Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platforms

1st Mohammed El-Hajj
EEMCS (SCS)
Twente University
Enschede, Netherlands
m.elhajj@utwente.nl

2nd Ahmad Fadlallah
Faculty of Arts and Science, USAL
USAL University
Beirut, Lebanon
a.fadlallah@usal.edu.lb

2022 32nd International Telecommunication Networks and Applications Conference (ITNAC) | 978-1-6654-7103-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITNAC55475.2022.9998413

***Abstract*—Highly constrained devices that are interconnected and interact to complete a task are being used in a diverse range of new fields. The Internet of Things (IoT), cyber-physical systems, distributed control systems, vehicular systems, wireless sensor networks, tele-medicine, and smart grid are a few examples of these fields. In any of these contexts, security and privacy might be essential aspects. Research on secure communication in Internet of Things (IoT) networks is a highly contested topic. One method for ensuring secure data transmission is cryptography. Since IoT devices have limited resources, such as power, memory, and batteries, IoT networks have boosted the term "lightweight cryptography". Algorithms for lightweight cryptography are designed to efficiently protect data while using minimal resources. In this research work, we evaluate and benchmark lightweight symmetric ciphers for resource-constrained devices. The evaluation is performed using two widely used platform: Arduino and Raspberry PI. In the first part, we implement 39 block ciphers on an ATMEGA328p micro-controller and analyze them in terms of speed, cost, and energy efficiency during encryption and decryption for different block and key sizes. In the second part, the 2nd round NIST candidates (80 stream and block cipher algorithms) were added to the first part ciphers in a comprehensive analysis for equivalent block and key sizes in terms of latency and energy efficiency.**

***Index Terms*—IoT, Constrained devices, lightweight cryptography, Raspberry PI, Arduino.**

I. Introduction

The Internet of Things (IoT) security is a strongly contested research topic. IoT is a type of network paradigm that uses sensor and Internet technology to transform everyday items into smart devices[1]. Such devices give people the ability to be connected any *time*, any *where*, using any *connectivity* to benefit from wide spectrum of *services*[2]. So digitization is not an option anymore, where it is involved in our daily life activities including smart homes, smart cities, wearables, e-health, etc[3]. The IoT end-devices are often operating in vulnerable environments, which leads to several security challenges that should be taken into consideration [4]. To overcome such challenges, various researchers have developed different cryptographic algorithms that can be used to secure IoT applications in order to ensure data protection and data privacy. But traditional cryptographic algorithms are not suitable to be implemented in the resource-constrained devices used in such application. The concept Lightweight Cryptographic (LWC) schemes arose to reflect the need of cryptographic algorithms that provide security with the use of efficient amount of resources [5]. This resource usage is determined by the key size, the number of rounds, the block size, the memory usage (ROM and RAM), the structure, and execution time. The objective of lightweight algorithms creation is to strike a balance in several aspects such as performance, low resource demand, and cryptographic algorithm strength and stability [6]. Various LWC algorithms (stream or block ciphers, hash functions) such as TWINE, PRESENT, SIMON and SPECK, QARMA were proposed as alternatives to traditional cryptographic algorithms (e.g., AES, RSA, SHA3). In spite of the increasing demand in this research area, few research works presented the benchmarking and comparison of well-known LWC algorithms between different hardware platforms of constrained devices. Moreover, no article had presented yet involved software implementation with analysis and comparison of any lightweight cryptography on Raspberry-Pi compared to others. The objective of this research work is to provide a comprehensive benchmarking of well-known lightweight cryptographic algorithms. These benchmarking results are obtained through a software implementation of the selected algorithms, implemented on the micro-controller ATMEGA328P-Arduino (UNO) and the Raspberry PI.

To the best of our knowledge and based on the literature review, this work is considered the first to evaluate the performances of lightweight cryptographic schemes; 119 different schemes were evaluated.

The rest of this paper is structured as follows: section II provides a literature review about the work done related to the implementation of LWC using different hardware platforms. Section III details the software and hardware setup done, the measuring metrics used, and the methodology applied to evaluate the communication and computation cost of implementing such schemes. Section IV presents the results of the experimentation, and section V analyzes and discusses the results achieved. Finally, section VI concludes the paper.

II. Background

Research and development of lightweight cryptography for use on resource-constrained IoT devices have advanced quickly over the past decade. The main goal is to create and use simple cryptographic algorithms that may be applied to such applications while providing the appropriate levels of security. With a focus on LED [7], Piccolo [8], and PRESENT[9], authors in [10] investigated various software implementations of lightweight ciphers for x86 processors. First, they examined a table-based implementations and then offered a theoretical model to forecast how different potential trade-offs will behave in relation to the processor cache delay profile.

The authors in [11] studied the lightweight properties of HIGHT block cipher and offer the optimized implementations of both software and hardware for IoT platforms, such as resource-constrained devices (8-bit AVR and 32-bit ARM Cortex-M3) and Application-Specific Integrated Circuits.

The authors in [12] implemented six ciphers —AES, SIMON, SPECK, PRESENT, LED, and TWINE in hardware with Register Transfer Level (RTL) design [13] and in software with a specially designed re-configurable processor. They presented a direct comparison of area, throughput, power, energy, and throughput-to-area (TP/A) ratio. Both hardware and software versions were implemented in identical Xilinx Kintex-7 FPGAs.

SIMON is a lightweight block cipher designed for hardware implementation. Implementing, optimizing, and modeling SIMON cipher design for resource-constrained devices with a focus on energy and power were the goals of the research done by authors in [14]. Scalar and pipelines design implementations FPGA technology were the two types that were explored in this research[15].

The hardware implementation of the block cipher RECTANGLE with various data-paths was the focus of the authors in [16]. They have devised, constructed, and assessed the five most effective RECTANGLE[17] cipher data-paths for various data bus sizes. The same implementation conditions were used for all of these data paths when they were implemented on various FPGA platforms, and the results were compared across all performance metrics. The ideal architecture for an application can be chosen based on the device and desired performance metrics.

In [18], the authors used the Artix-7, Spartan-6, and Cyclone-V FPGAs to implement the six NIST LWC Round 2 candidate ciphers (SpoC, GIFT-COFB, COMET-AES, COMET-CHAM, Ascon, and Schwaemm and Esch). Among all the schemes, it was clear that SpoC had the lowest area and power consumption, while Ascon had the highest throughput-to-area (TPA) ratio.

Authors in [19] implemented KLEIN-80, TWINE-80, Piccolo-80, SPECK (64, 96), and SIMON (64, 96) lightweight block ciphers on the Atmega128 processor in the AVR studio 5.1 simulation environment. The evaluation's findings indicate that the SPECK(64,96) cipher was the most energy-efficient and is suitable for wireless sensor networks. While the implementation of the TWINE-80 was the best appropriate with regard to memory utilization.

Almost all cited works were interested in the implementation of specific LWC schemes, while in this work we did a software implementation of almost 119 different schemes and compared their performance using two different hardware platforms.

TABLE I: 39 algorithms and AES as a relative reference

Family-cipher#	Algorithm	Type	Structure	Block size(bits)	Key size(bits)	rounds
Relative reference	**AES**	Block Cipher	SPN	128	128/192/256	10/12/14
1-1	**HIGHT**	Block Cipher	Generalized Feistel structure (GFS)	64	128	32
2-2/3/4	**KATAN**	Block Cipher	stream-cipher-like	32/48/64	80	254
3-5/6/7	**KTANTAN**	Block Cipher	stream-cipher-like	32/48/64	80	254
4-8/9/10	**LEA**	Block Cipher	Generalized Feistel Network (GFN)	128	128/192/256	24/28/32
5-11	**Piccolo**	Block Cipher	GFN	64	80/128 only 80 chosen	25/31
6-12/13	**PRESENT**	Block Cipher	SPN	64	80/128	31
7-14	**PRINCE**	Block Cipher	SPN	64	128	12
8-15	**QARMA**	Block Cipher	SPN	64	64	27
9-16	RECTANGLE	Block Cipher	SPN	64	128	25
10-1726	SIMON	Block Cipher	Feistel	32128	64256	3272
11-2736	SPECK	Block Cipher	Addition/Rotation/XOR (ARX)	32128	64256	2234
12-37/38	TWINE	Block Cipher	type-2 Generalized Feistel Network (GFN-2)	64	80/128	36
13-39	XTEA	Block Cipher	ARX	64	128	64

TABLE II: Measurement of LWC and their metrics

Measurement	**Metrics (in)**	**Tool of Measuring for Arduino**	**Tool of Measuring for Raspberry Pi**
Key size	bits	Algorithm specs	Algorithm specs
Block size	bits	Algorithm specs	Algorithm specs
Rounds number	number of rounds	Algorithm specs	Algorithm specs
ROM occupation	bits or bytes	Arduino IDE	size command
RAM occupation	bits or bytes	Arduino IDE	Valgrind
Code size	Kbytes	size occupied on memory	size occupied on memory
Encryption (ENC) or Decryption (DEC) speed Throughput	Bytes/s	programming + Eq1	programming + Eq1
ENC or DEC speed Latency	cycle/Block	programming + Eq**??**	programming + Eq**??**
Key schedule speed Throughput	Bytes/s	programming + Eq2	programming + Eq2
Key schedule speed Latency	cycle/Block	programming + Eq4	programming + Eq4
ENC or DEC Power (Throughput)	joules/s	current (power) sensor	current (power) sensor
ENC or DEC Energy (Latency)	joules/bit	current (power) sensor + Eq2'	current (power) sensor + Eq2'

III. Method and Experimental Setup

A. *Algorithms used for Evaluation*

Lightweight Cryptography of various structures, key sizes, and block sizes have been chosen. A wide range of differences in key size, block size, and rounds were realized as essential for analysis goals. Because of that, the cited algorithms of C language found with these specifications were reached to 39 different ciphers of 13 families shown in Table I. Furthermore, a package of 80 algorithms of 32 families presented in NIST round 2 competition[20] have been included in an extended study.

B. *Compilation*

This study was based on C language implementation as low-language to reach an adequate elimination of any software barrier between algorithms implementation and execution. Hence, the study uses MinGW as a container of GNU compiler collection (GCC) and its libraries. It is used in this study in the Linux operating system of the Raspberry Pi platform. Arduino IDE was used for compilation and execution for the Arduino platform.

C. *Measuring Concepts and Metrics*

Measurement for Lightweight Cryptography study have been gathered depending on related works and other similar research studies that analyzed and compared Lightweight Cryptography ciphers. These measuring concepts and their metrics are summarized in Table II

Below is the briefing and the equations used in table II:

1) **Arduino IDE** is used during the uploading phase of C codes onto Arduino Board to measure:
   - ROM occupation: by observing "program storage space" where the Arduino sketch is stored.
   - RAM occupation: by observing the unused space for local variables, then the used space would indicate the "global variable" of dynamic memory that is the SRAM (static random-access memory), which is where the sketch creates and manipulates variables when it runs.
2) Encryption and Decryption speed **Throughput** are measured in bytes/s in both platforms through programming loops and equation 1.

$$\frac{Ps}{\tau}(bytes/s) \tag{1}$$

Ps = is the size of text in ENC or DEC in bits.
τ = is the time taken during one ENC or DEC.

3) Key schedule speed **Throughput** concerning key expansion with and equation 2.

$$\frac{Ks}{\tau}(bytes/s) \tag{2}$$

Ks = is the size of text in expansion in bits.

4) ENC and DEC speed **Latency** are measured in B/s (Cycles/Block) by using speed Throughput equation 2:

$$\frac{f}{(1)/Bs}(Cycles/Block) = \frac{f * Bs * \tau}{Ps}(Cycles/Block) \tag{3}$$

f = is the processor frequency in hertz.
Bs = is the Block size of the algorithm in bytes.

5) Key scheduling speed **Latency** derived from equation 2:

$$\frac{f * K * \tau}{Ks}(Cycles/Block) \tag{4}$$

K = Key size of the algorithm in bytes.

6) ENC and DEC Power (**Throughput**) measured in joules/s (j/s) by using a current sensor (power sensor).
7) ENC and DEC Energy (Power **Latency**) in joules/bit is measured by using Energy Throughput and speed Throughput as:

$$\frac{ETh(j/s)}{(1) * 8}(Jouls/bit) = \frac{ETh * \tau}{Ps * 8}(Jouls/bit) \tag{5}$$

ETh = Energy Throughput in j/s of ENC and DEC.

D. Methodology

1) The following **methods** were used while bench-marking the different metrics for the selected cryptographic algorithms:
   - **Different Cryptographic Algorithms, Same platform**: Comparing different Algorithms on the same platform is done by measuring the throughput in bytes per second that would be satisfactory.
   - **Same Cryptographic Algorithms, Different platform**
   - **Different Cryptographic Algorithms, Different platform**: The difference in Key size and Block among Cryptographic Algorithms, can be assessed using bytes per second as a metric of measurement in the same platform. However, this cannot be used in different platforms (and neither cycles per byte). Here comes the notion of using cycle per block as a comprehensive measure of comparison between them.
2) **Repeating the experiments**: For numerous distinctive reasons, it can be decently troublesome to obtain measuring results like time and speed accurately in a single iteration of coding. Frequently, internal clocks that the software or executed program can read, have some degree of asynchronous precision from the core processor clock. More essentially, there is regularly a critical overhead included in such measuring results, such as the cost of context switches and sometimes timing overhead. That is, finding measurement of algorithms in this context should avoid procedure call overhead. One method is to run the Algorithm many times like loops in coding then, averaging the total time to acquire the best indication of overall performance results. Furthermore, the repetition of encryption or decryption process would smooth out random effects like IRQ (Interrupt request) signal due to external activity by adjusting the loop to an experimental number that attained.
   On that account, the formulas of speed throughput of ENC and DEC followed by Key schedule formulas would be refined as:

$$\frac{Ps * Nl}{\tau}(bytes/s) \tag{1'}$$

$$\frac{Ks * Nl}{\tau}(bytes/s) \tag{2'}$$

Nl = is number of loops.

3) **Programming Libraries**: We have implemented the needed formulas in one programming library named **metrics.h** for both platforms. The **metrics.h** library contains the implementation of Equations: 1', 2', 3, 4, and 5 for exclusive grouping results of the Algorithms in one hand pack. In addition, changing the number of loops would be very easy using such a method for simplicity in work and other tasks.

4) **Hardware Platform**: we used two different hardware platform in this study for comparison and benchmarking purposes:
   - ATMEGA328P is a single-chip micro-controller of the megaAVR family with an 8-bit RISC processor core architecure. It is used in basic Arduino boards like Arduino UNO.
   - Raspberry Pi 3 Model B V1.2 is the third generation of Raspberry Pi.
   - **Ammeter(Power measurement)** used: Power (Energy) is measured through the use of the Adafruit INA219 current sensor. This measuring process requires another Arduino board to read the sensor data of measuring that is fed from the load to the platforms. The monitoring process is combined by PLX-DAQ software tool through EXCEL with the necessary Arduino programming.

IV. Results

This section presents the main benchmarking observations. For each metric, the best ten and the worst five performing algorithms are selected in a quick and brief overview. The focus will be on the following measurements:

- ROM, RAM, code size, Key Schedule speed throughput and Latency of the 39 ciphers.
- Number of rounds, Encryption/Decryption speed throughput/Latency and Energy throughput/Latency of all the 119 ciphers.

Besides, the measurement tools used are stated briefly for each metric.

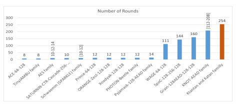
1) Analysis of the **Number of Rounds**: the range interval of the number of Rounds of the studied Algorithms is [8,254] as shown in Figure 1a . Among the five ciphers with the largest number of rounds, the Katan-Ktantan family is designed with the highest, while Ace-64-128 and TinyJambu family are with the least among the smallest 10.
2) Analysis of **Code Size**: According to the results in Figure 1b, the range intervals of code sizes are [3.6, 21] and [3.49, 17.3] in Kbytes for UNO and PI respectively. Of the 10 smallest code sizes, the least are Katan-32-80 and Present-64-128, whereas the biggest are Piccolo-64-80 and Rectangle-64-128 in UNO and PI respectively.
3) Analysis of **ROM Size**: The range interval is [1.48, 6.05] and [3.81, 9.67] in Kbytes for UNO and PI respectively. Figures 1c and 1d show the least cipher demanding ROM among the 10 smallest that are LEA family/Prince-64-128 and Present-64-80 respectively for UNO and PI. While the biggest demand is Piccolo-64-80 for both platforms.
4) Analysis of **RAM** Occupation: Figures 1e and 1f show the 10 smallest and 5 largest RAM size occupations in UNO and PI respectively. As it can be seen, the range intervals are [264, 994] in Bytes and [1.03, 21.34] in Kbytes respectively. The least are the LEA and the Simon-Speck families, and the largest are the Ktantan family and Rectangle-64-128 in UNO and PI respectively.
5) **Key Scheduling Speed**: Figures 1g and 1h exhibit the 10 largest and 5 smallest Algorithms for key schedule speed in UNO and Pi respectively. The range intervals are [0.99, 400000] in Kbytes/s and [0.029, 106.667] in Gbytes/s for Throughput. The best is Prince-64-128, and the worst is the Ktantan family in both platforms. The larger the throughput value is, the faster the algorithm is. Further, the range intervals for Latency are [0.00032, 161.34] in KCycles/Block and [0.18, 412.02] in Cycles/Block. The best is XTEA-64-128, and the worst is the Ktantan family in both platforms.
6) **Encryption and Decryption Speed**: Figures 1i and 1j reveal the 10 largest and 5 smallest Algorithms for ENC speed in UNO and PI respectively. The range intervals are [0.1,64.1] in Kbytes/s and [0.009,6.99] in Gbytes/s of UNO and PI respectively for Throughput. The best are Hight-64-128 and LEA-128-128 however the worst are Jumbo-128-128 and Ktantan-32-80 in UNO and PI respectively. Whereas, the range intervals for Latency are [2,2490.24] in KCycles/Block and [1.44,864.65] in Cycles/Block of UNO and Pi respectively. The best are Hight-64-128 and Speck-48-72 though the worst are Jumbo-128-128 and ISAP-K-128-64-128 in UNO and PI respectively.
7) Encryption and Decryption **Power and Energy Consumption**: Figures 1k and 1l show the 10 smallest and 5 largest Algorithms for ENC Power. The range intervals are [0.959, 14.28] in mj/s and [168,303] in mj/sfor Power. The least are SUNDAE-GIFT-0-128-128 and XTEA-64-128. However, the worst are Simon-128-256 and DryGASCON128k56-128-128 in UNO and PI respectively. Whereas, the range intervals for Energy are [0.097, 67.4] in µj/Byte and [35.4, 26870] in nanoj/ Byte (or [0.035, 26.87] in µj / Byte) of UNO and PI respectively. The least are SUNDAE-GIFT-0-128-128 and Speck-128-128 though the worst are Rectangle-64-128 and Ktantan-32-80 in UNO and PI respectively.

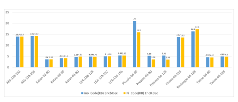
V. Discussion

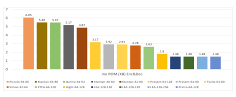
Some of the algorithms' measured speed and power Latencies (plus Key schedule) of Enc. They are presented in the following figures as a percentage of the relative Reference (RR) which is the AES. They are grouped in two ways: Block or Key sizes. Some groups with their corresponding figures are:

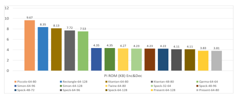
- 128 bits Block size and 96 bits Key size RR% groups of Key Scheduling speed Latency presented in Figures 2a and 2b.
- 32 bits Block size and 256 bits Key size RR% groups of ENC speed Latency presented in Figures 2c and 2d.
- 128 bits Block size (all of 128 bits key size) and 80 bits Key size RR% groups of ENC Energy (Power Latency) presented in Figures 2e and 2f.

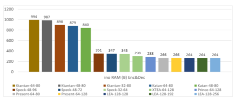
After adequate observation over all these grouping and their results between UNO and PI, it was realized that for more

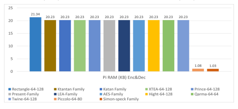

(a) Number of Rounds

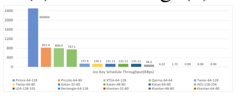

(b) Code Size

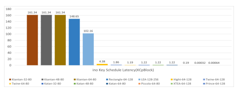

(c) ROM size1

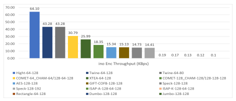

(d) ROM size(2)

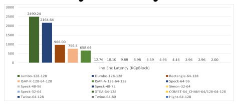

(e) RAM Usage(1)

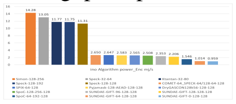

(f) RAM Usage(2)


(g) Throughput in Key Schedule


(h) Latency in Key Schedule


(i) Throughput Speed in ENC


(j) Latency Speed in ENC


(k) Power Consumption in ENC

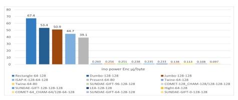
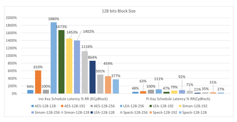
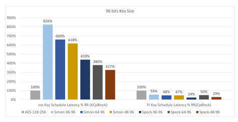
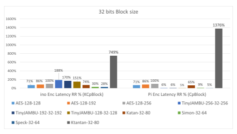
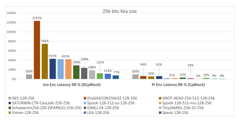

(l) Energy Consumption in ENC

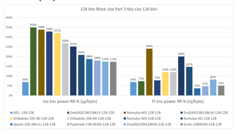
Fig. 1: Different measures for different metrics in UNO and PI platforms


(a) 128 Block size


(b) 96 Key Size


(c) 32 Block size


(d) 256 Key size


(e) 128 Block and Key sizes

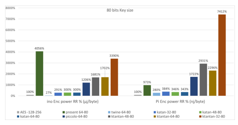
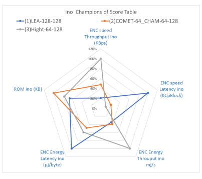
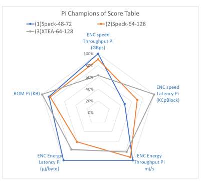
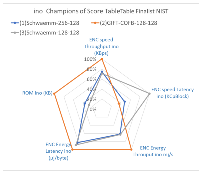

(f) 80 Key size

Fig. 2: Relative Reference (RR) and Block, Key Groups Arrangement


(a) Best 3 For UNO


(b) Best 3 For PI


(c) Best 3 for UNO

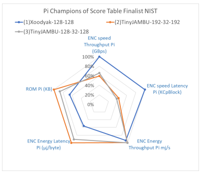

(d) Best 3 for PI

Fig. 3: NIST best Algorithms in UNO v.s. PI

than 85% of the algorithm throughout, the metrics used had gone better in PI. For example, Lea-128-192 was 1673% to RR in Key Schedule for UNO whereas it became 47% for PI. Statistically, in Key Schedule, there were 10 Algorithms below 100% for UNO while they became 33 Algorithms for PI, in ENC Speed Latency there were 28 Algorithms below 100% for UNO whereas the number became 79 Algorithms for PI, and in ENC Energy there were 33 Algorithms below 100% for UNO whereas the number became 65 Algorithms for PI. Consequently, Raspberry PI is revealing more Light weighting features and behavior of the designed Lightweight Algorithms that might be explained: "because of the Hardware and Software Architecture of the Raspberry Pi". The 67 winners in each group of Block or Key arrangement throughout Key Scheduling Speed Latency, ENC/DEC Speed Latency, and ENC/DEC Energy are listed in the Table 5. It can be concluded that 36 (84%) of 39 of the Algorithms are faster (fewer Cycles) in key schedule, 106 (94%) of 110 of the Algorithms are faster (fewer Cycles) in ENC, and 95 (86%) of 110 have taken less power in PI compared to UNO as shown in Table 6 with additional statistical comparison. Also, it was realized as a quick observation that most of the algorithms that gone worse in PI compared against UNO were of 80 bits Key like in Energy ENC, as shown in Figure 31. That may indicate reliance on 80 bits key in Algorithms design is not preferable, but this should take further investigation to deduce such a conclusion.

The overall comparison of Algorithms numbers and percentage in UNO v.s. PI is presented in TableIII.

A **Score Table** is presented in this section for some measuring metrics. The scores (or cards) are given to each Algorithm in each of the selected measures: ROM, ENC speed (throughput and Latency), and ENC Energy (throughput and Latency). The Algorithm with the least sum of all scores would be considered as the best. For the first list of algorithms presented in TableI, figures 3a and 3b clearly shows that LEA-128-128, OMET-64-CHAM-64-128, and Hight-64-128 are the best in UNO. While Speck-48-72, Speck—64-128, and XTEA-64-128 are the best in PI. Regarding the final list presented in Table**??**,figures 3c and 3d clearly show that Schwaemm-256-128, GIFT-COFB-128-128, and Schwaemm-128-128 are the best UNO. While Xoodyak-128-128, TinyJAMBU-192-32-192, and TinyJAMBU-128-32-128 are the best in PI.

TABLE III: Overall comparison of Algorithms numbers and percentage in UNO versus PI.

Condition	**Key Schedule speed UNO (39)**	**Key Schedule speed PI (39)**	**ENC speed UNO (110)**	**ENC speed PI (110)**	**Energy UNO (110)**	**Energy PI (110)**
Unit Normal Comparison	Kbytes / KCycles per s	Gbytes / Cycles per s	Kbytes / KCycles per s	Gbytes / Cycles per s	[0.097,67.4] µj/Byte	[0.035,26.87] µj / Byte
Overall sum of %	30,760	6,032	77,014	6,114	68,138	42,308
Overall sum of % decreases from UNO to PI	↘ 80%		↘ 92%		↘ 40%	
# of Algorithms decreases in RR %	3 (all of 64 Block size)	36 (92%)	6 (4 of key size 80)	106 (94%)	15 (8 of 9 of key size 80)	95 (86%)
# of Algorithms <100% in RR %	10 (25%)	33 (84%) (+ 23 added)	28 (25%)	79 (72%) (+ 51 added)	33 (30%)	65 (60%) (+ 32 added)
Overall sum of %	Is the sum of all the Algorithms percentage in RR					
↘	The number of Algorithms decreases					
<100%	Nnumber of Algorithms that are below 100% in RR					

VI. Conclusion

In conclusion, Lightweight cryptography is a challenging research domain through the last few years. In this work, a set of 119 ciphers had been evaluated and bench-marked using widely used platforms: Arduino and Raspberry PI. LEA-128-128, COMET-64-CHAM-64-128, Hight-64-128, Speck-48-72, Speck-64-128, and XTEA-64-128 were the most promising among the 119 compared Algorithms in power, speed, and ROM measurements. Furthermore, Schwaemm-256-128, GIFT-COFB-128-128, Schwaemm-128-128, Xoodyak-128-128, TinyJAMBU-192-32-192, and TinyJAMBU-128-32-128 are the best performing ciphers among the NIST Finalists[20].

References

[1] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[2] M. El-hajj, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of authentication techniques in internet of things (iot)," in *Cyber Security in Networking Conference (CSNet), 2017 1st*. IEEE, 2017, pp. 1–3.

[3] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A taxonomy of puf schemes with a novel arbiter-based puf resisting machine learning attacks," *Computer Networks*, vol. 194, p. 108133, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621002036

[4] M. El-Haii, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of cryptographic algorithms on iot hardware platforms," in *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 2018, pp. 1–5.

[5] T. Eisenbarth, Z. Gong, T. Güneysu, S. Heyse, S. Indesteege, S. Kerckhof, F. Koeune, T. Nad, T. Plos, F. Regazzoni, F.-X. Standaert, and L. van Oldeneel tot Oldenzeel, "Compact implementation and performance evaluation of block ciphers in attiny devices," in *Progress in Cryptology - AFRICACRYPT 2012*, A. Mitrokotsa and S. Vaudenay, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 172–187.

[6] N. F. Ibrahim and J. I. Agbinya, "A review of lightweight cryptographic schemes and fundamental cryptographic characteristics of boolean functions," *Advances in Internet of Things*, vol. 12, no. 1, pp. 9–17, 2021.

[7] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The led block cipher," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2011, pp. 326–341.

[8] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2011, pp. 342–357.

[9] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 450–466.

[10] R. Benadjila, J. Guo, V. Lomné, and T. Peyrin, "Implementing lightweight block ciphers on x86 architectures," in *International Conference on Selected Areas in Cryptography*. Springer, 2013, pp. 324–351.

[11] B. Kim, J. Cho, B. Choi, J. Park, and H. Seo, "Compact implementations of hight block cipher on iot platforms," *Security and Communication Networks*, vol. 2019, 2019.

[12] W. Diehl, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj, "Comparison of hardware and software implementations of selected lightweight block ciphers," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4.

[13] L. J. Hafer and A. C. Parker, "Register-transfer level digital design automation: The allocation process," in *15th Design Automation Conference*. IEEE, 1978, pp. 213–219.

[14] S. Abed, R. Jaffal, B. J. Mohd, and M. Alshayeji, "Fpga modeling and optimization of a simon lightweight block cipher," *Sensors*, vol. 19, no. 4, p. 913, 2019.

[15] B. H. Dwiel, N. K. Choudhary, and E. Rotenberg, "Fpga modeling of diverse superscalar processors," in *2012 IEEE International Symposium on Performance Analysis of Systems & Software*. IEEE, 2012, pp. 188–199.

[16] V. Dahiphale, H. Raut, and G. Bansod, "Design and implementation of novel datapath designs of lightweight cipher rectangle for resource constrained environment," *Multimedia Tools and Applications*, vol. 78, no. 16, pp. 23 659–23 688, 2019.

[17] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms," *Science China Information Sciences*, vol. 58, no. 12, pp. 1–15, 2015.

[18] B. Rezvani, F. Coleman, S. Sachin, and W. Diehl, "Hardware implementations of nist lightweight cryptographic candidates: A first look," *Cryptology ePrint Archive*, 2019.

[19] J. Hosseinzadeh and A. G. Bafghi, "Software implementation and evaluation of lightweight symmetric block ciphers of the energy perspectives and memory," *arXiv preprint arXiv:1706.03909*, 2017.

[20] M. S. Turan, K. McKay, D. Chang, C. Calik, L. Bassham, J. Kang, J. Kelsey *et al.*, "Status report on the second round of the nist lightweight cryptography standardization process," *National Institute of Standards and Technology Internal Report*, vol. 8369, no. 10.6028, 2021.