using pseudo random number generation in order to determine their applicability for use in the IoE smart grid applications.

III. EQUIPMENT

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

A. MCUs AND DEVELOPMENT BOARDS

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 [23]; the M3, M4, and M33 are intended as mid-range choices, balancing performance and energy efficiency [23] and the M7 is intended to provide maximum performance, making it ideal for high processing applications [23].

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 root of trust (RoT) and trusted execution environment (TEE) in the processors [23]–[27].

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 STM32F0Discovery [28], STM32VLDiscovery [29], STM32F4Discovery [30] and STM32F767-Nucleo144 [31] development boards— which are distributed with the M0 [24], M3 [25], M4 [26], [27] and M7 [27] processors respectively— were chosen after taking into consideration the availability of the boards, cost, processing speeds, development tools and available peripherals on the evaluation boards. The processors were chosen to cover a wide spectrum of the operating frequencies available in the processor series in addition to covering two of the three architectures available across the series. The operating frequency of the M3 distributed on the STM32VL is close to the minimum available frequency for the processor at 24MHz [29] whereas the M7 distributed on the STM32767-Nucleo comes close to the maximum available frequency for the processor at 216MHz [31]. The processors were also able to represent improvements made by ARM in the development of IoT processors in recent years; with the announcement dates for the chosen processors cover the ten (10) year period between 2004 and 2014.

B. CRYPTOGRAPHIC LIBRARY

The STMicroelectronics X-CUBE-CRYPTOLIB firmware library implements a variety of standard and non-standard cryptographic algorithms for use with the STM32 processor series. The algorithms chosen for this research —AES 128 in counter mode (AES128-CTR), SHA-256 and Elliptic Curve Digital Signature Algorithm (ECDSA) — have been certified for industrial use by the NIST Cryptographic Algorithm Validation Program [32]. The program provides independent, accredited ''validation testing of FIPS approved and NIST recommended cryptographic algorithms'' [33] 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 [33]. 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 both 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.

IV. EXPERIMENT METHODOLOGY

To adequately determine the capabilities of the new generation processors, three (3) main metrics were considered for this work:

- Algorithm Execution Time: 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.

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 Fig. 1 (a) and (b). 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. 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

(a)

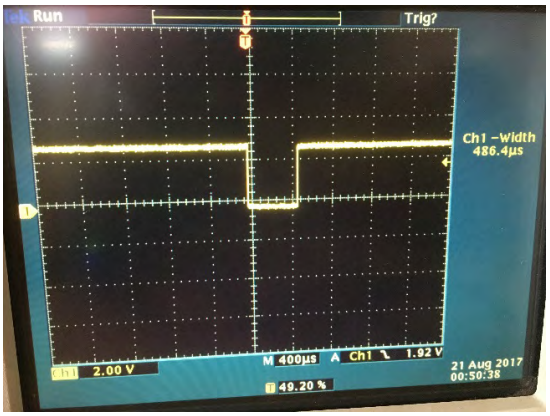
(b)

FIGURE 1. Execution time experiment setup. (a) Oscilloscope setup shown using the STM32F0Discovery. (b) Close-up of probe connections.

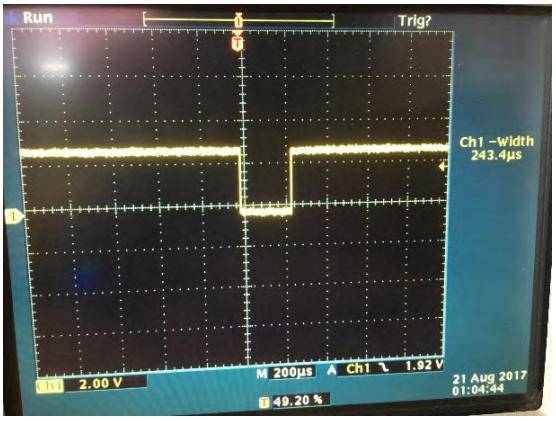
toggling points in order to measure the execution times of the next portions of the algorithm. The timestamps visible on each waveform were used to match the captured waveform as well as the measured execution time, with its relevant run number. Fig. 2 (a) - (c) gives the waveform captures seen for AES128-CTR when run on the STM32F0Discovery; illustrating the execution times measured when toggling the pin for the entire algorithm, the encryption portion of the algorithm and the decryption portion of the algorithm.

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

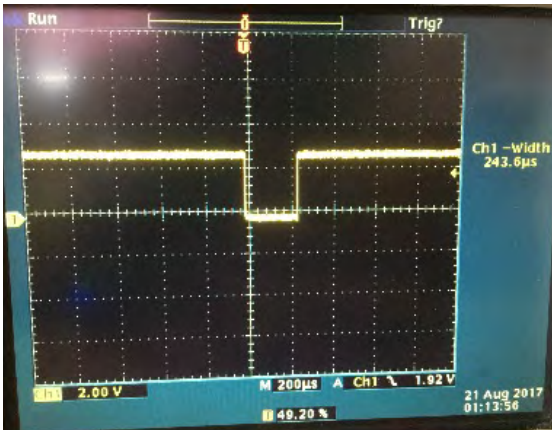
To determine the power consumption of the cryptographic algorithms, the current consumption module (I_{DD}) found on the development boards was used in conjunction with a 1.2 Ω shunt resistor in order to measure the voltage drop seen across the resistor during the execution of the cryptographic algorithms. 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.



(a)



(b)



(c)

FIGURE 2. Execution time waveforms for single run of AES128-CTR experiment on STM32F0Discovery. (a) Full Algorithm, (b) Encryption-only and (c) Decryption-only.

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 [30], 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. A high level, circuit diagram representation of the power consumption experiment is given in Fig. 3 whereas the physical setup for the experiment can be seen in Fig. 4 (a) and (b). JPx was used to denote the general jumper pin number for the I_{DD} module, as the specific number was dependent on the development board. MCU was used to denote the general microcontroller processor used on the different development boards and V_S was used to denote the supply voltage which is given as 5V, in the development user manuals, when the ST-Link/USB connection is utilised.

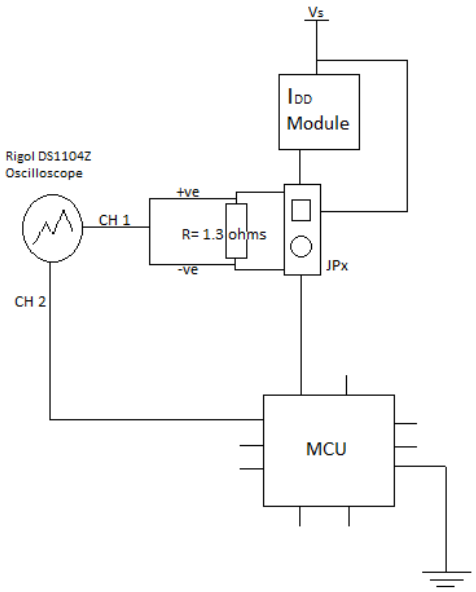
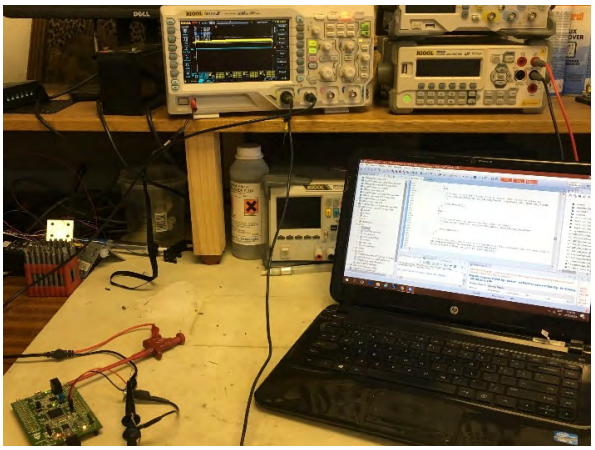


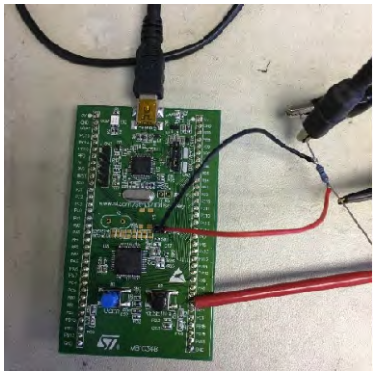
FIGURE 3. High level representation of power consumption experiment setup (based on Fig. 3 in [34]).

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 Fig. 5 as the V_{PP} measurement.

For this experiment, the power consumption was only measured for the whole cryptographic algorithm as a worst-case, consecutive execution scenario for a security mote. 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.



(a)



(b)

FIGURE 4. Physical setup of the power consumption experiment. (a) Entire setup and (b) Close-up of development board connections.

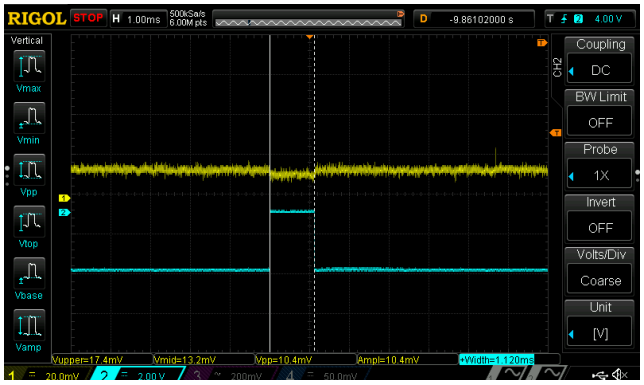
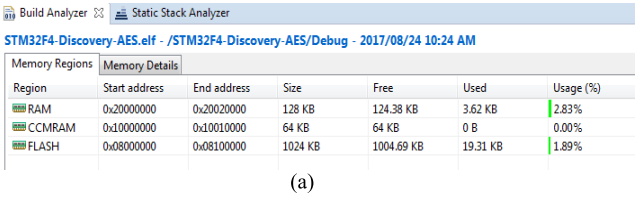
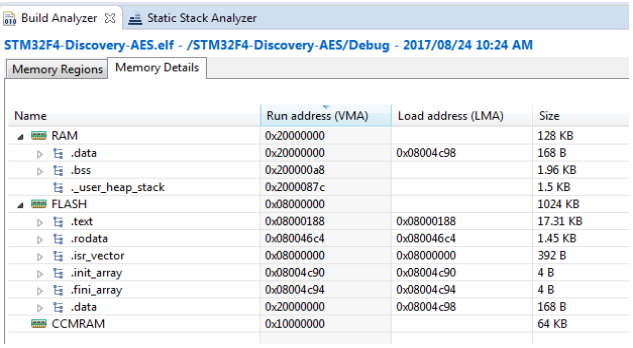


FIGURE 5. Execution time and bounded voltage waveform for SHA256 running on the STM32VLDiscovery.

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 [35]. This can be seen in Fig. 6 (a) and (b), which shows the memory occupation of AES128-CTR on the STM32F4Discovery. The analyser can



(a)



(b)

FIGURE 6. Memory Occupation of AES128-CTR on the STM32F4Discovery. (a) Summary version. (b) Detailed expansion version.

be switched also to view the stack utilisation of the algorithms however that was beyond the scope of this research.

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.

V. RESULTS

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 (X_i) over the number of runs (N), the mean ($\overline{X}$), standard deviation (σ_X) and standard error of the mean ($\sigma_{\overline{X}}$) were determined using (1) to (3).

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{1}$$

$$\sigma_X = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2} \tag{2}$$

$$\sigma_{\overline{X}} = \frac{1}{\sqrt{N}}\sigma_X \tag{3}$$

A. EXECUTION TIME

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. Fig. 7 gives the results of the execution of the full algorithm whereas Figs. 8 and 9 give the results of the execution times for the encryption and decryption operations respectively.

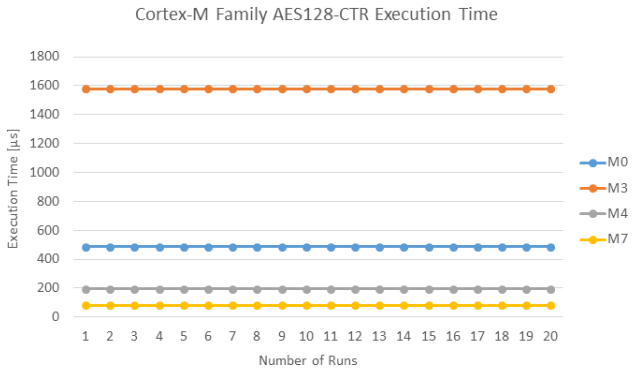


FIGURE 7. Execution time of Cortex-M series processors running AES128-CTR.

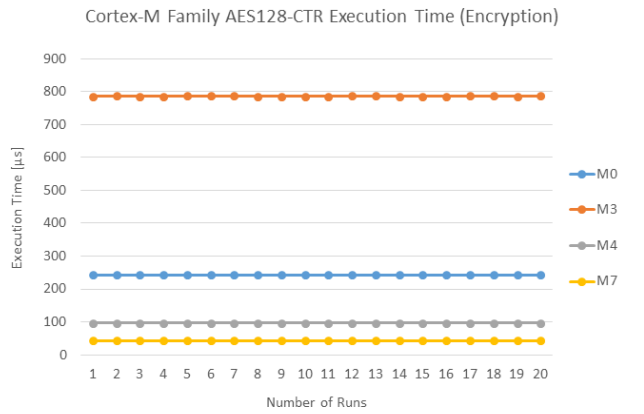


FIGURE 8. Execution time of Cortex-M series processors running AES128-CTR (Encryption Only).

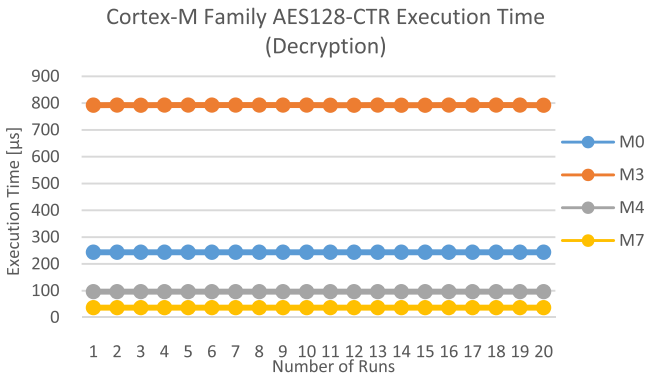


FIGURE 9. Execution time of Cortex-M series processors running AES128-CTR (Decryption Only).

Looking at the three figures, it could be seen that very little deviation occurred for the execution time of AES128-CTR over the twenty (20) runs. With an operating frequency of 24MHz, the M3 had the longest running times whereas 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 similar to the performance of the M7, which ran approximately 140.334 μs faster than the M4 with a 22.222% speed increase over processor. Comparing the execution times seen in Figs. 8 and 9, one could see that the time spent for encryption or decryption in AES128-CTR was nearly identical.

TABLE 1. Execution time mean, standard deviation and standard error for MCUs running AES128-CTR.

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

TABLE 2. Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (Encryption Only).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

TABLE 3. Execution time mean, standard deviation and standard error for MCUs running AES128-CTR (Decryption Only).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

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 1 to 3:

Using the result above, the total execution time for AES12-CTR on the Cortex-M processors was estimated as follows:

$$\begin{aligned}
\text{M0}:&\ 486.470\ \mu s \pm 0.039\ \mu s\\
\text{M3}:&\ 1578.500\ \mu s \pm 0.199\ \mu s\\
\text{M4}:&\ 193.870\ \mu s \pm 0.047\ \mu s\\
\text{M7}:&\ 80.667\ \mu s \pm 0.005\ \mu s
\end{aligned}$$

Similarly, using the results in Table 2, the execution time for the encryption operations could be estimated as:

$$\begin{aligned}
\text{M0}:&\ 243.385\ \mu s \pm 0.017 \mu s\\
\text{M3}:&\ 786.175\ \mu s \pm 0.047\ \mu s\\
\text{M4}:&\ 96.984\ \mu s \pm 0.015\ \mu s\\
\text{M7}:&\ 44.000\ \mu s \pm 0.000\ \mu s
\end{aligned}$$

The execution time for the decryption operations was estimated from the results in Table 3 as:

$$\begin{aligned}
\text{M0}:&\ 243.735\ \mu s \pm 0.021 \mu s\\
\text{M3}:&\ 792.670\ \mu s \pm 0.042\ \mu s\\
\text{M4}:&\ 96.704\ \mu s \pm 0.015\ \mu s\\
\text{M7}:&\ 37.071\ \mu s \pm 0.003\ \mu s
\end{aligned}$$

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 whereas 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.

The STM Cryptographic library implements two versions of ECDSA: one version that only implements signature generation and verification from preloaded public and private keys — identified as ECDSA (Sign-Verify) — and another version that utilises a pseudo random number generator (PRNG) for key generation — identified as ECDSA (Key Gen-Sign-Verify). The execution times for both algorithms on the Cortex-M processors were determined over twenty (20) runs. Fig. 10 gives the observed results of ECDSA without key generation whereas Fig. 11 gives the results for ECDSA with key generation.

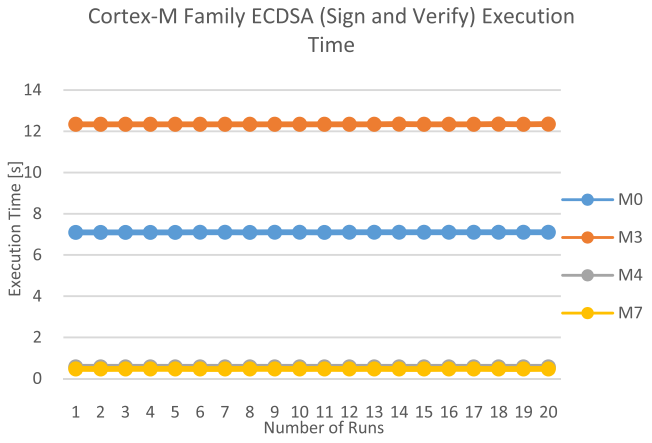


FIGURE 10. Execution time of Cortex-M series processors running ECDSA (Sign-Verify).

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

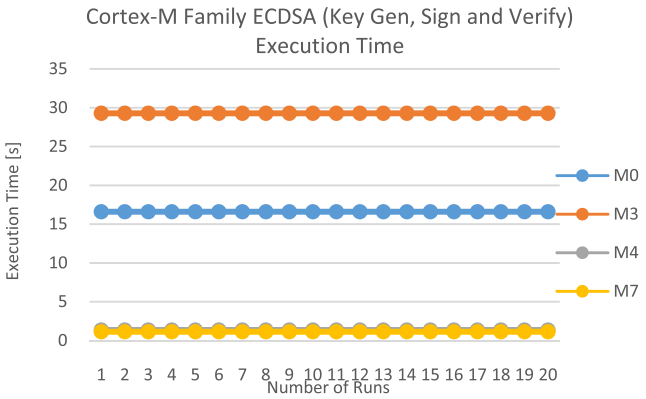


FIGURE 11. Execution time of Cortex-M series processors running ECDSA (Key Gen-Sign-Verify).

TABLE 4. Execution time mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

TABLE 5. Execution time mean, standard deviation and standard error for MCUs running ECDSA (Key Gen-Sign-Verify).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

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 and 5:

Using the mean and standard error the estimated execution time for ECDSA sans key generation was given as follows:

$$\begin{aligned} \text{M0}&: \ 7.101\ \text{s} \pm 0.000\ \text{s} \\ \text{M3}&: \ 12.342\ \text{s} \pm 0.001\ \text{s} \\ \text{M4}&: \ 0.572\ \text{s} \pm 0.000\ \text{s} \\ \text{M7}&: \ 0.471\ \text{s} \pm 0.000\ \text{s} \end{aligned}$$

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

$$\begin{aligned} \text{M0}&: \ 16.600\ \text{s} \pm 0.000\ \text{s} \\ \text{M3}&: \ 29.284\ \text{s} \pm 0.001\ \text{s} \\ \text{M4}&: \ 1.362\ \text{s} \pm 0.000\ \text{s} \\ \text{M7}&: \ 1.141\ \text{s} \pm 0.000\ \text{s} \end{aligned}$$

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.

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. Fig. 12 gives the results of the execution time for the full algorithm whereas Figs. 13 and 14 give the results of the execution time for hash generation and digest verification operations respectively.

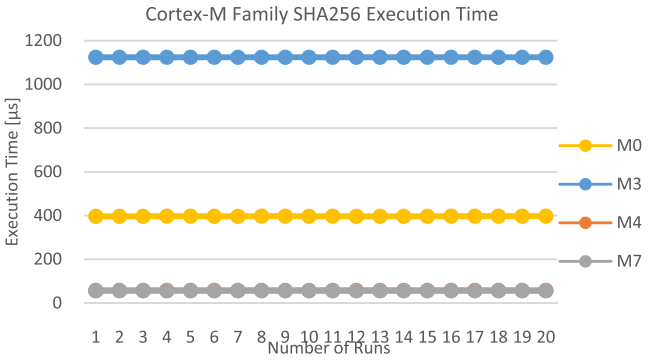


FIGURE 12. Execution time of Cortex-M series processors running SHA256.

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 6 to 8:

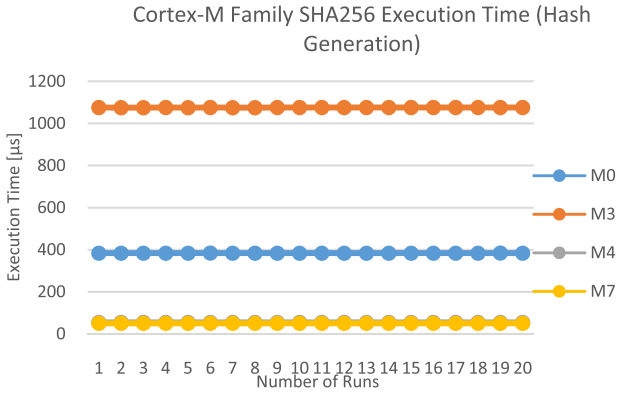


FIGURE 13. Execution time of Cortex-M series processors running SHA256 (Hash Generation Only).

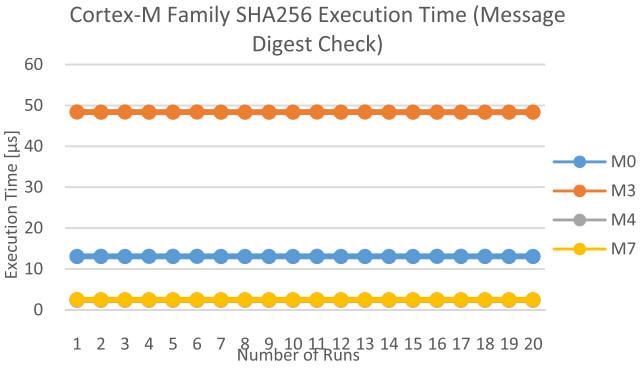


FIGURE 14. Execution time of Cortex-M series processors running SHA256 (Message Digest Checking Only).

TABLE 6. Execution time mean, standard deviation and standard error for MCUs running SHA256.

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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 time for SHA256 on the Cortex-M processors was estimated as follows:

$$\begin{aligned}
\text{M0}&: \ 397.000\ \mu s \pm 0.046\ \mu s \\
\text{M3}&: \ 1124.000\ \mu s \pm 0.000\ \mu s \\
\text{M4}&: \ 57.361\ \mu s \pm 0.008\ \mu s \\
\text{M7}&: \ 56.029\ \mu s \pm 0.004\ \mu s
\end{aligned}$$

From the results given in Table 7, the execution times for the hash generation operations could be estimated as:

$$\begin{aligned}
\text{M0}&: \ 384.040\ \mu s \pm 0.028\ \mu s \\
\text{M3}&: \ 1075.750\ \mu s \pm 0.099\ \mu s \\
\text{M4}&: \ 55.102\ \mu s \pm 0.004\ \mu s \\
\text{M7}&: \ 50.438\ \mu s \pm 0.001\ \mu s
\end{aligned}$$

TABLE 7. Execution time mean, standard deviation and standard error for MCUs running SHA256 (Hash Generation Only).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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

TABLE 8. Execution time mean, standard deviation and standard error for MCUs running SHA256 (Message Digest Checking Only).

	$\overline{X}$	σ_X	$\sigma_{\overline{X}}$
	[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 8 as:

$$\begin{aligned} \text{M0}&: \ 13.086\ \mu s \pm 0.002\ \mu s \\ \text{M3}&: \ 48.371\ \mu s \pm 0.003\ \mu s \\ \text{M4}&: \ 2.363\ \mu s \pm 0.000 \mu s \\ \text{M7}&: \ 2.491\ \mu s \pm 0.001\ \mu s \end{aligned}$$

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 μs was seen between the two processors while a difference of 0.128 μs 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 μs. 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.

B. 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):

$$I_{MCU} = V_{MCU} \div R \text{ Amperes (A)} \tag{4}$$

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

Equation (5) was then used to calculate the power consumption of the MCU:

$$P_{MCU} = V_S \times I_{MCU} \text{ Watts (W)} \tag{5}$$

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 (X_i) over the number of runs (N), the mean ($\overline{X}$), standard deviation (σ_X) and standard error of the mean ($\sigma_{\overline{X}}$) were determined using (1) to (3).

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 Fig. 15.

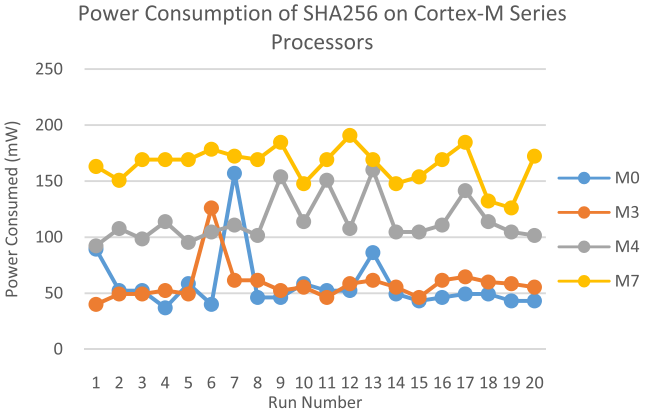


FIGURE 15. Power consumption of Cortex-M series processors running AES128-CTR.

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 above, the mean, standard deviation, and standard error were calculated on the results from the twenty (20) runs.

Using the calculated mean and standard error presented in Table 9, the consumption of the processors running AES128-CTR could be estimated as follows:

$$\begin{aligned} \text{M0}&: \ 72.462\ \text{mW} \pm 8.339\ \text{mW} \\ \text{M3}&: \ 37.885\ \text{mW} \pm 1.622\ \text{mW} \\ \text{M4}&: \ 106.154\ \text{mW} \pm 5.443\ \text{mW} \\ \text{M7}&: \ 143.385\ \text{mW} \pm 9.579\ \text{mW} \end{aligned}$$

It could be seen that while the M3 did consume the least power and offered the least deviation, the M4 presented a

TABLE 9. Power consumption mean, standard deviation and standard error for MCUs running AES128-CTR.

	$\overline{X}$		σ_X		$\sigma_{\overline{X}}$	
	[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

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.

The power consumption of the two (2) versions of the public key cryptographic algorithm ECDSA was tested on the M-series processors. Fig. 16 compares the power consumption of the MCUs running ECDSA (Sign-Verify) over the course of twenty (20), independent runs whereas Fig. 17 compares the power consumption of the MCUs running ECDSA (Key Gen-Sign-Verify) over twenty (20) runs.

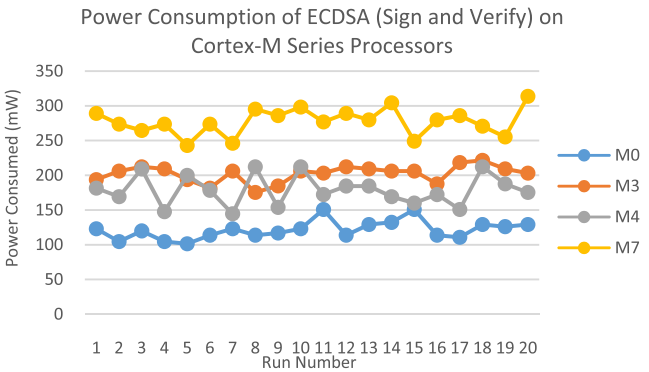


FIGURE 16. Power consumption of Cortex-M series processors running ECDSA (Sign-Verify).

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

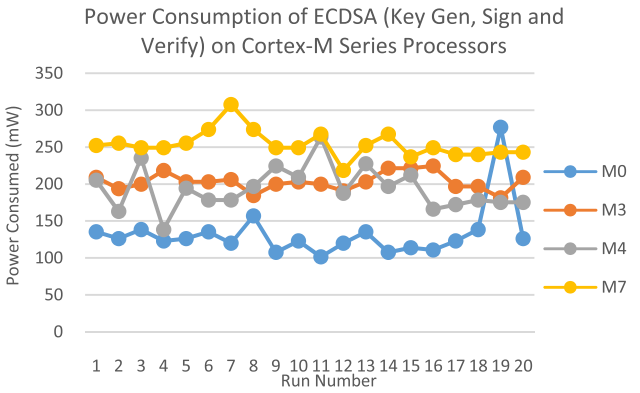


FIGURE 17. Power consumption of Cortex-M series processors running ECDSA (Key Gen-Sign-Verify).

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 better a more stable power consumption profile for the MCUs than ECDSA with key generation.

TABLE 10. Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Sign-Verify).

	$\overline{X}$		σ_X		$\sigma_{\overline{X}}$	
	[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

In order to provide mathematical confirmation for the aforementioned observations, using the mean and standard error given in Table 10, the consumption of the processors running ECDSA (Sign-Verify) was estimated as follows:

$$\begin{aligned} \text{M0}&: \ 121.538\ \text{mW} \pm 2.982\ \text{mW} \\ \text{M3}&: \ 202.308\ \text{mW} \pm 2.746\ \text{mW} \\ \text{M4}&: \ 178.923\ \text{mW} \pm 4.887\ \text{mW} \\ \text{M7}&: \ 277.538\ \text{mW} \pm 4.280\ \text{mW} \end{aligned}$$

Similarly, the consumptions of ECDSA (Key Gen-Sign-Verify) were estimated using the figures given in Table 11:

$$\begin{aligned} \text{M0}&: \ 132.308\ \text{mW} \pm 8.153\ \text{mW} \\ \text{M3}&: \ 203.077\ \text{mW} \pm 2.621\ \text{mW} \\ \text{M4}&: \ 194.038\ \text{mW} \pm 6.513\ \text{mW} \\ \text{M7}&: \ 253.692\ \text{mW} \pm 4.089\ \text{mW} \end{aligned}$$

Looking at the standard errors, one could see that the observed deviation for ECDSA without key generation was

TABLE 11. Power consumption mean, standard deviation and standard error for MCUs running ECDSA (Key Gen-Sign-Verify).

	$\overline{X}$		σ_X		$\sigma_{\overline{X}}$	
	[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

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 51.192% 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.

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 Fig. 18.

The power consumption observed in the four (4) processors was surprisingly unstable, with the M3 showing

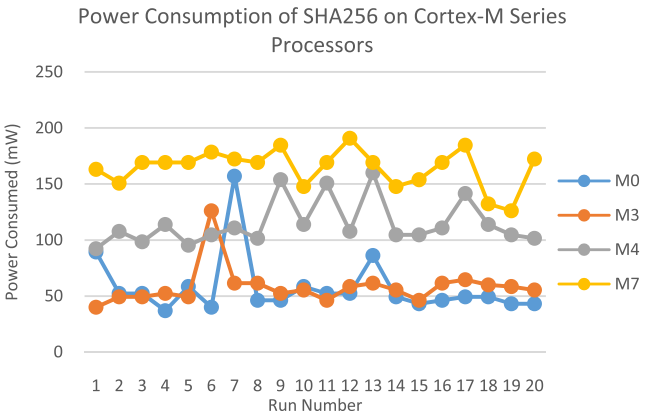


FIGURE 18. Power consumption of Cortex-M Series processors running SHA256.

TABLE 12. Power consumption mean, standard deviation and standard error for MCUs running SHA256.

	$\overline{X}$		σ_X		$\sigma_{\overline{X}}$	
	[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

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 12, the power consumption of the M-series processors was estimated as follows:

$$\begin{aligned} \text{M0}&: \ 57.538\ \text{mW} \pm 6.019\ \text{mW} \\ \text{M3}&: \ 58.231\ \text{mW} \pm 3.848\ \text{mW} \\ \text{M4}&: \ 114.615\ \text{mW} \pm 4.480\ \text{mW} \\ \text{M7}&: \ 164.462\ \text{mW} \pm 3.752\ \text{mW} \end{aligned}$$

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 had displayed the largest observed deviation from the processor series. The deviations observed from the remaining three processors were within range of the others whereas the M7 gave the largest, observed power consumption on average.

C. 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.

1) RAM

The detailed results of the RAM occupation analysis conducted for each cryptographic algorithm on the M-series processors are summarised in Table 13. A graphical representation of the percentage of RAM used by the algorithms is presented in Fig. 19.

TABLE 13. RAM occupation of cryptographic algorithms loaded onto Cortex-M series processors.

RAM	M0 [8kB]		M3 [8kB]		M4 [128kB]		M7 [512kB]	
	Used (kB)	Used (%)	Used (kB)	Used (%)	Used (kB)	Used (%)	Used (kB)	Used (%)
AES128 -CTR	1.67	20.85	1.64	20.51	3.62	2.83	2.85	0.56
ECDSA-S-V	1.57	19.63	1.54	19.29	3.52	2.75	1.57	0.31
ECDSA-G-S-V	1.57	19.63	1.54	19.29	3.52	2.75	1.57	0.31
SHA256	1.61	20.12	1.58	19.78	3.56	2.78	2.79	0.54

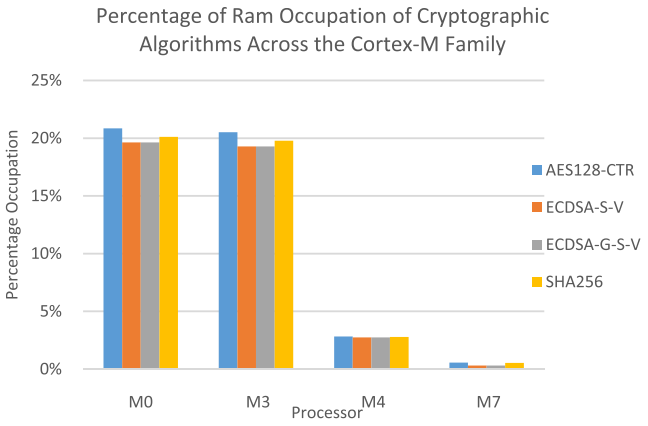


FIGURE 19. Comparison of the percentage RAM occupation of cryptographic algorithms loaded onto Cortex-M processors.

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.

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 14 with a graphical presentation of the percentage occupation given in Fig. 20:

TABLE 14. Flash occupation of cryptographic algorithms loaded onto Cortex-M series processors.

Flash	M0 [64kB]		M3 [128kB]		M4 [1024kB]		M7 [2048kB]	
	Used (kB)	Used (%)	Used (kB)	Used (%)	Used (kB)	Used (%)	Used (kB)	Used (%)
AES128-CTR	8.28	12.93	8.08	6.31	19.31	1.89	10.84	0.53
ECDSA -S-V	17.73	27.70	17.34	13.55	28.64	2.80	19.54	0.95
ECDSA -G-S-V	25.04	39.13	24.41	19.07	35.69	3.49	26.13	1.28
SHA256	6.32	9.88	6.15	4.80	17.8	1.74	8.57	0.42

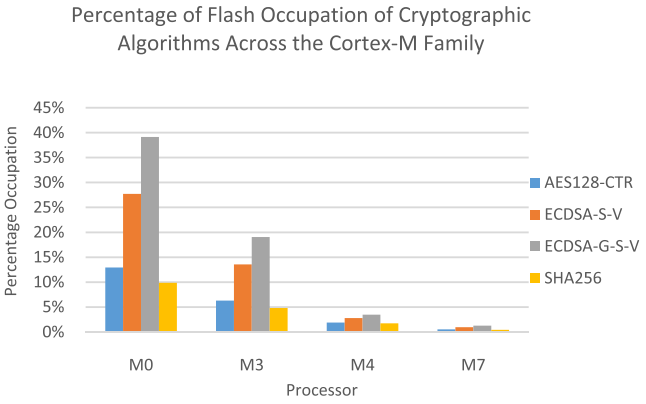


FIGURE 20. Comparison of the percentage Flash occupation of cryptographic algorithms loaded onto Cortex-M processors.

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.

VI. DISCUSSION

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 Section 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 in between executions. Fig. 21 gives the average execution time performances determined for the identified cryptographic algorithms.

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.

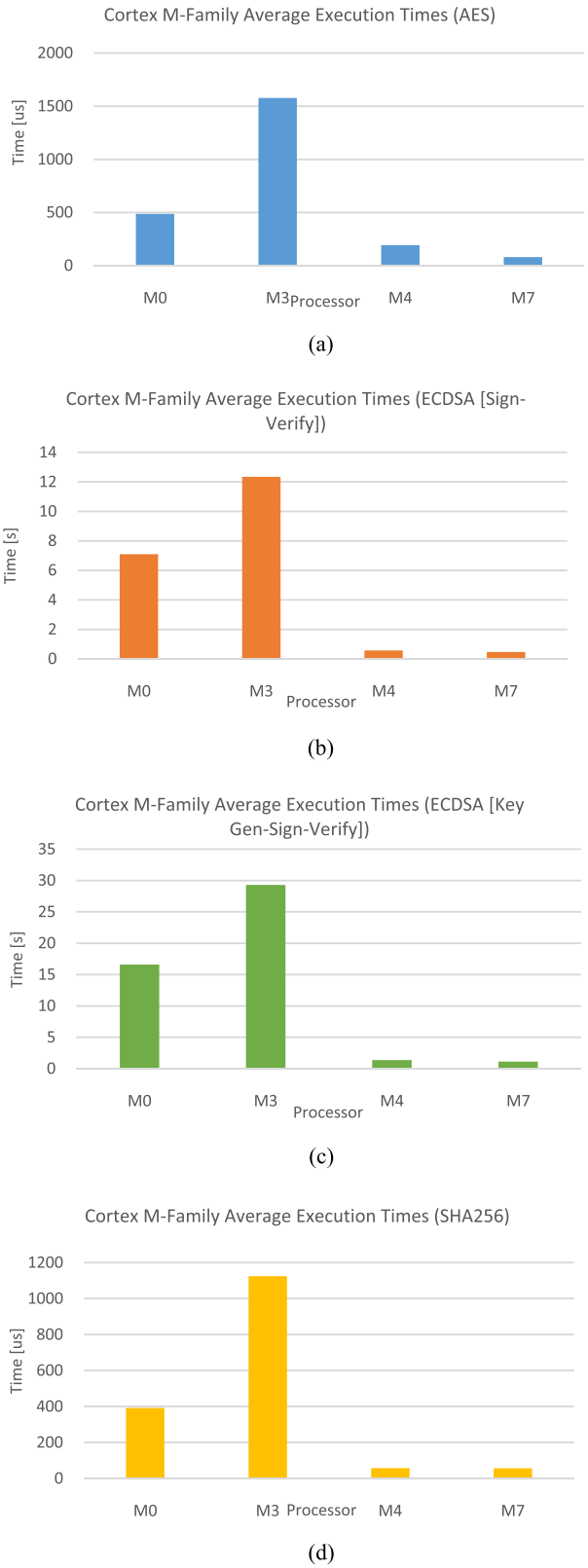


FIGURE 21. Average execution times of cryptographic algorithms on Cortex-M processors. (a) AES128-CTR, (b) ECDSA (Sign-Verify), (c) ECDSA (Key Gen-Sign-Verify) and (d) SHA256.

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 as they are specifically designed to provide a variety 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. Fig. 22 gives a comparison of the power consumption for the four (4) cryptographic algorithms run on the Cortex M processors.

Of the tested algorithms, AES128-CTR gave the lowest power consumption whereas ECDSA gave the highest

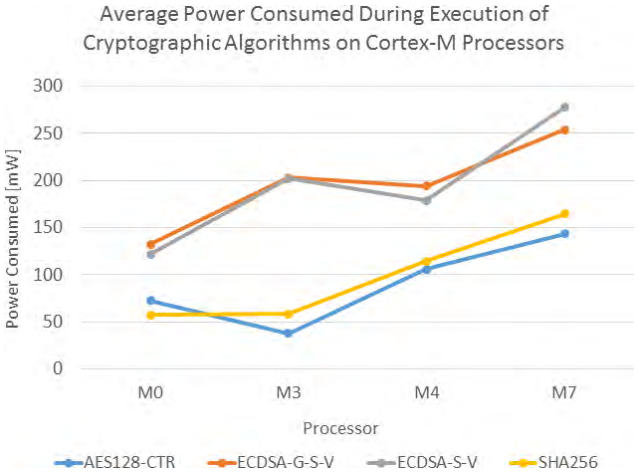


FIGURE 22. Average power consumption of cryptographic algorithms on Cortex-M processors.

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 Fig. 22, 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.

To determine the best overall performer in the execution of the cryptographic algorithms, the average consumption and execution times seen for the processors as a single unit needed to be considered. Fig. 23 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.

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 [13]–[16] 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 and memory occupation are given

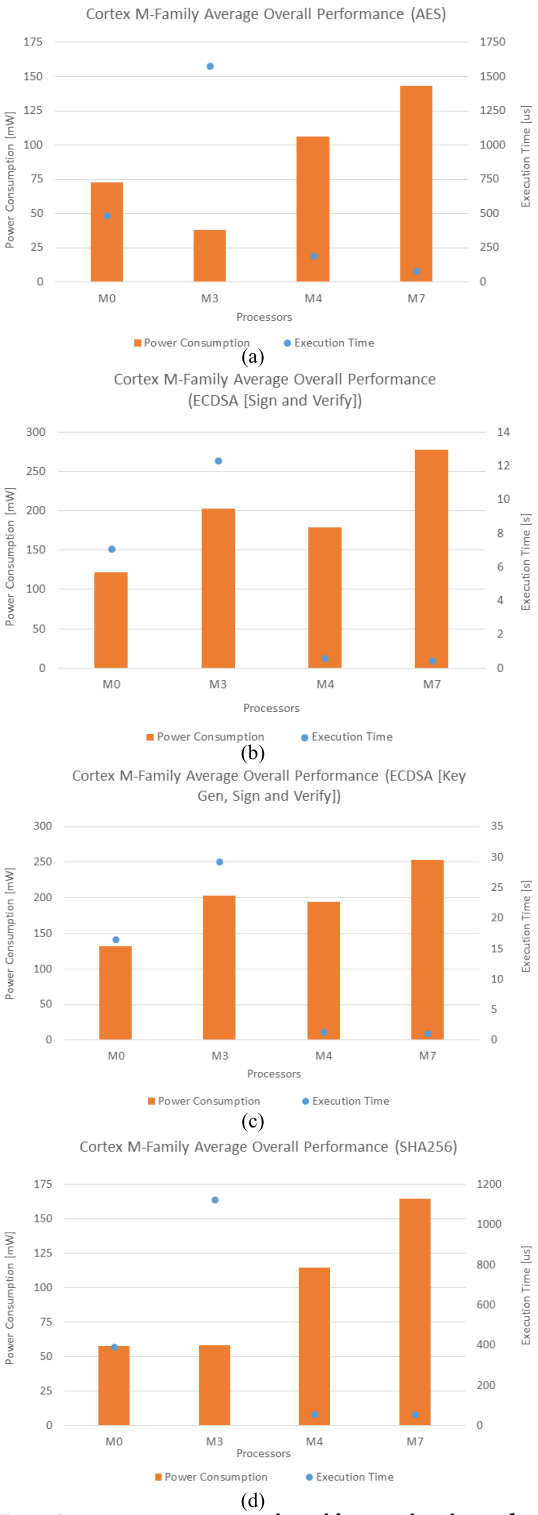


FIGURE 23. Average power consumption with execution times of cryptographic algorithms on Cortex-M processors. (a) AES128-CTR, (b) ECDSA (Sign-Verify), (c) ECDSA (Key Gen-Sign-Verify) and (d) SHA256.

in Figs. 24 and 25. 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 for the Atmega128L was used to determine the processor's performance in encrypting and decrypting a 64-byte block

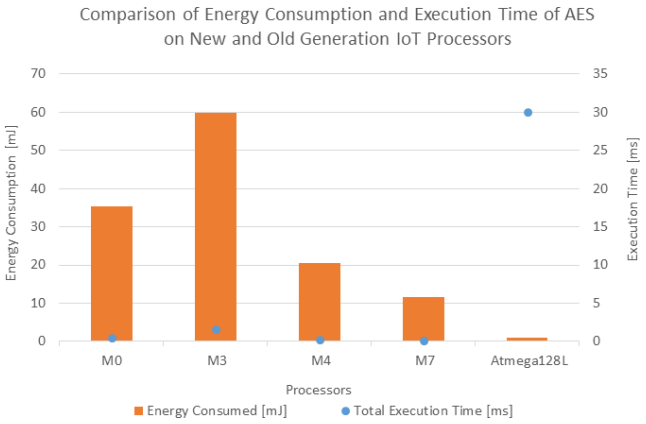


FIGURE 24. Energy consumption and execution time comparison for Atmega128L and Cortex-M processors (AES).

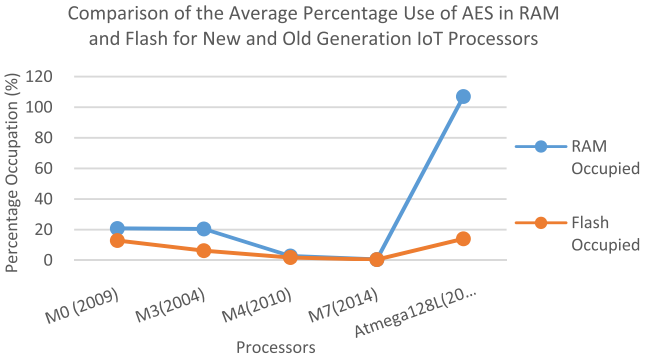


FIGURE 25. Memory occupation comparison for Atmega128L and Cortex-M processors (AES).

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 (6):

$$\text{Energy (J)} = \text{Power (W)} \times \text{time (s)} \tag{6}$$

Comparing the energy consumption performance of the Atmega to those of the Cortex processors, as shown in Fig. 24, 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. The execution time of the Atmega however, 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 Fig. 25 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 Cortex0 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.

As in the analysis of AES, a similar comparison of old and new generation processors was made against the performance of standard ECDSA on the Cortex-M processors against the results seen in [22] for the memory optimised version of ECDSA when run on the MicaZ's Atmega128L processor. The execution time and energy consumption results are given in Table 15 and the memory occupation results are given in Fig. 26.

Considering the execution time, it was seen that the Atmega128L, with fewer processing and memory resources, was able to vastly outperform the M0 and M3 when running the optimised validation process for ECDSA. The longest operating time seen for the 255-bit prime was approximately 59.893% faster than the M0 running standard ECDSA and was approximately 79.924% faster than the M3 running standard ECDSA. The M4 and M7 gave a better execution time performance than the best performing implementation of the optimised algorithm however were vastly outperformed in the energy consumption for the cryptographic performance. It was here that the value was seen in the optimisation of the ECDSA verification process as the energy consumption of the Atmega128L was significantly lower than the consumption of the Cortex Processors.

TABLE 15. Comparison of execution time and energy consumption of ECDSA on old and new generation processors.

Processor	Execution Time	Energy Consumption
	[s]	[mJ]
M0	7.101	863.081077
M3	12.342	2496.780385
M4	0.572	102.365471
M7	0.471	130.726166
Atmega128L (159-bit)	0.841	25.246
Atmega128L (191-bit)	1.339	40.183
Atmega128L (223-bit)	2.001	60.016
Atmega128L (255-bit)	2.848	85.429

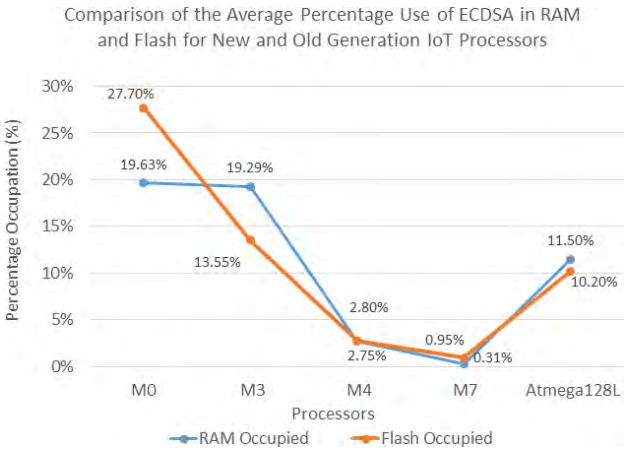


FIGURE 26. Memory occupation comparison for Atmega128L and Cortex-M processors (ECDSA).

The performance of the optimised algorithm did not falter when considering the memory occupation. The Atmega128L gave a smaller percentage occupation than the M0 and M3 processor but was beaten by the memory occupation of the M4 and M7, which gave significantly better occupation results with the standard ECDSA implementation.

What the results from the comparisons showed was the value in the optimisation of cryptographic algorithms for use in the IoT. The algorithm optimisations proposed in [22] were able to improve upon the performance of ECDSA such that the Atmega128L, which has less operational power and smaller memory resources than the Cortex processor, was able to provide a better overall performance than the newer Cortex M0 and M3 processors which ran the standard ECDSA algorithm. Should such optimisations be implemented on the newer generation processors; the use of ECDSA and other elliptic curve algorithms in IoT and IoE

applications could be achieved without requiring the inclusion of hardware accelerators to prevent network delays. Such optimisations however, would require verification and certification for industrial use under the NIST Cryptographic Algorithm Validation Program or other, similar standardisation processes in order to ensure that adequate security levels are still being achieved by the modified cryptographic algorithm.

VII. CONCLUSION AND FUTURE WORK

Previously, owing to the limited resources available, processors for WSNs and smart grids were left insecure and without security processes, resulting in large networks 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 quickly run cryptographic operations 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.

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 or an algorithm optimisation 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 of the Cortex-M performance results seen in this research with the performance results seen on previous applications of AES and ECDSA on the Mica2 and MicaZ platforms served to illustrate the improvements made in the capabilities of MCU platforms for the IoT over the past decade and the value in algorithm optimisation towards reducing excessive delay and energy consumption that could result from the adaptation of a cryptographic algorithm. The research has provided a detailed analysis regarding the capabilities of new generation IoT processors when running software cryptographic services and 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 authors' 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. However, work still needs to be done towards developing solutions for the other areas of smart grid security as cryptography alone only forms one part of a complete, IoE security solution.

As part of future work, the authors aim to continue work towards a general, implementable design for a general secure endpoint device at the IIoT edge, which can be adapted for use in smart grid applications. In particular, the authors aim to expand upon the research conducted within this work and compare the performances of the software-implemented cryptography algorithms to their hardware-implemented counterparts, revisit the experiments conducted in this work having expanded the list of cryptographic algorithms under consideration, test the performance of the Cortex-M processors when running the algorithms used within this work with longer key expansions and test for the performance differences that could be seen when a TRNG is used for key generation as opposed to a PRNG. Finally, the authors aim 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 IoE edge while maintaining a good longevity and maintainability.

REFERENCES

[1] K. Wang, X. Hu, H. Li, P. Li, D. Zeng, and S. Guo, "A survey on energy Internet communications for sustainability," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 3, pp. 231–254, May 2017.

[2] G. E. P. Kumar, K. Baskaran, R. E. Blessing, and M. Lydia, "Securing the smart grid network: A review," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, Chennai, India, Dec. 2016, pp. 1–6.

[3] C. Donitzky, O. Roos, and S. Sauty. (2014). *A Digital Energy Network: The Internet of Things and the Smart Grid*. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/iot-smart-grid-paper.pdf

[4] M. B. Line, I. A. Tøndel, and M. G. Jaatun, "Cyber security challenges in Smart Grids," in *Proc. 2nd IEEE PES Int. Conf. Exhib. Innov. Smart Grid Technol.*, Manchester, U.K., Dec. 2011, pp. 1–8.

[5] K. Wang *et al.*, "Wireless big data computing in smart grid," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 58–64, Apr. 2017.

[6] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security Privacy*, vol. 7, no. 3, pp. 75–77, Jun. 2009.

[7] M. Fabro, T. Roxey, and M. Assante, "No grid left behind," *IEEE Security Privacy*, vol. 8, no. 1, pp. 72–76, Feb. 2010.

[8] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," *IEEE Security Privacy*, vol. 8, no. 1, pp. 81–85, Feb. 2010.

[9] K. Wang, M. Du, S. Maharjan, and Y. Sun, "Strategic honeypot game model for distributed denial of service attacks in the smart grid," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2474–2482, Feb. 2017.

[10] A. Keranen, M. Ersue, and C. Bormann. (2014). *Terminology for Constrained-Node Networks*. [Online]. Available: https://tools.ietf.org/html/rfc7228#ref-IOT-SECURITY

[11] F. Jameel, "Network security challenges in smart grid," in *Proc. 19th Int. Multi-Topic Conf. (INMIC)*, Islamabad, Pakistan, 2016, pp. 1–7.

[12] A. Yang, E. Pagnin, A. Mitrokotsa, G. Hancke, and D. S. Wong, "Two-hop distance-bounding protocols: Keep your friends close," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2017.2771769.

[13] C. P. Antonopoulos, C. Petropoulos, K. Antonopoulos, V. Triantafyllou, and N. S. Voros, "The effect of symmetric block ciphers on WSN performance and behavior," *Proc. IEEE 8th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Barcelona, Spain, Oct. 2012, pp. 799–806.

[14] C. C. Chang, S. Muftic, and D. J. Nagel, "Measurement of energy costs of security in wireless sensor nodes," in *Proc. 16th Int. Conf. Comput. Commun. Netw.*, Honolulu, Hi, USA, 2007, pp. 95–102.
[15] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, "Evaluation of security mechanisms in wireless sensor networks," in *Proc. Syst. Commun. (ICW'ICHSN'ICMCS'SENET')*, Montreal, QC, USA, 2005, pp. 428–433.
[16] A. Trad, A. A. Bahattab, and S. B. Othman, "Performance trade-offs of encryption algorithms for wireless sensor networks," in *Proc. World Congr. Comput. Appl. Inf. Syst. (WCCAIS)*, Hammamet, Tunisia, 2014, pp. 1–6.
[17] E. Barker. (2016). *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms*. Accessed: Sep. 4, 2017. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175B.pdf
[18] *Data Encryption Standard (DES)*, FIPS Standard 46-3, 1999.
[19] EMC Corporation. (2017). *RSA Laboratories—RC6 Block Cipher*. Accessed: Sep. 4, 2017. [Online]. Available: https://www.emc.com/emc-plus/rsa-labs/historical/rc6-block-cipher.htm
[20] R. L. Rivest, "The RC5 encryption algorithm," in *Proc. 2nd Int. Workshop Fast Softw. Encryption*, Leuven, Belgium, Dec. 1995, pp. 86–96.
[21] Z. Liu, J. Groszschaedl, Z. Hu, K. Jarvinen, H. Wang, and I. Verbauwhede, "Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the Internet of Things," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 773–785, May 2017.
[22] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo, and L. Zhou, "On emerging family of elliptic curves to secure Internet of Things: ECC comes of age," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 3, pp. 237–248, Jun. 2017.
[23] Arm Limited. (2017). (Cortex-M). Accessed: Aug. 30, 2017. [Online]. Available: https://www.arm.com/products/processors/cortex-m
[24] Arm Limited. (2017). *Cortex-M0*. Accessed: Aug. 30, 2017. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m0
[25] Arm Limited. (2017). *Cortex-M3*. Accessed: Aug. 30, 2017. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m3
[26] Arm Limited. (2017). *Cortex-M4*. Accessed: Aug. 30, 2017. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m4
[27] Arm Limited. (2017). *Cortex-M7*. Accessed: Aug. 30, 2017. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m7
[28] STMicroelectronics. (2017). *STM32F0—ARM Cortex-M0 Microcontrollers*. Accessed: Sep. 4, 2017. [Online]. Available: http://www.st.com/en/microcontrollers/stm32f0-series.html?querycriteria=productId=SS1574
[29] STMicroelectronics. (2017). *STM32F1—ARM Cortex-M3 Microcontrollers*. Accessed: Sep. 4, 2017. [Online]. Available: http://www.st.com/en/microcontrollers/stm32f1-series.html?querycriteria=productId=SS1031
[30] STMicroelectronics. (2017). *STM32F4—ARM Cortex-M4 High-Performance MCUs*. Accessed: Sep. 4, 2017. [Online]. Available: http://www.st.com/en/microcontrollers/stm32f4-series.html?querycriteria=productId=SS1577
[31] STMicroelectronics. (2017). *STM32F7—ARM Cortex-M7 Microcontrollers*. Accessed: Sep. 4, 2017. [Online]. Available: http://www.st.com/en/microcontrollers/stm32f7-series.html?querycriteria=productId=SS1858
[32] STMicroelectronics. (2017). *X-CUBE-CRYPTOLIB—STM32 Cryptographic Firmware Library Software Expansion for STM32Cube (UM1924)*. Accessed: Sep. 5, 2017. [Online]. Available: http://www.st.com/en/embedded-software/x-cube-cryptolib.html
[33] S. Keller. (2009). *Cryptographic Algorithm Validation Program*. Accessed: Sep. 5, 2017. [Online]. Available: https://www.nist.gov/programs-projects/cryptographic-algorithm-validation-program
[34] M. Mielke. (2017). *Using the IDD Current Measurement Feature on the STM32L053 Discovery Board*. Accessed: Sep. 5, 2015. [Online]. Available: https://eewiki.net/display/microcontroller/Using+the+IDD+Current+Measurement+Feature+on+the+STM32L053+Discovery+Board
[35] Atollic. (2017). *TrueSTUDIO Pro—ARM Development Tools—Subscription FAQ*. Accessed: Sep. 4, 2017. [Online]. Available: http://info.atollic.com/pro-upgrade-faq

LEHLOGONOLO P. I. LEDWABA received the BCIS degree from Monash University, South Africa, in 2016, and the B.Sc. degree (Hons.) in computer science from the University of Pretoria, South Africa, in 2017, where she is currently pursuing the M.Sc. degree in applied sciences. She holds the Masters Studentship at the Council for Scientific and Industrial Research (CSIR), Pretoria, South Africa. Her research interests include security for cyber-physical systems, wireless sensor networks, and the industrial internet of things.

Ms. Ledwaba received membership from the Golden Key International Honours in 2016 for academic excellence.

GERHARD P. HANCKE (S'99–M'07–SM'11) received the B.Eng. and M.Eng. degrees from the University of Pretoria, South Africa, in 2002 and 2003, respectively, and the Ph.D. degree in computer science from the Computer Laboratory, Security Group, University of Cambridge, in 2008. He is currently an Assistant Professor with the City University of Hong Kong, Hong Kong. He is also an extraordinary Senior Lecturer with the University of Pretoria. His research interests include system security, embedded platforms, and distributed sensing applications.

HEIN S. VENTER (M'07) received the B.Sc. degree, the B.Sc. degree (Hons.), and the M.Sc. degree, and the Ph.D. degree from the University of Johannesburg (formerly the Rand Afrikaans University), South Africa, all in computer science. He is currently a Professor and the Research Group Leader with the Digital Forensic Science Research Group, University of Pretoria, South Africa, where he collectively supervises 30 Computer Science postgraduate students, including two B.Sc. (Hons.), seven M.IT., 12 M.Sc. and nine Ph.D. students. In his research career, over the past 16 years, he has established an international research reputation in cyber security and cyber forensics. Over the past nine years, he has been focusing mainly on cyber forensics research.

Prof. Venter has been a member of ACM since 2007 and the American Academy for Forensic Sciences since 2009. He is the General Chair of the Information Security for South Africa Conference.

SHERRIN J. ISAAC (M'04) was born in Ethiopia in 1967. He received the B.Eng. degree in electronics and telecommunications and the M.Eng. in electrical degree from the University of Zambia, Lusaka, Zambia, in 1988 and 1994, respectively, and the M.B.A. degree from the University of Pretoria, Pretoria, South Africa, in 2006.

He joined the University of Zambia, as a Project Engineer in 1989. Since 1998, he has been with the Council for Scientific and Industrial Research, Pretoria, where he is currently a Research Group Leader in embedded intelligent systems. His current research interests include cyber physical systems for smart infrastructure with respect to wireless sensor networks, security, distributed embedded artificial intelligence, real-time control, operations, and maintenance.

Mr. Isaac is a Professional Engineer in South Africa and a member of the South African Institute of Electrical and Electronic Engineers.