ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 776-782
doi: 10.18178/wcse.2019.06.115

A Lightweight Block Cipher Implementation in the Resource – Constrained Internet of Things

Roman Alex F. Lustro[1+], Ariel M. Sison[2], Jaydwin T. Labiano[3], and Ruji P. Medina[4]

[1]Technological Institute of the Philippines, Philippines
[2]Emilio Aguinaldo College, Philippines
[3]Isabela State University, Philippines
[4]Technological Institute of the Philippines, Philippines

Abstract. The lightweight block ciphers are created for the resource-constrained Internet of Things (IoT). Hence, their implementations on software and/or hardware should augment the resource application. Thus, this study investigates the implementation of the modified Speck on two Arduino platforms to determine its performance on software and hardware separately. Through the analysis and comparison of experimental data results, it was found out that the ModSpeck has 0.25% higher statistical test compared to NIST standard accepted success rate and only 4% slower than its original cipher due to the employment of key derivation function in terms of software implementation. On its hardware performance tests, it uses only 50% more energy and consumes only 1.8% more memory as compared to selected lightweight algorithms. Its randomness exhibits good competitiveness in various aspects such as high throughput, low energy consumption and remarkable figure of merit.

Keywords: Ciphers, Internet of Things, Resource-Constrained Devices, Performance Evaluation, Speck, Simon, AES.

1. Introduction

The resource-constrained Internet of Things (IoT) is gaining much attention in the digital world as there has been an exponential growth in the world, especially in telehealth, manufacturing and in urban areas to form smart cities. It is expected to connect 30 billion devices and is predicted to create about USD 7.1 trillion contributions to the global economy by 2020[1]. The use of this resource-constrained technology will not only make better the quality of life of people but also play a part in the world economy [2].

The concept of the resource-constrained Internet of Things has tied in closely with the popularization of home automation [3] and has been increasing greatly in recent years due to much higher affordability and simplicity through smartphone and tablet connectivity. The integration of information technologies with the home environment, systems and appliances are able to communicate in an integrated manner which results in convenience, energy efficiency, and safety benefits. As we are using Arduino IoT devices which is a popular open-source single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible [4].

However, resource-constrained IoT devices are vulnerable to a massive amount of security and privacy issues, which are known to the manufacturers which are either neglected or treated as a second thought [2]. The careful balancing of security level and performance metrics is a challenging task for the implementation of a lightweight cipher in either software or hardware. Lightweight ciphers implementation on software minimizes memory footprint and a number of execution cycles while hardware implementation must optimize power, energy, area, and speed [5]. Security assurance has become a pivotal concern because of the

+ Corresponding author. Tel.: +639354000214
E-mail address: theisuguru@gmail.com

large amount of heterogeneously interconnected devices which demand immediate attention to make IoT vision a reality.

Therefore, this paper investigates the implementation of the modified Speck on two Arduino platforms: *UNO and Mega* to determine its performance on software and hardware separately and contrasted with the lightweight block ciphers: Speck, Simon [6] and the industry standard AES [7]. That is, for every candidate cipher and the two chosen implementations, the Figure of Merit (FOM) was computed as the average performance value on the two devices. The FOM is the result of higher throughput and lower efficiency, both pointing to a better overall performance of the block ciphers implementation.

2. The Modspeck Lightweight Block Cipher

The researchers in [8] addressed the pressing security problem of the Speck lightweight block cipher based on the study of [9] by enhancing its key scheduling through the introduction of a key derivation process as depicted in Fig. 1. This serves as the algorithms key generator and uses for key input in the key scheduling and encryption process. The result shows that ModSpeck leaves Speck behind in terms of randomness as its three variations successfully passed all the statistical analysis test of NIST [10] and DieHarder [11] making it cryptographically secured. Furthermore, the researchers recommend that it is deemed important to implement the ModSpeck in a resource-constrained device such as Arduino.

```
Key Derivation Process
PreKey ← P1 ⊕ P2|| P2 ⊕ P3|| P3 ⊕ P4|| P4 ⊕ P1'
Q ← PreKey = 0..n
for p = 0..a
k ← Pa+1 >>>aŋ
end for

Key Expansion
for i = 0..T-2
ℓ [i+m-1] ← (k[i] + S-α ℓ[i]) ⊕ i
k[i+1] ← Sβ k[i] ⊕ ℓ[i+m-1]
end for

Encryption
for i = 0..T-1
x ← S-αx+ y)⊕k[i]
y ← Sβy⊕ (S-αx+ y) ⊕k[i]
end for

Where:
n = word size (32, 48, 64)
m = number of key words(must be 3 or 4 if n = 32, 2 or 3 if n = 48, 2 or 3 or 4 if n= 64)
T = number of rounds = 26 or 27 if n = 32, m = 3 or 4
                     = 28 or 29 if n = 48, m = 3 or 4
                     = 32, 33, or 34 if n = 64, m = 3 or 4
(α, β, ŋ) = (8, 3, 7)
x,y = plaintext
r,p = tempkeygen
ℓ[m-2].. ℓ[0], k[0] = key words
Q = rlut
```

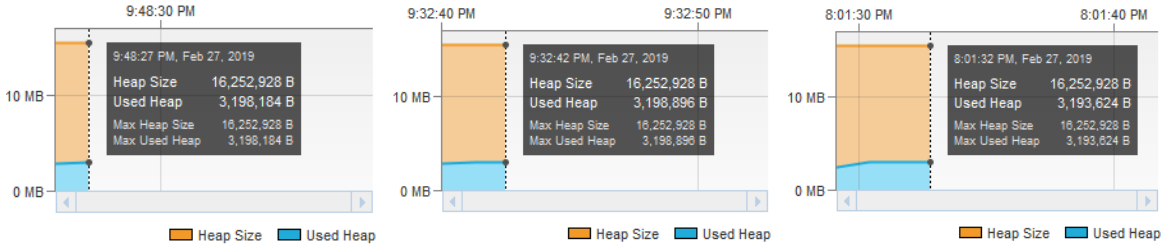
Fig. 1: The Algorithm of k – SPECK Key Derivation, Key Scheduling and Encryption Process

3. Experimental and Result Analysis

3.1. Methodology

One of the most basic design issues for resource-constrained devices is the power consumption; hence this study optimizes power by reducing the source code memory footprint and simplify the cipher cryptographic functions while preserving adequate security level. The code optimization was performed at the Java language level. The configuration of the system used for the experiment are as follows: Version: Intel (R) Core (TM) i5 – 7200U CPU @ 2.50GHz 2.70GHz RAM @ 4GB System: 64 – bit operating system, x64 bit processor. The researchers focused on cipher java codes in such a way that the memory size is

optimized by loop scheduling, variable, and memory partition. Optimization is a program conversion system, which attempts to identify the memory leak and increase the performance of the code with less compilation time [12]. Here, the researchers use the profiling feature of NetBeans IDE 8.2, which is extremely useful to identify the slowest portions of the program. As reflected in Fig. 2, it is noticeable that the surviving generation is stable, indicating that there is no memory leak in ModSpeck. After performing the profiling, the Java codes were ported in two Arduino (like .ino file) platforms with the corresponding parameters as denoted in TABLE 1 to accommodate the optimized code in the above resource-constrained devices.



(a) (b) (c)

Fig. 2: Memory profiling of the three variants (a) 128/128, (a) 128/192, (c) 128/256 of ModSpeck using NetBeans IDE 8.2

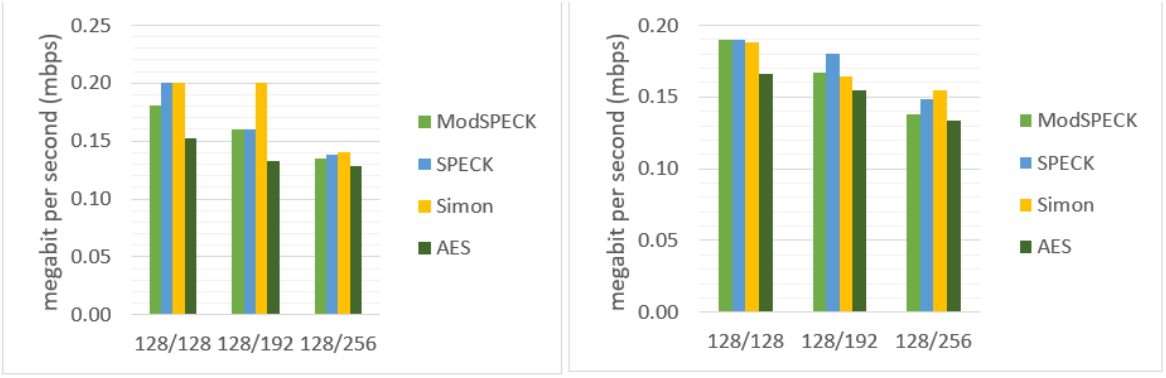
Table 1: Arduino Uno and Mega parameters

Parameters	**Uno**	**Mega**
Processor	ATmega 328P	ATmega2560
Clock Speed (MHz)	16	16
Flash Memory (kB)	32 kB	256 kB
EEPROM (kB)	1	4
SRAM (kB)	2	8
Voltage Level	5V	5V

3.2. Software performance

To measure the software performance of ModSpeck in terms of throughput denoted in megabits per second using (1) based on the study of [13], the following parameters were considered. *Block Size:* it is simply the length of the input and output size in bits that can be transferred by the keyed block cipher. *Execution Time:* refers to the number of processor cycles spent executing a set of instructions. The number of processor cycles is given by the number of cycles of the processor's clock. *Frequency:* the clock speed on how many cycles a process can complete in one second. The throughput is usually expressed to describe the number of processed bits per second, depending on the processor's frequency.

$$\text{Throughput}_{mbps} = \frac{\text{BlockSize}_b}{\text{Execution Time}_{ns}} * \text{Frequency}_{MHz} \quad (1)$$



(a) (b)

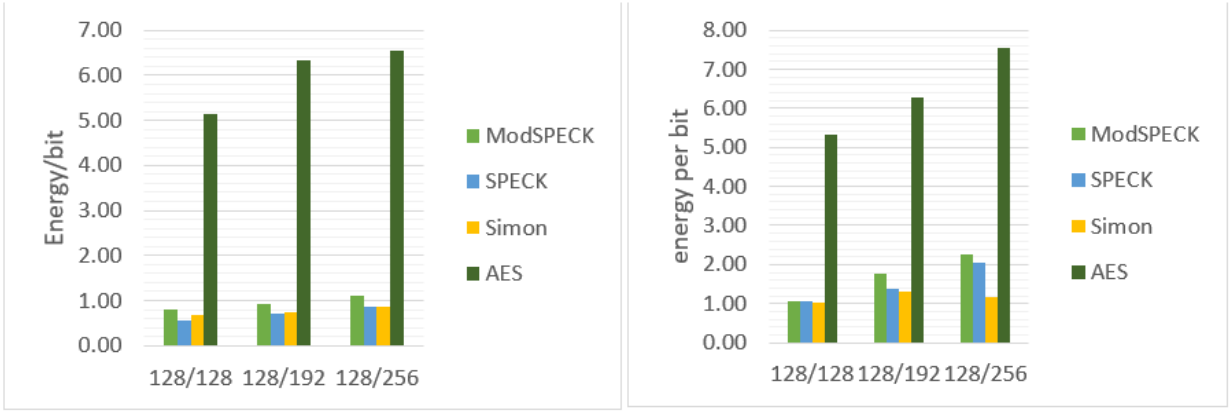
Fig. 3: Throughput performance of the different block ciphers implemented in Arduino (a) UNO, (b) Mega boards

Fig. 3 shows that ModSpeck achieves the same throughput in the 128/192 variant while standing a little bit behind 128/128 and 128/256 variants of Speck when implemented in Arduino Uno. Similarly, the same result on throughput was achieved on the 128/128 variant as contrasted to Speck when implemented in Arduino Mega. It is primarily because of bigger code size, ram consumption and execution time due to the introduction of a key derivation function.

3.3. Hardware performance

For the hardware performance metric, power consumption (*in joules*), the execution time (*in nanoseconds*), the frequency (MHz) and the block size (bits) are used to compute the energy per bit consumption of the ciphers. The power consumption was obtained using the java Running Average Power Limit (jRAPL) [14] which user's interface is simple, that is, for any block of code in the application whose energy performance information is to the interest of the user, simply enclose the code block with a pair of *statCheck* invocations to measure the energy consumption of the *doWork* method. The value is the difference between beginning and end. To measure the energy efficiency performance, (2) was used as referenced in the study of [15].

$$\text{Efficiency}_{energy/bit} = \left(\frac{\text{Power}_{joule} * \text{ExecutionTime}_{ns}}{\text{Frequency}_{MHz} * \text{BlockSize}_{bit}}\right) \quad (2)$$



(a) (b)

Fig. 4: Efficiency performance of the different block ciphers implemented in Arduino (a) UNO, (b) Mega board

Fig. 4 shows that the ModSpeck consumed the higher but minimal only amount of energy/bit when compared with Speck and Simon on both Arduino Uno and Mega implementation. Nevertheless, AES consumed the highest energy/bit because of the design of the cipher as the operations used are not elementary which should be the primary element of a lightweight block cipher. This increase in energy consumption is the tradeoff between the cipher's security and its hardware performance.

3.4. The Figure of Merit (FOM)

The performance parameter $P_{i,d}$ for every implementation $_i$, and device $_d$ were computed based on the study of [15]. The obtained value $P_{i,d}$ from (3) merges the three parameters M = {RAM consumption, code size, execution time} respectively:

$$Pi, d = \sum_{m \in M} W_m \frac{V_{i,d,m}}{\min(V_{i,d,m})} \quad (3)$$

the $V_{i,d,m}$ denotes the cost of the metric m on the device d for the implementation i; W_m is the proportional weight of metric m and $\min_i$ ($V_{i,d,m}$) is the minimum value of the metric m from full treated implementations of all treated ciphers on the same target platform d. Where: W_m = 1 and choose the implementation with the lowest $P_{i,d}$ for every target device and every cipher. In addition, for every candidate lightweight block cipher and the two chosen implementations i_1, i_2, the Figure of Merit (FOM) was computed using the (4) as the average performance value on the two devices. The FOM is the result of higher throughput and lower efficiency, both pointing to a better overall performance of the block ciphers implementation. Thus, the lower the FOM, the better the cipher is.

$$\text{FigureofMerit(FOM)}_{i_1,i_2} = \frac{P_{i_1,\text{UNO}+} P_{i_2,\text{MEGA}}}{2} \quad (4)$$

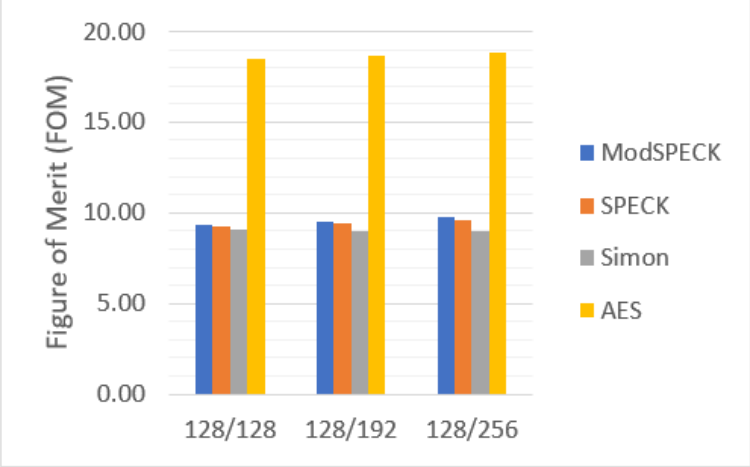


Fig. 5: Comparison of the figure of merit (FOM) of the different ciphers implemented in two Arduino board

In the design of lightweight cryptography, the balance between cost, security, and performance should be complied with. Generally two of these three objectives, such as security and cost, security and performance, or cost and performance can be straightforwardly attained; however, attaining all the three objectives simultaneously is very difficult [16],[17]. Hence, this paper able to achieve higher security at the same cost with a minimal increase in performance. The minimal increase in the figure of merit of the ModSpeck as shown in Fig. 5 still makes it the better choice for implementation in a resource-constrained IoT as compared to Speck since it is more cryptographically secured.

3.5. Discussion

Table 2: Summary on the overall performance evaluation of the different lightweight block ciphers implementation on Arduino boards.

ALGORITHM		UNO						MEGA						Figure of Merit (FOM)
		Global Variables (kb)	Code Size (kb)	RAM (bits)	Execution Time (Cycles)	Throughput (mbps)	Efficiency (energy/ bit)	Global Variables (kb)	Code Size (kb)	RAM (bits)	Execution Time (Cycles)	Throughput (mbps)	Efficiency (energy/ bit)	
128/128	ModSPECK	0.535	18.224	3136	11355	**0.18**	**0.79**	0.535	20.520	3136	10690	**0.19**	**1.08**	**9.34**
	SPECK	0.397	17.952	3120	10353	**0.20**	**0.56**	0.397	20.368	3120	10797	**0.19**	**1.07**	**9.25**
	Simon	0.397	16.917	3040	10787	**0.19**	**0.66**	0.397	17.683	3040	10543	**0.19**	**1.04**	**9.05**
	AES	0.505	68.624	6544	13445	**0.15**	**5.14**	0.505	68.600	6544	12321	**0.17**	**5.33**	**18.51**
128/192	ModSPECK	0.535	18.224	3136	13010	**0.16**	**0.91**	0.535	20.520	3136	12270	**0.17**	**1.77**	**9.50**
	SPECK	0.397	17.952	3120	12846	**0.16**	**0.71**	0.397	20.368	3120	11343	**0.18**	**1.39**	**9.40**
	Simon	0.397	16.917	3040	10435	**0.20**	**0.75**	0.397	17.683	3040	10432	**0.20**	**1.30**	**9.03**
	AES	0.505	68.624	6544	15433	**0.13**	**6.33**	0.505	68.600	6544	13216	**0.15**	**6.30**	**18.65**
128/256	ModSPECK	0.535	18.224	3136	15226	**0.13**	**1.10**	0.535	20.520	3136	14816	**0.14**	**2.26**	**9.73**
	SPECK	0.397	17.952	3120	14898	**0.14**	**0.85**	0.397	20.368	3120	13766	**0.15**	**2.05**	**9.62**
	Simon	0.397	16.917	3040	10234	**0.20**	**0.87**	0.397	17.683	3040	10221	**0.20**	**1.16**	**9.01**
	AES	0.505	68.624	6544	16007	**0.13**	**6.54**	0.505	68.600	6544	15321	**0.13**	**7.55**	**18.79**

TABLE 2 depicts the overall summary of the computed performances using the different metrics on the studied lightweight block ciphers together with the conventional and industry standard AES since it is the point of reference among the different ciphers developed nowadays due to its impeccable security advantages and execution on both software and hardware. All the variants of the Simon cipher achieve the best overall performance in the figure of merit primarily because it is designed for hardware implementation. The above presentations are surely what the target of low-cost devices is expected, and the less space occupied, the wider scope the applications have, not setting aside the ModSpeck in this standpoint.

4. Conclusion and Recommendation

In this study, the researchers investigate the implementation of the modified Speck on two Arduino platforms and determine its performance on software and hardware individually. The throughput in megabits per second was obtained from the software implementation while hardware emphasizes its efficiency on

energy per bit consumption. Even though it is noticeable that Speck and Simon achieved better Figure of Merit as compared to ModSpeck but still making the latter on the footprints of a lightweight block cipher far at par with AES based on the performance comparison. This research showed the development of ModSpeck, a more cryptographically secured lightweight block cipher and its implementation on resource-constrained devices without compromising the speed, energy, and memory consumption. The latter is the vital prerequisite in the development of lightweight cryptography suitable for the resource-constrained internet of things.

Having proved that the ModSpeck achieved exemplary performance in the two resource-constrained devices, the researchers recommend that the said algorithm be considered and utilized as a new standard for lightweight cryptography. Furthermore, as the result of this study showed that due to the addition of KDF, cipher's speed is compromised. Then, it is suggested that the modified algorithm can further be analyzed to optimize through its key derivation function's execution to lessen speed repercussion on its methods and processes.

5. References

[1] A. G. Priyanka Jain, "Review of IoT Market and Open Source Technologies in IoT," *Int. J. Eng. Res. Comput. Sci. Eng.*, vol. 5, no. 2, pp. 0–5, 2018.

[2] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *J. Netw. Comput. Appl.*, vol. 125, no. November 2018, pp. 251–279, 2019.

[3] N. Barodawala, B. Makwana, Y. Punjabi, and C. Bhatt, "Home Automation Using IoT," *Int. J. Smart Home*, vol. Vol. 11, N, no. December, pp. 219–242, 2017.

[4] P. Kauthale, A. Dhanapune, and P. S. N. Patil, "IOT Based Home Automation system using Raspberry Pi-3," *Int. Res. J. Eng. Technol.*, vol. 4, no. 1, pp. 1541–1544, 2017.

[5] T. Hayajneh, Z. A. Khalaf, B. J. Mohd, K. M. Ahmad Yousef, and M. Z. A. Bhuiyan, "Hardware design and modeling of lightweight block ciphers for secure communications," *Futur. Gener. Comput. Syst.*, vol. 83, pp. 510–521, 2017.

[6] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "Simon and Speck: Block Ciphers for the Internet of Things," *Proc. 52nd Annu. Des. Autom. Conf. - DAC '15*, no. July, pp. 1–6, 2015.

[7] J. Daemen and V. Rijmen, *The Rijndael Block Cipher: AES Proposal*. 2003.

[8] R. A. F. Lustro, A. M. Sison, and R. P. Medina, "Performance Analysis of Enhanced SPECK Algorithm," in *ICIBE' 18 Proceedings of the 4th International Conference on Industrial and Business Engineering*, 2018, pp. 256–264.

[9] A. A. Chew, L C N; Shah, I N M; Adbullah, N A N; Zawawi, N H A; Rani, H A; Zakaria, "Randomness Analysis on Speck Family Of Lightweight Block Cipher Cryptography Development Department , Cyber Security Malaysia ," *Int. J. Cryptol. Res.*, vol. 5, no. June 2013, pp. 44–60, 2015.

[10] L. E. Bassham *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," no. April, 2010.

[11] R. G. Brown, "Robert G. Brown's General Tools Page." [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php. [Accessed: 13-Aug-2018].

[12] S. Deb and B. Bhuyan, "Performance evaluation of Grain family and Espresso ciphers for applications on resource constrained devices," *ICT Express*, vol. 4, no. 1, pp. 19–23, 2018.

[13] W. Diehl, F. Farahmand, P. Yalla, J. P. Kaps, and K. Gaj, "Comparison of hardware and software implementations of selected lightweight block ciphers," *2017 27th Int. Conf. F. Program. Log. Appl. FPL 2017*, 2017.

[14] K. Liu, G. Pinto, Liu, and Y. David, "Data-Oriented Characterization of Application-Level Energy Optimization," in *Fundamental approaches to software engineering: 18th International Conference, FASE 2015 held as part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015*, 2015, vol. 9033, pp. 316–331.

[15] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großsch, and A. Biryukov, "Triathlon of Lightweight Block Ciphers for the Internet of Things," *Univ. Luxemb.*, 2015.

[16] M. Pourghasem, E. G. Sheikhloo, and R. E. Atani, “Light Weight Implementation of Stream Ciphers for M-Commerce Light Weight Implementation of Stream Ciphers for M-Commerce Applications,” no. November 2014, 2014.

[17] C. Pei, Y. Xiao, W. Liang, and X. Han, “Trade-off of security and performance of lightweight block ciphers in Industrial Wireless Sensor Networks,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2018, no. 1, 2018.